Multimed Tools Appl
DOI 10.1007/s11042-011-0840-z

A survey of visual sensor network platforms

Bulent Tavli · Kemal Bicakci · Ruken Zilan · Jose M. Barcelo-Ordinas

Abstract Recent developments in low-cost CMOS cameras have created the opportunity of bringing imaging capabilities to sensor networks. Various visual sensor platforms have been developed with the aim of integrating visual data to wireless sensor applications. The objective of this article is to survey current visual sensor platforms according to in-network processing and compression/coding techniques together with their targeted applications. Characteristics of these platforms such as level of integration, data processing hardware, energy dissipation, radios and operating systems are also explored and discussed.

Keywords Visual sensor networks · Embedded systems · Vision platforms · Image acquisition · System integration · IEEE 802.15.4 · IEEE 802.11 · Embedded system interfaces

1 Introduction

Wireless Sensor Networks (WSNs) are becoming a mature technology after a decade of intensive worldwide research and development efforts. WSNs primary function is to collect and disseminate critical data that characterize physical phenomena around the sensors [7, 31, 55]. Availability of low-cost CMOS cameras has created the opportunity to build low-cost Visual Sensor Network (VSN) platforms able to capture, process, and disseminate visual data collectively [44]. Emerging applications such as visual surveillance and vehicle traffic monitoring can be enriched with visual data.

B. Tavli (✉) · K. Bicakci
TOBB University of Economics and Technology, Ankara, Turkey
e-mail: btavli@etu.edu.tr

K. Bicakci
e-mail: bicakci@etu.edu.tr

R. Zilan · J. M. Barcelo-Ordinas
Technical University of Catalonia, Barcelona, Spain

R. Zilan
e-mail: rzilan@ac.upc.edu

J. M. Barcelo-Ordinas
e-mail: joseb@ac.upc.edu

Published online: 01 July 2011

High-resolution images and videos require complex compression and coding algorithms and high bandwidth usage. These requirements imply that visual sensor nodes dissipate significantly higher power than scalar sensor nodes. Furthermore, sensors used in relaying data traffic have to be capable of buffering large number of packets. To reduce the bandwidth utilization, vision-processing techniques capable of reducing the amount of traffic by intelligently manipulating the raw data can be used at the cost of allocating more computation resources.

Main motivations of this study are two-folds. First, we discuss and compare VSN platforms based on processing and vision computing techniques. VSNs are distinguished from WSNs by their capability to present information-rich visual data, however, effective use of visual data depends on the use of appropriate processing techniques. There are differences in the techniques implemented or realizable in current VSN platforms because technical challenges involve many design trade-offs and decisions are usually enforced by hardware limitations and energy constraints. Second, we investigate the capabilities and limitations of VSN platforms in detail. We believe that protocol, algorithm, and application development for VSNs cannot be of practical use unless the underlying enabling technologies and platforms are well understood. Other surveys (such as [5, 44]) give a more general perspective on VSNs.

We refer readers interested in specific VSN topics which are not in the scope of our survey to other relevant surveys in the literature written on various aspects of VSNs (general overview [44], challenging issues [5], multimedia streaming [29], security [16], cross-layer design [51]).

The rest of this paper is organized as follows: Section 2 presents the reasons for designing special platforms for VSN applications. Section 3 provides an overview of image compression, video coding, and vision computing techniques used in VSN platforms. Section 4 presents an overview of the currently available VSN platforms. Section 5 categorizes these platforms according to integration level, data processing hardware, energy dissipation, radios, operating system and software architecture, hierarchy, and applications. We present a critical evaluation of VSN platforms and discuss open problems in Section 6. Section 7 presents conclusions of this survey.

2 Platform design for VSNs

The main differences between WSNs and VSNs are in acquiring, processing, and transferring data. Visual data requires higher bandwidth usage due to the amount of data to be transmitted and higher power consumption due to the complexity of coding and vision processing algorithms and high volume data acquisition. The spectrum of WSN platforms ranges from lightweight platforms (e.g., Mica2) through intermediate platforms (e.g., Yale XYZ) to PDA-class platforms (e.g., Intel Stargates) [20]. Table 1 presents the features of several WSN platforms. Lightweight platforms (e.g., Mica2, Mica2Dot, MicaZ, and Telos) are highly resource-constrained; thus, they are only suitable for simple sensing and detection tasks. The Yale XYZ platform, which is a typical example of intermediate platforms, has more memory and processing resources than the lightweight platforms. At the higher performance set, there is PDA-class platforms (e.g., Stargates) which are more powerful than the intermediate platforms but also consume more power.

Vision computing techniques can reduce the amount of visual data to be transmitted at the cost of more computation. To illustrate this fact, let us consider an application that controls registration plates of vehicles traveling in an urban area. Such application is useful for the implementation of local administration policies to reduce the amount of pollution, by alternating the circulation of vehicles based on the parity of their number plates (this kind of a policy has been introduced for the Beijing Olympic Games). It is possible to use a

Table 1 Comparison of wireless sensor network platforms

Mote	Microcontroller	Data memory	Storage memory	Radio	Data rate
Mica2	ATmega128l (8-bit)	4 KB	512 KB	CC1000	38.4 Kbps
Mica2Dot	ATmega128l (8-bit)	4 KB	512 KB	CC1000	38.4 Kbps
MicaZ	ATmega128l (8-bit)	4 KB	128 KB	CC2420	250 Kbps
Tmote Sky	MSP430F (16-bit)	10 KB	1,024 KB	CC2420	250 Kbps
Imote	ARM7 (32-bit)	11 KB	–	Zeevo TC2001	723.2 Kbps
XYZ	OKI ML67Q5002 (32-bit)	32 KB + 2 MB RAM	256 KB	CC2420	250 Kbps
Stargate	Intel XScale (32-bit)	64 MB	32 MB	IEEE 802.11b CC2420	1–11 Mbps 250 Kbps

VSN for monitoring the vehicle plates in three different ways: (a) capturing real-time video and sending the data to the base station, which requires real-time video compression and large bandwidth; (b) taking still images of the vehicles and sending the images to the base station, which requires image compression and vision computing (i.e., detection of vehicles); (c) extraction of the plate numbers from the captured still images and sending these to the base station, which requires sophisticated vision computing.

Direct utilization of WSN platforms for vision applications is not feasible due to several facts itemized as follows:

- Lightweight WSN platforms are not capable enough in terms of computational power and memory for video acquisition and vision computing tasks; hence, direct use of these platforms for VSN applications is not feasible. Furthermore, the radios used with lightweight platforms do not have enough bandwidth to support streaming video with reasonable quality.
- Intermediate WSN platforms (e.g., XYZ node) have better computational capabilities; however, they are also equipped with low bandwidth radios (e.g., IEEE 802.15.4 compliant CC2420). Thus, real-time video streaming is also not possible with these platforms. The alternative for real-time video streaming relies on the use of IEEE 802.11 radio networks that can support bandwidths of an order of magnitude higher than IEEE 802.15.4.
- PDA-class platforms equipped with IEEE 802.11 radios are suitable for real-time video streaming due the highly capable processors and large memory available on board. However, the energy dissipations of these devices are more than an order of magnitude higher than the energy dissipations of lightweight platforms (e.g., Meerkats [3]). Furthermore, the energy consumption for communication also increases with the bandwidth (e.g., CC2420 consumes roughly 50 mW and IEEE 802.11b radios consume more than 1 W).

Combination of powerful-enough and energy-efficient hardware with in-network processing techniques is mandatory for vision-based applications. Processing costs, given as joules per instruction, are approximately two to three orders of magnitude lower than communication costs that are given as joules per bit, on available embedded platforms, such as Mica2 and Yale XYZ. However, there are numerous processing options to choose from as the vehicle plate monitoring application discussed above exemplified. Next, we

will review image compression, video coding and vision computing techniques with an emphasis on their use in VSN platforms.

3 Compression, coding, and vision computing in VSN platforms

Vision is the most important functionality provided by VSNs. Efficient and effective utilization of visual data depends on the intelligent use of vision computing techniques, along with image compression and video coding. Because of energy efficiency considerations, VSN platforms are designed with limited hardware capabilities; hence, vision-computing in VSN platforms is highly constrained and very challenging. In this section, we provide an overview of coding and compression algorithms together with in-network processing techniques in the context of VSN platforms and applications. In the next section, while providing a description of each VSN platform in detail we indicate targeted application(s) for each of these platforms together with information on processing techniques used.

3.1 Compression and coding

Transform coding also known as intra-coding (e.g., Discrete Cosine Transform—DCT and Wavelet Transforms—DWT) is a method used for lossy compression of still images and video. Intra-coding typically achieves poor compression for video since it does not exploit temporal correlation although it has high robustness. For video compression, transform coding is combined with motion-compensated prediction (inter-coding). One of the most used techniques is block-based predictive motion compensation technique. Inter-coding achieves high compression at the cost of high complexity. This means that one or more frames should be stored at the encoder and the decoder. Furthermore, predictive motion compensation [39] suffers from sensitivity to synchronization between coder and decoder. Error prone wireless channels often cause packet losses affecting the integrity of data and inducing prediction mismatches. Thus, the use of predictive motion-compensated coding techniques necessitates powerful computation and ample storage capabilities, which are not well suited for VSN platforms. A full search block motion estimation algorithm incurs around 65,000 operations per pixel per second for a 30 fps (frames per second) video as reported in [39]. To illustrate the communication and computation energy dissipation terms we use the example presented in [38][1]: processing with QCIF (176×144) resolution and 8-bit/pixel coding would cost the energy equivalent of transmission of approximately 4,400 Kb and processing with CIF (352×288) resolution and 8-bit/pixel coding would cost the energy equivalent of transmission of approximately 17,600 Kb.

Color provides multiple measurements at a single pixel of the image. The 24-bit RGB encoding uses 3 bytes for each basic color (red-green-blue) enabling $(2^8)^3$ colors. Other schemes use 15-bit RGB (5 bits for each color component) or 16-bit RGB with emphasized green sensitivity. YCbCr encoding, typically used in JPEG and MPEG coders, allows a better representation for image storage and transmission. The amount of data to be sent becomes larger as the color-coding technique becomes richer. For example, a sensor using CIF (352×288) resolution with 512 KB FLASH memory can store 41.37 1-bit (black and

[1] The energy cost of transmitting 1 Kb of data to a distance of 100 m, assuming Raleigh fading channel, BPSK modulation, 10^{-6} BER and fourth power distance loss is 3 Joules, which is approximately the same energy cost as of executing 3 million instructions by a 100 million instructions per second (MIPS)/W processor.

white) frames, 5.17 8-bit color frames or 2.58 16-bit color frames. There is, thus, a clear trade-off between sensor capabilities (i.e., buffering, computation, and energy) and resolution, coding, compression, and video frame rate (see Fig. 1).

Most of the cameras and imagers allow the choice of a wide range of resolutions and color encodings. Table 2 shows image resolutions and encoding and Table 3 shows compression algorithms used in VSN platforms. Below, we provide a general discussion and comparison of VSN platforms based on compression and coding techniques. Detailed descriptions of VSN platforms are specified in Section 4.

The behavior of JPEG compression is investigated in most of the platforms. For example, in [42] it is shown that by using the FireFly Mosaic platform, the JPEG processing time varies little with the quality level but greatly with the resolution of the image. Other platforms, like Panoptes [14] are designed with more powerful hardware (Intel ARM 206 MHz) and able to support more complex compression algorithms. Panoptes designers compare a standard JPEG compression algorithm (ChenDCT) with their own algorithm optimized for the Panoptes platform. They showed that optimized compression could be performed with significantly lower energy (i.e., they reported only execution times, but assuming similar power per unit time, longer execution time implies more energy dissipation). Cyclops designers also report the implementation of their own JPEG compression algorithm, showing that their JPEG implementation consumes only 105.5 μJ to compress the test images, while the standard integer implementation from the Independent JPEG Group takes 133.15 μJ to compress the image with the same amount of Peak Signal to Noise Ratio performance.[2]

Cyclops, Panoptes, Meerkats, FireFly Mosaic, CITRIC and Vision Mote (see Table 3) use JPEG or modified JPEG compression algorithms to perform intra-frame coding. No data compression or coding results are reported for either MeshEye [19] or MicrelEye [23] platforms. There is an evident need of low-complex and high efficient image compression mechanisms able to offer acceptable QoS in terms of power-rate-distortion parameters or

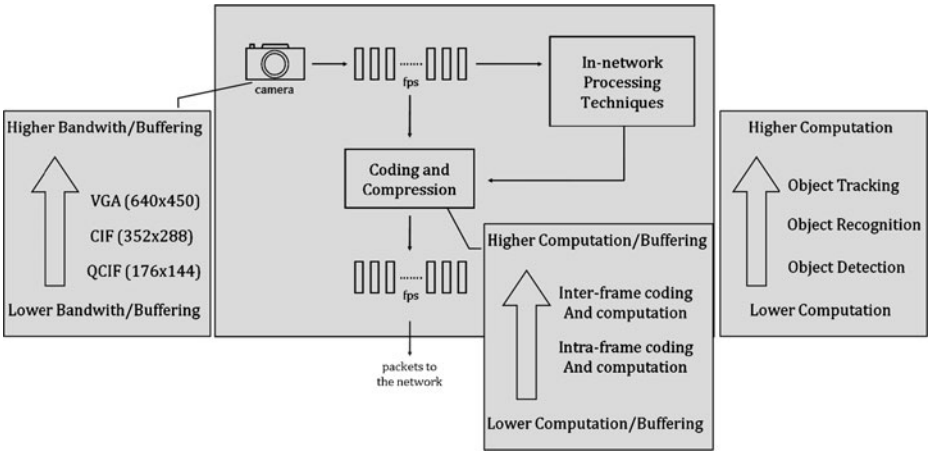


Fig. 1 Computation trade-offs in VSNs

[2] Because of their simplicity, PSNR (Peak Signal-to-Noise-Ratio) and MSE (Mean Square Error) are most widely used models for assessing image/video quality. We note that there are better models available in the literature [50].

Table 2 Comparison of visual sensor network platforms

Platform	Level of integration	MCU	Memory	Radio	Energy dissipation	Imager/ Resolution	Encoding	Operating system/ Software	Hierarchy	Model (Push/ Pull)
Cyclops [40]	Imager and radio are on two different boards	ATMEL ATmega128L (8-bit) processor on both imaging board (IM) & networking board (NB)	512 KB FLASH 64 KB SRAM on IB, 4 KB SRAM 128 KB FLASH on NB	CC1000 (38.4 Kbps)	110.1 mW (max) 0.8 mW (sleep)	Agilent ADCM-1700 (352×288)	8-bit monochrome 16-bit YCbCr 24-bit RGB	TinyOS/ nesC	Tier1	Pull
MeshEye [19]	Fully integrated	ARM7TDMI based ATMEL AT91SAM7S (32-bit)	64 KB SRAM 256 KB FLASH internal memory, 256 MB external on-board FLASH memory	CC2420 (250 Kbps)	175.9 mW (max) 1.8 mW (sleep)	Agilent ADNS-3060 (30×30), Agilent ADCM-2700 (640×480)	6-bit greyscale 24-bit color	No-OS	Tier1 and Tier2	Push and Pull
Panoptes [14]	Custom off-the-shelf hardware components	StrongARM (32-bit)	64 MB	IEEE802.11	5.3 W (max) 58.0 mW (sleep)	Logitech 3000 USB WebCam (640×480, 320×240, 160×20)	32 bit RGB	Linux/ Python	Tier3	Push and Pull
Meerkats [3]	Custom off-the-shelf hardware components	XScale PXA255 (32-bit)	32 MB FLASH 64 MB DRAM	IEEE802.11b (1–11 Mbps)	3.5 W (max) 49.2 mW (sleep)	Logitech 4000 USB WebCam (640×480)	32 bit RGB	Linux	Tier3	Push
FireFly Mosaic [42]	Imager and radio are on two different boards	LPC2106 ARM7TDMI (32-bit) processor on IB, ATMEL Atmega1281 (8-bit) processor on NB	64 KB RAM 128 KB FLASH on IB, 8 KB RAM 128 KB FLASH on NB	CC2420 (250 Kbps)	572.3 mW (max) 0.3 mW (sleep)	CMUcam3 (352×288)	8-bit RGB 8-bit YCbCr	Nano-RK	Tier2	Push
MicrelEye [23]	Fully integrated	ATMEL FPSLIC SoC, with an AT40K MCU (8-bit)	36 KB onboard SRAM 1 MB external SRAM	LMX9820A Bluetooth (230.4 Kbps)	500.0 mW (max)	Omnivision OV7640 (320×240)	16-bits[a]	No-OS/C	Tier1 and Tier2	Push
XYZ-ALOHA [48]	Imager and radio are on two different boards	ARM7TDMI-based (32-bit) OKI ML67Q5002 on NB	32 KB RAM 256 KB FLASH onboard, 2 MB external RAM on NB	CC2420 (250 Kbps)	238.6 mW (max) 2.2 mW (sleep)	ALOHA imager (NA)	1-bit	SOS	Tier1	Push
CITRIC [6]	Imager and radio are on two different boards	PXA270 (32-bit) on IB TI MSP430 (16-bit) on NB	64 MB SDRAM and 16 MB FLASH on IB, 10 KB RAM and 1 MB FLASH on NB	CC2420 (250 Kbps)	970.0 mW (max) 527.0 mW (min)	Omnivision OV9655 (1,280×1,024, 640×480, 40×30)	8-bit/10-bit	Linux / C,C++	Tier2	Push and Pull
Vision Mote [57]	Fully integrated	ATMEL 9261 ARM 9 (32-bit)	128 MB FLASH 64 MB DRAM	CC2430 (250 Kbps)	489.6 mW (max)[b] 5.2 mW (sleep)[b]	CMOS Camera (640×480, 320×240)	–	Linux	Tier2	–

[a] 4:2:2 YUV image capture and only 8-bits for luminance and 8-bits for one of the chrominance components are used

[b] The source voltage is not reported and assumed to be 3.6 V [1, 57]

Table 3 Compression/Coding techniques, In-network processing techniques and Applications of VSN platforms

Platform	Compression/ Coding techniques	In-network processing (vision computing) techniques	Targeted applications
Cyclops [40]	JPEG compression	Matrix operation libraries	Object detection
		Edge detection (Sobel algorithm)	Hand posture recognition
		Background subtraction (Object detection)	
		Coordinate conversion	
MeshEye [19]	–	Background subtraction (Object detection)	Distributed intelligent surveillance
		Stereo matching algorithm (Object location)	
		Object acquisition and extraction	
Panoptes [14]	JPEG compression	Motion detection filtering algorithm	Video surveillance
	Differential JPEG compression		
Meerkats [3]	JPEG compression	Object/event detection based on background subtraction or frame differencing	Moving body tracking
FireFly Mosaic [42]	JPEG compression	Frame differencing, Color tracking, Convolution, Edge detection	Assisted living application
		Connected component analysis, Face detection, etc.	Home activity clustering
MicrelEye [23]	–	Background subtraction	Image classification
		Feature extraction	People detection
		Classification algorithms	
XYZ-ALOHA [48]	Address event representation	Histogram reconstruction	Pattern recognition (hand recognition, hand gesture recognition)
		Motion, Edge and Centroid detection[a]	
CITRIC [6]	Compressive sensing	Background subtraction	Single target tracking
	JPEG compression	Frame differencing	Camera localization using multi-target tracking
		Object acquisition and extraction	
		Markov chain Monte Carlo data association	
Vision Mote [57]	JPEG compression	–	Water conservancy engineering

[a] Different AER image emulators are used

other quality metrics. The only prototypes that implement and report a scheme different than JPEG compression or its versions are XYZ-ALOHA and CITRIC. CITRIC implements compressed sensing; a new approach for simultaneous sensing and compression explained in this subsection.

XYZ-ALOHA uses AER (Address Event Representation). AER only outputs a few features of interest of the visual scene by detecting intensity differences (motion) information. These sensors are sensitive to light and motion so that only pixels that include the brightest one generate the events first and more periodically than others. Thus, only pixels which realize a difference in the light intensity generate the events. Since, the sensor sends metadata and not images, there is a great saving in number of bits and thus in energy.

There are alternatives other than modifying standard JPEG compression such as Distributed Video Coding (DVC) [15, 37–39] based on Slepian-Wolf and Wyner-Ziv

theorems. DVC exploits the possibility to compress two statistically dependent signals using a separate encoding and a joint decoding approach, which is different from the classical predictive encoding that jointly encodes and decodes. This approach moves the complexity from the source node to the end node, giving room to design low-complexity and low-power sensors at the cost of the sink node. As reported in [37], sink complexity may be a drawback if high node density sensor networks are deployed in terms of how many sensors such a sink could support and what DVC compression efficiency may be achieved with low encoding complexity. Nevertheless, the potential benefits of DVC in VSNs are due to the higher coding efficiency allowing (a) reduction in transmission rates; (b) low power consumption; (c) improved error resilience; (d) multi-view correlation exploitation.

None of the existing VSN platforms and prototypes introduced in Section 4 has implemented and tested DVC. However, at the academic/research level there is a growing base of information about the use of DVC in VSNs. For instance, PRISM [39] aims to bring block-motion block-DCT to the decoder. In [26], the use of Lapped Biorthogonal Transform (LBT) instead of DCT/DWT (Discrete Cosine/Wavelet Transforms) and Golomb+Multiple Quantization (MQ) methods instead of Huffman coding or arithmetic coding is proposed. To further reduce the compression complexity the authors propose sharing the processing tasks between nodes belonging to the same cluster. The result is a low-complexity and low-memory demanding image compression algorithm that reduces hardware cost and energy consumption. In [22], a DVC scheme is proposed for VSNs which exploits the correlations among video frames from adjacent nodes via robust media hashing. The global motion parameters are estimated by the decoder (the sink) and sent to the encoders.

Compressed Sensing or Compressive Sampling (CS) [4, 11, 26] aims to compress data before storing the whole information to be compressed. CS works on the sparseness representation of the signal—instead of acquiring the whole signal [4] the complete set of coefficients are computed, the largest ones are encoded and the rest of the coefficients are discarded. CS assumes data is in $y=Ax$ form, where A is a kxn sensing sparseness matrix ($k \ll n$). CS guarantees that for a certain matrix A, x may be reconstructed from y whenever x is compressible in some domain. Vector y is easy to compute (even in a distributed manner), has small size and does not require a prior knowledge about the data, being therefore a good candidate for compressing in VSN applications. Distributed Compressed Sensing (DCS) [12] exploits both intra and inter-signal correlation structures and is based on the concept of joint sparseness of a signal ensemble. Compressive Wireless Sensing (CWS) [2] reconstructs the sensed data from noisy random projections of the sensor network data and obtains good power-distortion trade-offs given the lack of prior knowledge about the signal. Network Coding [8] can help to minimize the amount of energy required per packet. Furthermore, the use of joint network coding [54] and distributed source coding to send multiple correlated sources to a common receiver is also proposed.

CS and other related techniques are promising emerging techniques for coding and compression of data. At this moment, current platforms and prototypes focus on well-known intra-frame compression algorithms and rely on in-network processing for data reduction. An exception is CITRIC platform [6] where image compression via CS is implemented and tested. In CITRIC's CS implementation different sensors measure sparse but correlated signals and send them to a sink that is able to jointly reconstruct all the signals precisely.

Data processing in VSN is more computationally intensive than scalar sensors. That makes energy dissipation for video coding/compression an important feature to be analyzed and researched. In scalar sensors, there is a clear objective in analyzing connectivity energy consumption (transmission and reception) of individual sensors and global network lifetime in terms of protocol energy performance. Nevertheless, visual sensors consume higher data

processing energy than scalar sensors and therefore, there is a need to optimize energy consumption for data processing in individual visual sensors. In this sense, an analytical model [17] is introduced to characterize the relationship between encoding power consumption and its quality performance. The objective is to extend the traditional Rate-Distortion analysis to include energy constraints. Rate-Distortion and Time-Distortion trade-offs also are evaluated in CITRIC [6], where it is shown that CS Time-Distortion tends to become zero while JPEG needs a residual computation time of 70 ms per image for the lowest distortion performance. Incorporating power consumption and rate-distortion analysis to video sensors is a promising research avenue since most of the current studies related to video coders in VSNs are using intra- and inter-frame compression techniques. The next subsection is dedicated to in-network vision computing mechanisms.

3.2 Vision computing

Techniques aimed to manipulate image content and, thus, allowing the transmission of a reduced amount of information are generally named as in-network processing techniques within VSN domain. Most of these techniques come from the vision computing field. Vision computing can reduce the amount of data to be sent to the sink. Well-known visual computing techniques, such as object recognition or object tracking [21], can reduce the amount of raw data to be transmitted. Although there are many techniques for object recognition depending on the applications, general object recognition methods can be categorized as in Fig. 2. Object/event detection and recognition can be done by using the identified features of the object. Object tracking can be done by localization algorithms (e.g., active contour-based, feature-based, and model-based).

Vision-computing techniques implemented in VSN platforms are summarized in Table 3. It is important to mention that most of these techniques require powerful processors and large memory resources if they are not customized for implementation in VSN platforms. Vision-computing techniques optimized for VSN platforms are implemented, tested and

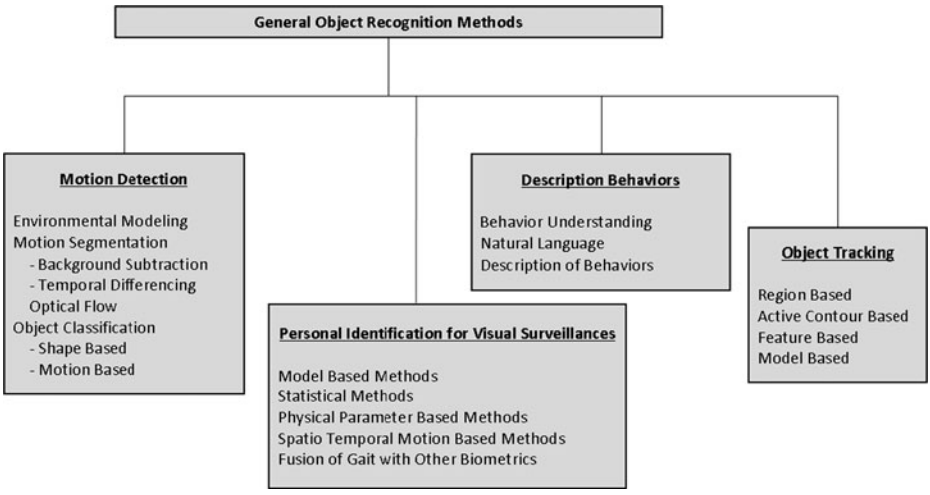


Fig. 2 General object recognition methods

reported for most of the VSN platforms. Here, similar to what we have done for compression and coding general overview of vision computing in VSN platforms is provided.

Most of the platforms utilize background subtraction, frame differencing, and edge detection as the main in-network processing techniques. Background subtraction is based on extracting the object of interest from the image and removing the background. It must be robust against changes in illumination and should avoid detecting non-stationary background objects (moving leaves, rain, snow, etc). Moving average filters smooth backgrounds with illumination changes or moving objects. Frame differencing detects changes between two consecutive frames. Differencing the current frame with a background frame averaged over past frames adds robustness against illumination changes. For example, Meerkats, Cyclops, MeshEye, MicrelEye, and CITRIC perform background subtraction and frame differencing as object detection mechanisms. MicrelEye takes a fixed background frame at the beginning and then performs pixel-by-pixel differencing between each frame and the background frame. Cyclops use moving average filters to smooth background changes. Firefly Mosaic uses a Gaussian Mixture Model (GMM) to separate foreground from background. GMM is able to detect objects even when the object stands still.

Edges could be considered as boundaries between dissimilar regions in an image. Computation of edges are fairly cheap and recognition of an object is easy since it provides strong visual clues, however, edge detection can be affected by the noise in an image. Although there are several different methods to perform edge detection, generally they can be categorized into two classes: (a) gradient-based (e.g., Sobel algorithm); (b) Laplacian-based. While the first one detects the edges by looking for the maximum and minimum in the first derivative of the image, the second one searches for zero crossings in the second derivative of the image to find edges. Cyclops supports Sobel libraries to perform edge detection. Delay incurred by edge detection as a function of image size is investigated for XYZ platform [48]. For example, 8-bit Sobel edge detection lasts respectively 3,560 ms, 248 ms, 65 ms and 15 ms for 320×240, 256×64, 64×64 and 32×32 resolutions. However, the use of an FPGA block can decrease delay at reasonable energy costs.

The results reported on energy dissipation and efficiency of the vision algorithms suggests that vision-computing is an essential component of the VSN paradigm. It is shown in [3] that processing energy dissipation is no longer negligible and is comparable to (even higher than) the energy dissipation for networking. Meerkats performs object detection based on background subtraction that simply detects motion taken between two snapshots at different times. Nodes wake-up periodically, perform object detection, and send a wake-up message to neighbors which, in response, start taking pictures. This duty-cycle method consumes more batteries than using the n-tier architecture, proposed in [25]. Power consumption and performance of object classification tasks are reported for MicrelEye [23]. The authors implement three hardware and software architectures (serial, parallelized and optimized) and demonstrated a power consumption of 430 mW at 5 fps for the serial implementation, 500 mW at 9 fps for the parallel implementation and 500 mW at 15 fps for the optimized implementation, showing that simple in-network processing tasks can be performed at moderate frame rates. Energy consumption will be one of the main challenges in the design of future in-network processing techniques.

Exploiting spatial correlation between the images obtained by neighboring nodes can reduce power consumption. Collaborative image coding [53] computes differences between neighbor camera frames and local frames. Then, using edge detection (e.g.; Sobel operator) dominant features that are to be sent to the sink are computed. Before initiating the process it is necessary to obtain the background to later reconstruct the image, where synchronization of the cameras is the critical part. Experiments in [53] are

performed using an Intel StrongARM SA 1100 as processor and LMX3162 as transceiver. An application in which, after an initial training phase, multiple overlapping cameras merge regions and collaborate in object activity detection is proposed for Firefly Mosaic platform [42]. However, the trade-off between the increase in signal processing for target detection and the transmission of data in light-weight sensors still remains as an active research issue. Collaborative image coding is a promising technique for VSNs; however, it has also drawbacks. In networks with scalar sensor nodes, the sensing area and the transmitting node's coverage area are basically the same. In visual sensor networks, sensing nodes are cameras with a FoVand a DoV (Depth of View). FoV is the maximum volume visible from a camera while DoV is the amount of distance between the nearest and farthest objects that appear in an image [32]. Overlapping cameras can cover the same sensing area depending on the FoVand the DoV; however, if the transmission radius is not large enough then VSN nodes with correlated data cannot directly reach each other. In this situation, collaborative image coding needs specific sensor coordination algorithms to reach out of range nodes.

4 VSN platforms

In this section we present an overview of nine available VSN platforms. There are three criteria we use to select the VSN platforms described below. First, all platforms have local processing capabilities. They do not transmit visual data in raw form. Second, they all operate with battery not with power adapters. Third, all nine platforms selected are sufficiently characterized. For instance, information on energy dissipation characteristics and in-network processing techniques is reported.

Due to the several options with respect to integration level as discussed in Subsection 5.1, there is no general architecture which all platforms comply to. However, we observe that all platforms share a similar set of hardware components. A typical VSN platform consists of at least four building blocks; a microcontroller, a memory (on-board and/or external), imager(s) and wireless transceiver (built-in or integrated) and is powered with a limited battery resource. In this section, we aim to present all VSN platforms which have the aforementioned physical attributes and are well-characterized (e.g., energy dissipation characteristics are provided). Nevertheless, in addition to the nine platforms we presented in detail, we also presented several other platforms in Section 4.10.

VSN platforms presented in this section are similar not only in their physical attributes but also in their capabilities and operation. Typically, VSN platforms capture visual data from the environment, process the data locally and then transmit processed data via its wireless transceiver to a sink either directly or over relays with multihop communication. The visual data can be either still image or video. The ability to process the visual data is crucial for VSNs because sending raw data is usually not a viable option due to the power and bandwidth constraints. VSN platforms are equipped with radios that can both transmit to and receive from the base station hence processed visual data can be transmitted to the sink either with a push model (e.g., event detection triggers the visual data transfer) or a pull model (e.g., information is requested by the sink with a broadcast query message). However, depending on the application one particular model can dominate the other. For example, in a surveillance application push model is the dominant mode of operation—detection of a moving object triggers VSN nodes to transmit data to the sink node. In Table 2, we present Push/Pull model of operation for each platform based on the test cases reported.

In the following subsections, we organize the information provided for each VSN platform under four headings and strive for providing content consistently as much as possible. However, the goal of consistency can not always be achieved due to unavailable information and missing details in relevant publications. Main structural attributes of the platforms are summarized in Table 2. Compression/coding and in-network processing techniques used in VSN platforms together with information on targeted applications are presented in Table 3.

4.1 Cyclops

The information for Cyclops [40] is organized under the following headings:

4.1.1 Hardware

Architecture of Cyclops [40] is given in Fig. 3. Cyclops is designed to operate with Mica2 mote, a WSN platform from Crossbow technology [10]. The imager is a CIF (352×288) resolution camera. The MCU is an 8-bit processor, which is responsible for controlling Cyclops, communicating with Mica2 and performing image inference. In addition to limited amount of internal MCU memory, there is an external SRAM (64 KB) as well as an external FLASH (512 MB). Image capture is performed by the CPLD (Complex Programmable Logic Device).

4.1.2 Software/Protocols

Cyclops firmware is written in nesC language. The operating system is TinyOS, which provides clear interfaces and standard scheduler and services. Cyclops software consists of three set of components: drivers, libraries and the sensor application. Drivers are components used for interfacing the MCU with the peripheral subsystems (e.g., CPLD drivers). There are two different classes of libraries: primitive structural analysis libraries (e.g., matrix manipulation libraries, statistics and histogram libraries) and high-level algorithmic libraries (e.g., background subtraction, object detection, motion vector recognition libraries). The sensor

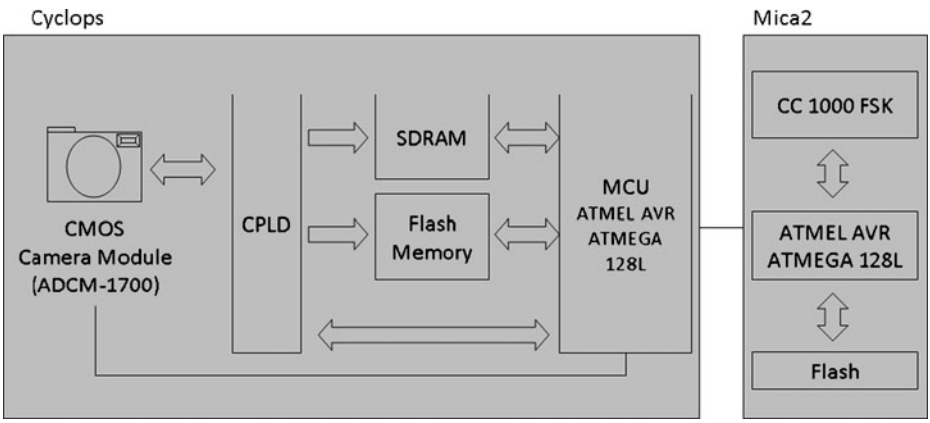


Fig. 3 Hardware architecture of Cyclops

application orchestrates the overall operation of Cyclops and provides a high-level interface with the host. Cyclops sees the external world through the Active Eye Layer module, which is the Cyclops image capture and control stack. Mica2 works on a TinyOS networking stack released in TinyOS 1.0 [28].

4.1.3 Applications and processing techniques

Cyclops is used in object detection and hand posture recognition applications. Detection of a moving object within the Field-of-View (FoV) is a critical task in many sensing applications. For this purpose Cyclops periodically captures images and reconstructs an estimate of the stationary background by using a moving average methodology. The foreground object is detected by taking the difference between the instantaneous image and the estimated background. Experimental evaluations reported that Cyclops achieves 78.4% detection efficiency. Hand posture detection is important for human–computer-interaction and it is a well-known pattern recognition problem. Cyclops converts a set of training images into feature vectors by using an orientation histogram transformation, which gives robustness to illumination variations and provides translational invariance. The performance of the algorithm is tested by using static hand gestures form a subset of American Sign Language, which resulted in 92% successful recognition.

4.1.4 Notable features and/or functionalities

CPLD is used as a lightweight frame grabber because image capture requires faster data transfer and address generation than what a lightweight processor can provide. Cyclops is designed for maximum energy efficiency and for this purpose any hardware block that is not needed to be active during a particular operation is automatically relaxed to its lower power state. For example, CPLD clock is halted by the MCU when its services are not required; likewise, SRAM is kept in sleep state when the memory resources are not needed.

4.2 MeshEye

The information for MeshEye [19] is organized under the following headings:

4.2.1 Hardware

Figure 4 illustrates the hardware architecture of MeshEye [19], which is a single board fully integrated vision sensor platform designed with two low-resolution imagers and one high-resolution camera. The main processor of MeshEye has 64 KB SRAM and 256 KB FLASH memory. For image buffering and storage there is an MMC/SD FLASH memory card of 256 MB size. The wireless communication module is a CC2420 radio.

4.2.2 Software/Protocols

CC2420 radio implements the IEEE 802.15.4 standard. MeshEye does not have an operating system. Resource management and process scheduling are performed through FSM-based approaches details of which are not provided. On the other hand, two data link protocols are implemented and reported. The first protocol is a simple data/ACK protocol, where each new data packet is sent upon reception of an ACK from the receiver for the previous packet and in the lack of an ACK retransmission occurs upon the expiration of a

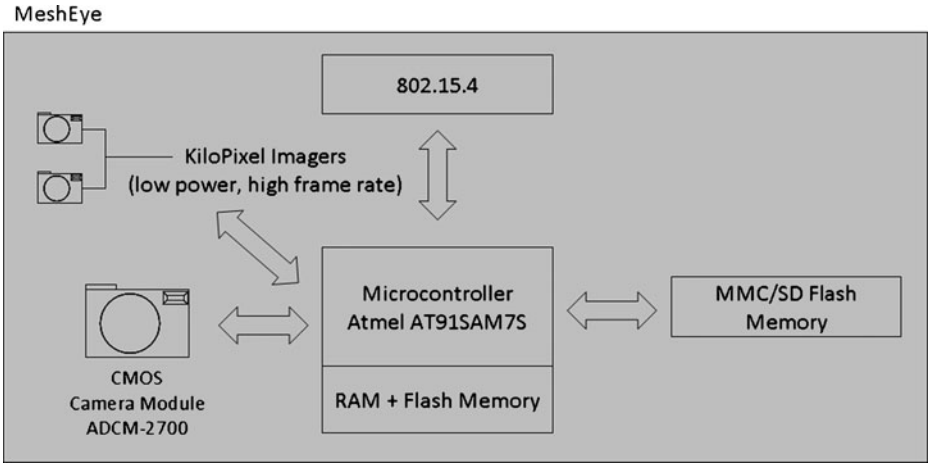


Fig. 4 Hardware architecture of MeshEye

certain timeout. In the second protocol only the first and last packets of a data stream are acknowledged by the receiver. The list of all lost data packets is sent together with the last ACK, which is then retransmitted to the receiver at the end of the transmission.

4.2.3 Applications and processing techniques

The design of MeshEye has been pursued with distributed intelligent surveillance application in mind. The vision algorithms of MeshEye consist of object detection, stereo matching, and object acquisition phases. Object detection is achieved by computing the difference of each captured frame from the estimated background. The difference map is then processed by thresholding, smoothing, and blob filtering. Once the Region of Interest (ROI) is detected for one imager, stereo matching phase begins, where the difference images of two low-resolution images are matched by correlation processing to confirm that the ROI detected in previous phase is a positive match. Once an object is detected and stereo matched, the high-resolution camera takes a snapshot of the ROI for further processing.

4.2.4 Notable features and/or functionalities

Low resolution imagers in MeshEye architecture are used for detection of a moving object within their FoVs. Once an object is detected the high resolution camera is triggered and starts to acquire the high resolution images from the FoV. Such a system level design considerably improves the energy efficiency of the overall architecture without sacrificing the quality of the object detection (i.e., high resolution imager does not dissipate energy when there is no moving object, yet, high resolution images can be acquired rapidly upon detection of a moving object).

In design of MeshEye off-the-shelf low-power components are used with standard interfaces and total number of components are minimized to keep the energy dissipation as low as possible (e.g., flash memory card, KiloPixel imagers, and the radio are connected to the processor through a single SPI interface). MeshEye is specifically designed for low-power surveillance applications. Due to the lack of FPGA or CPLD based frame capturing and buffering components in MeshEye design, real-time streaming video is not possible.

4.3 Panoptes

The information for Panoptes [14] is organized under the following headings:

4.3.1 Hardware

Hardware architecture of Panoptes [14] is illustrated in Fig. 5. The Panoptes platform is originally developed on an Applied Data Bitsy board, which utilizes a processor with 64 MB of on-board memory. A video camera is utilized for acquisition with resolutions ranging from VGA (640×480) to lower resolutions (160×120). The communication module is an off-the-shelf IEEE 802.11 card. The system is later migrated to the Crossbow Stargate platform [10]. Panoptes use USB 1.0 as the primary I/O interconnect with a maximum of 12 Mbps aggregate bandwidth due to the fact that at the time of implementation low-powered embedded devices do not support higher bandwidth (455 Mbps) USB 2.0.

4.3.2 Software/Protocols

IEEE 802.11 protocols are used for communication. Linux operating system is used due to its flexibility. Python is used for providing adaptive sensor functionality, which is an interpreted programming language. The advantage of using Python is that it allows both the advantage of the speed of optimized compiled code and the flexibility of a scripting language in construct. In Panoptes software architecture video sensing functionality is divided into several components including capture, compression, filtering, buffering, adaptation, and streaming. Phillips Web Camera interface with video for Linux is used for video capture.

A priority buffering and adaptation scheme is developed for Panoptes, where data packets are handled according to the priority level and timestamp they are labeled with.

The application software system consists of three main components: the user interface, the video sensor software, and the video aggregation software. The user interface allows basic queries to be run on the video database along with other functionalities. The video sensor software detects the motion within the FoV and records the motion within the FoV

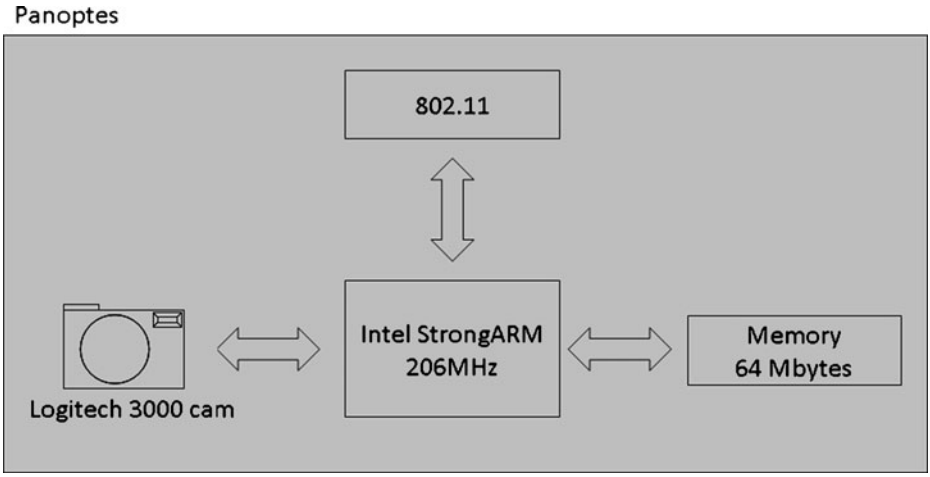


Fig. 5 Hardware architecture of Panoptes

for a user-defined amount of time. Furthermore, the recorded motion is transformed into a bitmap of active motion detected pixel blocks to allow efficient database access for future queries. Video aggregation software orchestrates the visual surveillance system as a whole. It defines and executes visual queries by using the pre-processed information provided by the available software components. For example, by using aggregated image bitmaps for a single event (*event_overview_map*), the system rapidly identifies events of interest. Various aspects of the Panoptes platform are analyzed through direct experimentation (e.g., USB performance, compression performance, component interaction, energy dissipation, and buffering and adaption performance).

4.3.3 Applications and processing techniques

Panoptes platform is used in an intelligent visual surveillance application called "The little sister sensor network application". JPEG, differential JPEG, and conditional replenishment compression formats are set up on the Panoptes platform. The performance of off-the-shelf JPEG compression algorithm (ChenDCT) is compared with an implementation which takes advantage of Intel's Performance Primitives for the StrongARM and XScale processors.

Filtering is used to eliminate the redundant or uninteresting image frames or video sequences. The implementation is based on a brute-force pixel-by-pixel algorithm. The filtering element in Panoptes allows a user to specify how and what data should be filtered.

4.3.4 Notable features and/or functionalities

Panoptes is one of the earliest video sensor network platforms designed for high quality video delivery over IEEE 802.11 networks with a power requirement less than 5 W. Since Panoptes is designed with more powerful hardware, it is able to support more complex compression algorithms.

Panoptes platform achieves approximately 10 fps using a high-quality image (640×480 image resolution). Furthermore, using gray-scale images reduces the image size by 1/3.

4.4 Meerkats

The information for Meerkats [3, 27] is organized under the following headings:

4.4.1 Hardware

Figure 6 shows the hardware architecture of Meerkats [3]. The Meerkats testbed consists of Meerkats nodes designated as ordinary VSN nodes and a laptop acting as a data sink. Meerkats nodes are built on top of a Crossbow Stargate platform [10], which features a CPU operating at 400 MHz with 32 MB FLASH memory and 64 MB RAM on the main board. The camera can capture VGA (640×480) quality image frames. The communication module is an Orinoco Gold IEEE 802.11b PCMCIA wireless card.

4.4.2 Software/Protocols

IEEE 802.11 protocols are used for communication. The operating system used on the Stargate platform is an embedded Linux system. Furthermore, Stargate comes with built in battery monitoring capability, which is extremely useful for testing the energy dissipation characteristics.

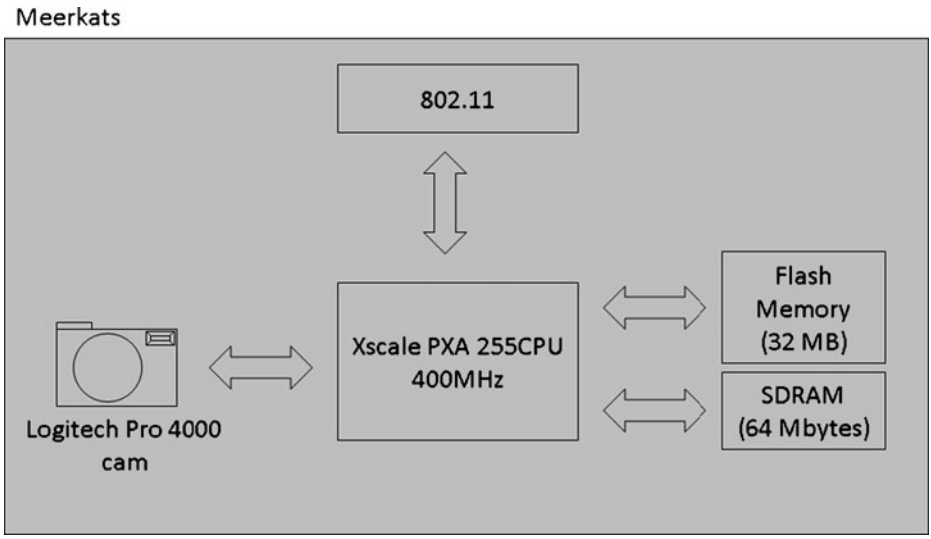


Fig. 6 Hardware architecture of Meerkats

Meerkats software architecture has three main components: (a) resource manager module; (b) visual processing module; and (c) communication module. Resource manager module oversees the overall operation of the Meerkats node. It controls the visual sensor and the communication subsystems. Energy efficiency of the platform is among the responsibilities of the resource manager, which orchestrates the operation of the platform on a duty-cycle basis (e.g., the node periodically wakes up, acquires image, and returns back to sleep or idle mode depending on the energy dissipation and QoS requirements). Visual processing module is the nerve center of the sensing operations of Meerkats. The operation of visual processing module is triggered by the resource manager, which simultaneously activates the webcam. Visual processing module processes the acquired sequence of images for possible moving objects. Visual processing unit returns resource manager module a set of parameters including an event detection flag, the number of moving blobs, and the velocities of the blobs. Eventually, critical data is transmitted to the sink. Communication module is responsible for transmitting data and control traffic towards the data sink possibly through multiple hops.

Network layer functionality is performed by the Dynamic Source Routing (DSR) protocol. DSR version running on Meerkats nodes is ported from the PocketPC Linux DSR kernel module. Meerkats use both UDP (control packets used for synchronization and signaling) and TCP (data packets used for image data transmission) for transport layer functionality.

4.4.3 Applications and processing techniques

Meerkats testbed is designed for detecting and tracking moving bodies within the FoV. The performance metric of the surveillance application is defined as the ratio of the expected number of missed events to the expected number of bodies in the network.

Main function of Meerkats' visual processing module is to detect objects of interest within the FoV. Image acquisition, image understanding, image processing, and data compression operations are among the duties of the vision processing module. If an event

(i.e., moving blobs) is detected through a fast motion analysis algorithm, the relevant portion of the visual data is JPEG compressed.

4.4.4 Notable features and/or functionalities

Meerkats platform is designed by integrating carefully selected and readily available off-the-shelf components. For example, Stargate platform is used because it features a USB connector, which is necessary for interfacing the webcam used as the visual sensor. Two Meerkats nodes may coordinate a master-slave coordination scheme for energy efficiency purposes, where the master periodically scans the FoVand alerts the slave via the radio link if it detects a moving object.

4.5 FireFly Mosaic

The information for FireFly Mosaic [42] is organized under the following headings:

4.5.1 Hardware

Figure 7 shows the hardware architecture of FireFly Mosaic [42]. FireFly Mosaic platform is created by integrating FireFly WSN platform with a vision board featuring CMUcam3 embedded vision processor (there is also an additional board for PC interface). The vision board consists of a CMOS camera chip, a frame buffer, and an MCU. The imager is capable of supporting 50 fps in CIF resolution. The frame buffer is an Averlogic AL440b FIFO chip. Processing of images is performed by a 32-bit LPC2106 ARM7TDMI running at 60 MHz with 64 KB on-chip RAM and 128 KB on-chip FLASH memory. The FireFly WSN nodes are equipped with a CC2420 radio and an ATMEL Atmega1281 8-bit processor.

4.5.2 Software/Protocols

CMUcam3 provides various open source image processing and vision libraries. The FireFly platform utilizes Nano-RK real-time multi-tasking priority driven reservation-based power-aware operating system, which provides the necessary means to maintain network wide tight global synchronization. Such level of time synchronization is of utmost importance for synchronous image capturing and communication scheduling. The communication between the imaging board and networking board is based on the Serial Line IP protocol. Inter-node

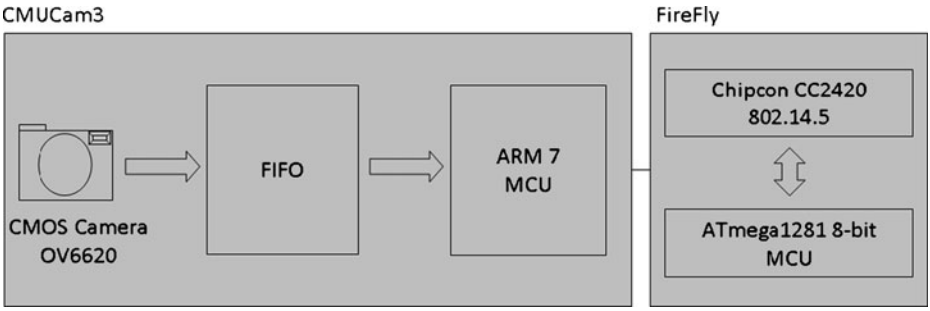


Fig. 7 Hardware architecture of FireFly Mosaic

communication is achieved by using TDMA-based link layer architecture (i.e., RT-link collision-free TDMA link protocol) with scheduled and routing tables constructed according to the communication networking and FoV based network topology. It is reported that Nano-RK in conjunction with RT-link protocol can achieve sub-millisecond time synchronization. The network formed by FireFly Mosaic nodes utilize a Camera Network Graph (CNG), which captures the relationships between the cameras and their FoVs (e.g., CNG captures the overlapping FoVs' of cameras), to optimize the routing within the network. Upon construction of the CNG the TDMA schedule is computed by using a two-step procedure.

4.5.3 Applications and processing techniques

The target application scenarios for FireFly Mosaic include the broad field of distributed vision-tasks (e.g., assisted living applications for elderly). FireFly Mosaic platform is used in a home activity clustering application. In this application the system automatically identifies the regions within an apartment, where particular activities frequently occur. The end result of the application is a Markov model, which characterizes the transition probabilities of activity regions. In this application the TDMA timeline consists of 192 ms time epochs, which is the time to capture and process images. The frame rate achieved is 5.2 fps. There are five main steps in activity clustering: (a) the first phase is the training phase, where a test object is used to determine the correlations within the camera network and the CNG is built; (b) using a GMM (Gaussian Mixture Model) the foreground and background are separated and locations of the occupants are recorded over time; (c) the GMM is processed and significant local activities for each camera is identified; (d) local activity clusters for the same region from all relevant cameras are merged by using the CNG; (e) main activity states and their transition probabilities are determined.

The open source library of CMUcam3 includes JPEG compression, frame differencing, color tracking, convolutions, histogramming, edge detection, connected component analysis, and face detector and other processing techniques.

4.5.4 Notable features and/or functionalities

Firefly Mosaic is the first VSN system to integrate multiple coordinating cameras performing local processing. FireFly nodes include an AM receiver for external tight time synchronization which facilitates capturing synchronous images as well as scheduling appropriate communications patterns.

4.6 MicrelEye

The information for MicrelEye [23] is organized under the following headings:

4.6.1 Hardware

Hardware architecture of MicrelEye [23] is presented in Fig. 8. The processor is an ATMEL FPSLIC SoC, which includes an AT40K MCU, a Field Programmable Gate Array (FPGA) with 40 K gates, and 36 KB of onboard SRAM (16 KB can be used for data and 20 KB is reserved for program storage). The external memory for frame storage is a 1 MB SRAM. Wireless capabilities of the node are provided by the integration of LMX9820A Bluetooth transceiver, which includes a radio, a baseband controller, and a memory block.

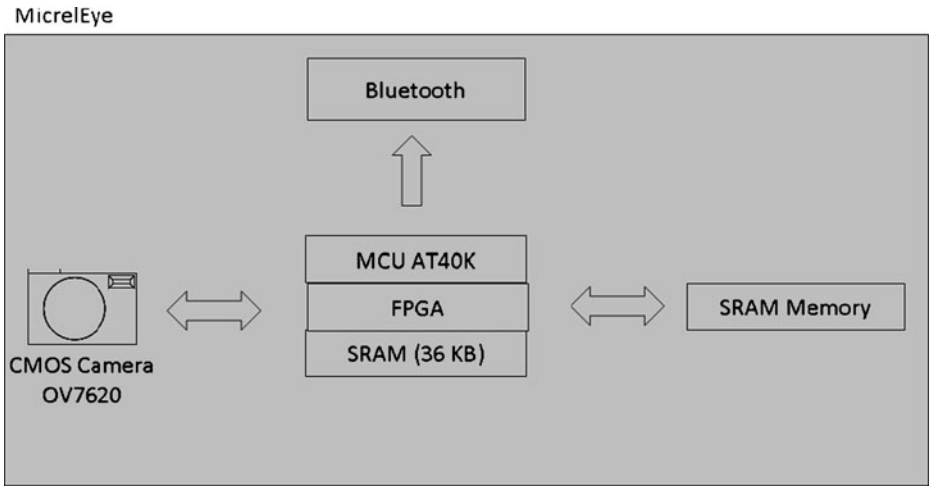


Fig. 8 Hardware architecture of MicrelEye

4.6.2 Software/Protocols

MicrelEye has no operating system. Bluetooth serial port profile is used in MicrelEye which allows the establishment of a virtual serial port between the transceiver and a remote device. The algorithm implemented in MicrelEye is split between the FPGA and the MCU to achieve parallelism, where image processing tasks involving high speed logic and high computational power (e.g., background subtraction, sub-window transfer) are performed at the FPGA and the higher level operations (e.g., feature extraction, support vector machine operations) are performed at the MCU. The part of the algorithm run on the MCU is written in C.

4.6.3 Applications and processing techniques

The MicrelEye node is used for people detection, where a smart camera positioned at a critical position (e.g., main entrance of a building) discriminates between the objects within its FoV (i.e., whether the object is a human being or not). The main motivation is developing an intelligent system capable of understanding certain aspects of the incoming data by performing image classification algorithms. The classification accuracy is validated by using well-known standard data sets and the classification results obtained are close to the results obtained by the standard SVM classifiers.

The image stream coming from the CMOS imager includes both luminance and chrominance components. After a complete frame is transferred to the FPGA, background subtraction is performed by pixel differencing with the reference frame (which can be updated as needed). The ROI within the frame (128×64) is extracted and transferred to the internal memory and the remaining higher level operations are performed by the MCU. The feature vector is extracted from the ROI, which is normalized to [0,1] interval by using a highly efficient algorithm. The feature vector consists of 192 elements (the averages of the rows and the columns). The feature vector is fed into a State Vector Machine-like (SVM-like) structure (called ERSVM), which is used to recover unknown dependencies. SVM is a "learning from examples" technique and requires a set of training data to be able classify the incoming feature vectors, which is provided before the classification operation starts. The end result is a binary classification of whether the feature vector describes a human being or not.

4.6.4 Notable features and/or functionalities

In MicrelEye, having both an MCU and an FPGA block on the same chip eliminates the energy dissipation on the capacitive loading introduced by the inter-chip PCB connections. The main function of the FPGA is to accelerate computationally demanding vision tasks which cannot be handled by the MCU efficiently (e.g., image capture high speed logic, SRAM memory access management, most of the image processing tasks for detection, interface between the FPSLIC and the transceiver, the finite state machine governing the overall system operation).

MicrelEye is evaluated by using three implementation strategies: (a) serial implementation where the MCU and FPGA operations stop and wait for other operation to finish; (b) parallel implementation where FPGA and MCU operate concurrently without waiting for each other; (c) optimized implementation where several intelligent shortcuts are introduced to improve the efficiency.

4.7 XYZ-ALOHA

The information for XYZ-ALOHA [23] is organized under the following headings:

4.7.1 Hardware

Hardware architecture of XYZ-ALOHA platform [48] is given in Fig. 9. XYZ nodes use a processor that features an ARM7TDMI 32-bit microcontroller. The processor has 32 KB of internal RAM and 256 KB of FLASH. There is an additional 2 MB external RAM available on the node. The radio transceiver of the platform is a CC2420 radio. Three different platforms are built on top of XYZ sensor nodes with (a) ALOHA image sensor; (b) OV 7649 camera module from OmniVision; (c) software image emulator. The first of these platforms is named XYZ-ALOHA. OV7649 camera module can capture images at VGA (640×480) and QVGA (320×240) resolutions.

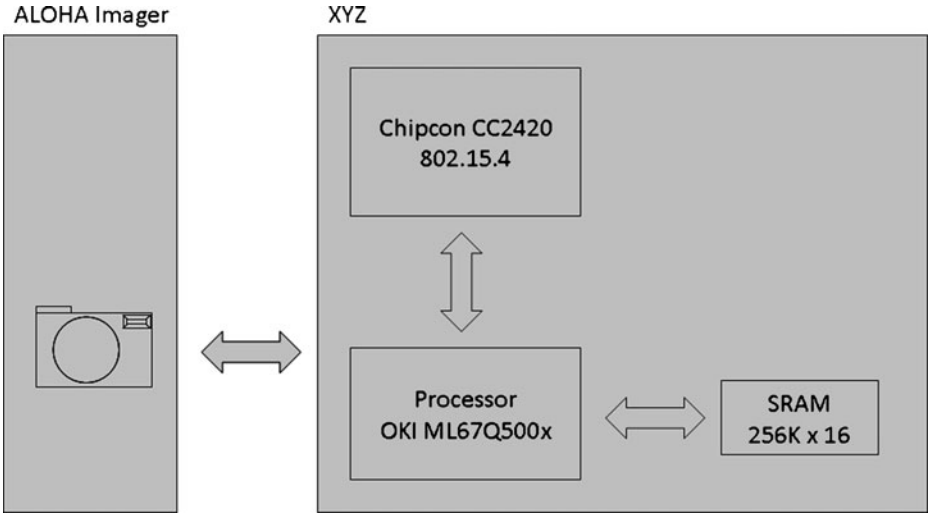


Fig. 9 Hardware architecture of XYZ-ALOHA

4.7.2 Software/Protocols

XYZ platform operates on SOS which is a lightweight operating system that follows an event driven design similar to TinyOS. Unlike TinyOS, SOS supports the use of dynamically loadable modules. XYZ implemented a specialized API in SOS that consists of device driver, sleep mode driver, radio manager and frequency manager modules.

IEEE 802.15.4 compliant medium access control is ported to XYZ which allows XYZ to interoperate with other devices running IEEE 802.15.4.

4.7.3 Applications and processing techniques

The reason why XYZ platform is utilized in three different configurations is to evaluate different aspects of the AER concept. In the first platform (XYZ-ALOHA) where the imager is an AER array, each element (pixel) of an ALOHA array signals its X and Y coordinates to the host upon the charge on it exceeds a threshold (depending on the amount of charge of the pixel induced by the incident photons), which is called an event. The second platform is built by integrating XYZ nodes with an off-the-shelf camera and the third platform utilizes a software emulator of AER, which takes an 8-bit grayscale input from a USB camera and outputs a queue of events to an output file. The reason for creating the third platform is to create a flexible infrastructure for rapid evaluation of new AER imager designs.

XYZ-ALOHA performance is tested for several application scenarios including pattern recognition (e.g., letter recognition, hand gesture recognition). Each element of the ALOHA array behaves like a sigma-delta modulator and the frequency of event generation is dependent on the light intensity at each pixel. Hence, image reconstruction from the sensor outputs require significant amount of post-processing. Histogram reconstruction is the technique chosen for reconstruction of images from ALOHA outputs due to its relative simplicity. The results of the experiments have shown that AER shortens the computation time and simplifies the task due to the fact that it does not require multiplications.

Using the software AER emulator, three different AER imagers are emulated: a motion detector, an edge detector and a centroid detector.

4.7.4 Notable features and/or functionalities

The XYZ-ALOHA platform is designed for extracting a small set of features representing certain attributes of a scene rather than acquiring full size images with a high degree of redundancy. This is a unique approach possible only with custom-designed address-based image sensors (other VSN platforms utilize a standard CMOS camera capturing a full image). The sensing modality associated with the XYZ-ALOHA system is called Address Event Representation (AER), which measure parameters of interest from the environment at the sensor level (e.g., location, direction of motion, lighting).

AER concept has many interesting properties such as ultra-low (a few μW) power dissipation (due to the elimination of redundant data acquisition), information selectivity (only the parameters of interest are sensed), and privacy preservation (full images are difficult to reconstruct from AER data).

AER is a biologically inspired communication and coding model for information flow from a sensor to a receiver, indeed, an address event channel is a model of the transmission of neural information in biological sensory systems. An AER image sensor accumulates charge in proportion with the light incident on it and when the accumulated

charge exceeds a threshold the address of the element (AER elements are connected to a bus with a matrix topology) is transmitted, hence, they do not need to be queried to pull information, instead, information is pushed to the receiver by the sensors themselves. In an AER array of sensor elements sensitive to light intensity, brighter sensors produce events more frequently than the others (the same concept applies also for other sensory data types).

4.8 CITRIC

The information for CITRIC [23] is organized under the following headings:

4.8.1 Hardware

Figure 10 shows the hardware architecture of CITRIC platform [6] which is created by integrating an imaging board designed by CITRIC team with a well-known networking platform i.e., Tmote Sky [30]. Tmote Sky is equipped with a 16-bit MCU (10 KB RAM and 48 KB FLASH), a IEEE 802.15.4-compliant radio, and 1 MB external FLASH memory. Imaging platform includes a 1.3 megapixel CMOS imager, a frequency-scalable PDA class CPU, 16 MB FLASH, and 64 MB RAM. The imager provides images with SXGA resolution and can be scaled down to 40×30 resolution. The processor has 256 KB internal SRAM. The microphone on the board is connected to the Wolfson WM8950 mono audio ADC.

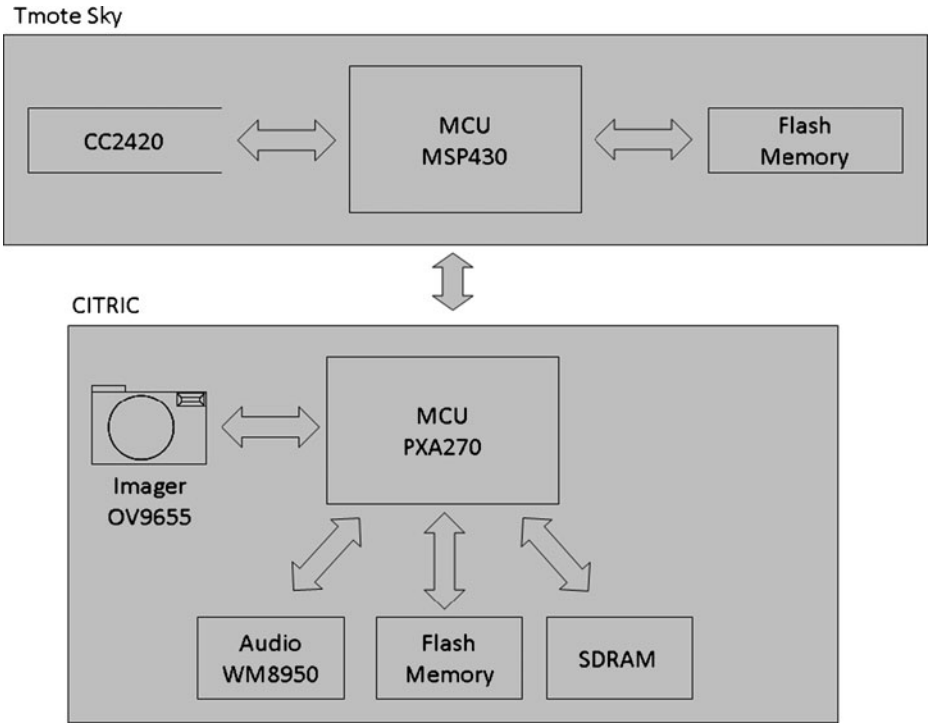


Fig. 10 Hardware architecture of CITRIC

4.8.2 Software/Protocols

CITRIC team uses Tmote Sky which is an off-the-shelf mote running TinyOS/NesC. Data transmission is achieved using IEEE 802.15.4 protocol. Embedded Linux is chosen as OS of the imaging board for rapid prototyping and ease of programming and maintenance. For JPEG compression, IJG library provided by embedded Linux OS is utilized.

4.8.3 Applications and processing techniques

CITRIC is designed to support the computational demands of a wide range of distributed pattern recognition applications. Visual surveillance is the basic scenario considered in the design of CITRIC as a system and the overall system design is inspired by the needs of security of a perimeter administered by a central entity.

The performance of CITRIC as a VSN platform is demonstrated via three representative applications: image compression, target tracking, and camera localization. In image compression application the speed of image compression is investigated, furthermore, rate-distortion and time-distortion tradeoffs between JPEG and CS schemes are measured. In the single target tracking application the foreground object is separated from the estimated background by using frame differencing, thresholding, and median filtering. Bounding boxes are computed for each foreground object identified and sent to the server (possibly from multiple VSN nodes). In camera localization application multiple cameras track multiple mobile targets and construct the paths traversed by the targets. These tracks are used to determine the location of the cameras by using an adaptive localization method.

4.8.4 Notable features and/or functionalities

The shared computing model is introduced with CITRIC. The client/server interface enables users to access the network resources concurrently. The first user of a node becomes the manager of the node and can assign different tasks to it. Users are able to listen to managed nodes but not assign tasks until the manager user logs out of the node. Usability of the system and easiness to develop new applications is one of the design goals of CITRIC.

4.9 Vision Mote

The information for Vision Mote [23] is organized under the following headings:

4.9.1 Hardware

Hardware architecture of Vision Mote [57] is presented in Fig. 11. Vision Mote has 128 MB Flash and 64 MB SDRAM memory. An unspecified CMOS camera is used for visual data acquisition.

4.9.2 Software/Protocols

Mote is operated on Linux OS. OpenCV [35] machine vision library based on Video for Linux (V4L) is provided for video capture and analysis. Communication with the base station is realized through ZigBee protocol. There is also an energy control API for application developers.

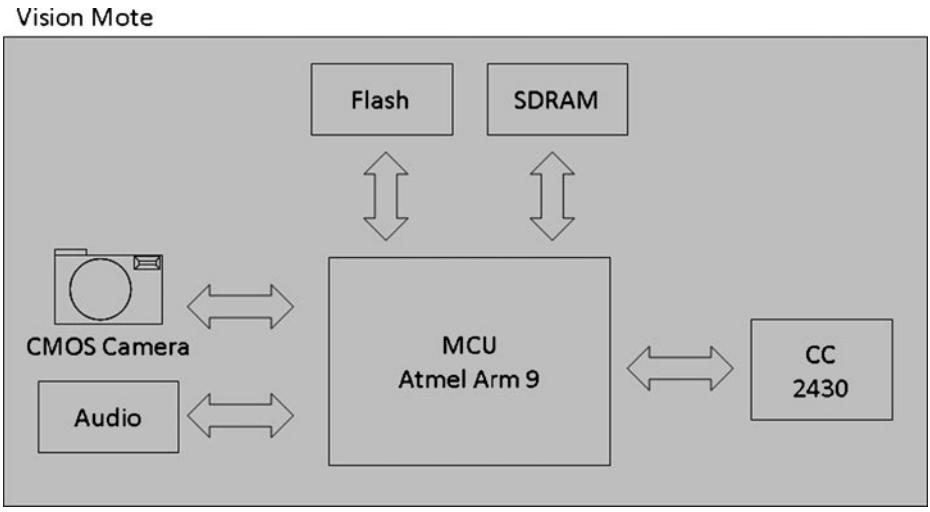


Fig. 11 Hardware architecture of Vision Mote

4.9.3 Applications and processing techniques

Vision Mote is developed for water conservatory engineering applications. The goal is to acquire multi-view image or video information of FoV easily. JPEG is used as the compression algorithm. Processing time for JPEG coded image is reported to be 10 ms and 16 ms for 320×240 resolution and 640×480 resolution, respectively.

4.9.4 Notable features and/or functionalities

The result of an experiment which measures the wireless transmission rate is reported. Although in theory ZigBee supports transmission rates up to 250 Kbps, in practice maximum transmission rate achievable is shown to be only 35 Kbps.

4.10 Other platforms

In addition to the nine aforementioned VSN platforms, there are several other platforms with imaging and wireless communications capabilities such as eCAM [36], WiCa [24] and VideoWeb [34]. However, these platforms are either not sufficiently characterized or do not satisfy our criteria to be a VSN platform (e.g., VideoWeb is not battery-operated). In this subsection we provide summarized information for these platforms.[3]

eCAM [36] is a miniature platform (total size of the platform is less than 1 cm^3) created by integrating a VGA video camera with the Eco networking platform. The camera module (C328-7640) can operate as a video camera or as a still camera with built-in JPEG compression capability. The image sensor within the camera is OmniVision OV7640. The camera module supports various image resolutions (GVA/CIF/160×128/80×64) and it can capture up to 30 fps. Network platform includes a wireless radio (Nordic VLSI nRF24E1),

[3] There are many other smart camera platforms, which are not specifically designed as standalone VSN platforms (a comprehensive survey of smart camera platforms is presented in [41]).

which operates at 2.4 GHz range and can transmit up to 1 Mbps data rate. eCAM is designed for applications that require very small form factor platforms; therefore, it has very limited in-node processing capabilities. Hence it is not in the same class with the aforementioned VSN platforms.

WiCa [24] is a wireless smart camera based on an SIMD (Single Instruction Multiple Data) processor, which is a technique employed to achieve data level parallelism. WiCa has four main components: VGA color image sensor, SIMD processor (IC3D—a Xetal family of SIMD processor) for low level image processing suitable for parallel processing, a general purpose processor for higher level operations (ATMEL 8051), and a communications module (Phillips Aquis Grain ZigBee module developed around ChipCon CC2420). Both processors have access to a dual port RAM that enables them to share a common workspace, which enables both processors to collectively use the data and even pipeline the processing of data in a flexible manner. The software for IC3D is written in C++ and they are designed in such a way that any image processing operation performed on each image line is completed within a single clock cycle. Such an approach is necessary to utilize the full capacity of an SIMD processor. WiCa platform is used in several applications including distributed face detection. WiCa is promising VSN platform with its unique design and its use of an SIMD processor rather than an FPGA chip for low-level image processing operations. However, energy dissipation characteristics of WiCa (i.e., energy dissipation measurements for the whole platform) are not reported, thus, we do not include WiCa among the VSN platforms we use for platform categorization.

VideoWeb [34] is a software-reconfigurable high-bandwidth wireless network architecture using IP cameras and wireless bridges and routers. No on-camera processing is performed in VideoWeb approach; instead tiered-processing of the video data is achieved with low-level processing servers (e.g., for detecting moving objects) and high-level processing servers (e.g., for face recognition). Multi-objective optimization on video resolution and compression quality are performed to provide insights on performance trade-offs. The designers of VideoWeb choose to use power adapters considering that for long-term applications battery-powered cameras are not appropriate.

5 Taxonomy of VSN platforms

It is possible to categorize the VSN platforms presented in Section 4 according to different attributes they possess. Such taxonomies provide a better understanding of the design and operation of VSN platforms.

5.1 Integration

According to the level of integration, VSN platforms can be divided into three categories: device-level integration, multiple-board-level integration, and single-board-level integration.

In the *device-level integration approach*, a VSN platform is constructed by integrating off-the-shelf components (e.g., Panoptes and Meerkats). Components of the platform are all off-the-shelf components, which are presented in Figs. 5 and 6. The Panoptes node consists of an Applied Data Bitsy board utilizing a webcam and an IEEE 802.11 based networking card. The Meerkats node is based on the Crossbow Stargate platform, an IEEE 802.11b wireless card, and a webcam.

In *multiple-board-level integration*, image acquisition and the processing subsystem and radio transceiver subsystem are separately designed and integrated. A video acquisition and

processing board is designed as a daughter board of the networking board. Cyclops (Fig. 3), FireFly Mosaic (Fig. 7), XYZ-ALOHA (Fig. 9), and CITRIC (Fig. 10) VSN nodes are built with integration of video acquisition and processing boards with Mica2, XYZ, FireFly, and Tmote Sky general purpose sensor platforms, respectively.

In *single-board-level design*, all functionalities of a VSN platform including the transceiver and imager are implemented on a single board e.g., MeshEye (Fig. 4), MicrelEye (Fig. 8) and Vision Mote (Fig. 11). MeshEye, MicrelEye, and Vision Mote are built specifically for visual sensor networking on a single board. The central processing units are at the center of the operation of the nodes, which are directly connected to the transceiver, external memory, and the camera(s).

5.2 Data processing hardware

Data acquisition and processing for VSN requires computation and memory resources significantly beyond those that can be provided by most of the generic sensor platforms (e.g., Mica, Telos). Some VSN nodes are equipped with relatively powerful processors and large memory components to meet the requirements of the application demands (i.e., Panoptes and Meerkats). However, the energy dissipations of these platforms are high (several Watts).

To achieve both energy efficiency and computational capabilities powerful enough to handle imaging operations, an alternative architecture for VSN platforms has emerged (e.g., Cyclops). Rather than using general-purpose processors for all of the image-related, computationally-involved or high frequency operations, the burden of performing low-level operations are carried out directly on specially designed hardware components. For example, in Cyclops architecture, specialized operations such as on-demand access to high speed clocking at capture time, calculating image statistics and background subtraction is performed by a CPLD rather than by the main processor.

Visual sensor platforms can benefit from reconfigurable processing engines, which include microprocessor, FPGA, and SRAM. Since inter-device PCB connections are eliminated by using such processing engines, energy consumption is reduced when compared to externally connected microprocessor and FPGA chips. Furthermore, the hybrid processor + FPGA architecture is optimal for applications that do not need the computational power provided by higher-end FPGAs with high-energy consumption. The MicrelEye platform is designed with such a processing engine (8-bit microcontroller, FPGA with 70 Kgates, and 36 KB SRAM).

Rather than utilizing a lightweight processor for high energy efficiency (e.g., Cyclops) or a high performance processor with low energy efficiency (e.g., Panoptes), a middle way can be followed by using a medium weight processor with moderate energy dissipation characteristics. In FireFly Mosaic, CITRIC, MeshEye and VisionMote 32-bit processors with relatively higher operating frequencies are used for both image capture and wireless networking functions.

5.3 Energy dissipation

While the energy consumption of WSN nodes is dominated by communication, this is generally not true in VSN nodes. For example, it was shown that communication energy dissipation constitutes 91% and 62% of the total energy dissipation in Telos and MicaZ scalar sensor nodes, respectively [40]. Unlike in scalar sensor nodes, energy dissipation in visual sensor nodes is dominated by the computation energy rather than the communication

energy. For example, communication energy dissipation constitutes only 22% of the total energy dissipation of MicrelEye VSN platform [23]. Since vision involves handling large chunks of data, visual sensors are typically power hungry and vision algorithms are computationally expensive [40]. Therefore, the hardware that is suitable for handling such computational challenges is liable to exhaust the limited energy sources of sensors quickly, unless they are configured appropriately. In sensor network applications, vision can only play a meaningful role if the power consumption of the nodes is not significantly increased [40].

The energy dissipation characteristics of VSN platforms described in Section 4 are collected and presented in Table 2. Note that we present the maximum and minimum (if available) energy dissipation values reported for the platforms investigated. We do not include more detailed energy dissipation characteristics (e.g., energy cost of one picture transmission) since such measurements are not reported for most of the platforms.

Cyclops energy dissipation is the lowest, due to the fact that Cyclops is designed specifically for energy efficiency. It has a lightweight 8-bit processor and small amount of memory suitable for low speed processing and low-resolution images. Furthermore, the whole system is designed to support the capability for automatic relaxation of each subsystem to its lower power state adaptively. However, we note that even the most basic image processing application dissipates more energy with lower capability processors than they dissipate with moderate capability processors [18].

The XYZ-ALOHA platform's power dissipation is mostly due to the XYZ sensor node's operation, since the ALOHA imaging module is not a conventional imaging system and its power dissipation is extremely low. On the other hand, most of the power dissipation of the FireFly Mosaic platform is due to the imaging subsystem (i.e., CMUCam3), and the combined power dissipation by the radio and the processor running the radio is less than 13% of the total power in active mode. Likewise in CITRIC when the imaging board is in idle mode, less than 20% of the energy dissipation is due to the networking platform (when the imaging board is active energy dissipation of the imaging board constitutes an even larger percentage of the total energy dissipation).

Panoptes and Meerkats are designed as high-end platforms with powerful computational capabilities, and their power dissipation figures are approximately an order of magnitude higher than the rest of the platforms. It should be noted that both of these platforms are equipped with IEEE 802.11 radios and capable of transmitting at least an order of magnitude, larger volume of data than the IEEE 802.15.4 radios used in the lower power platforms.

5.4 Radios

There are four type of radios used in the platforms we investigated: (a) CC1000; (b) CC2420 or CC2430; (c) Bluetooth; and (d) IEEE 802.11. The Mica2 node, which is the host for Cyclops, uses CC1000 radio, which supports a maximum of 38.4 Kbps raw channel rate. MeshEye, XYZ (XYZ-ALOHA), FireFly (FireFly Mosaic), Tmote Sky (CITRIC) and Vision Mote are equipped with IEEE 802.15.4/ZigBee compliant CC2420 or CC2430, which supports 250 Kbps data rate. Although the data rate supported by CC2420 is higher than CC1000, it is still not high enough to transfer reasonable quality images in real time. We note that transmission rate achievable in practice is significantly lower than the theoretical maximum of 250 Kbps [57]. A JPEG-compressed 176×144 (QCIF) image requires 7 KB of memory, which is more than 50 128-Byte IEEE 802.15.4 packets [42]. Note that at most 4.3 QCIF frames can be transmitted per second with a 250 Kbps channel.

MicrelEye node is equipped with a Bluetooth transceiver, which is chosen for its low power consumption. Since Bluetooth devices come with several usage profiles, it is easy to utilize them in embedded applications. Although it is possible to reach data rates up to 704 Kbps over RFComm standard, in MicrelEye 230.4 Kbps data rate is utilized. Nevertheless, the power dissipation is approximately 40% more than CC2420 power dissipation [19, 23].

Both Panoptes and Meerkats utilize IEEE 802.11 transceivers. Since both platforms are equipped with powerful processing hardware and are capable of generating real-time video, IEEE 802.15.4 or Bluetooth transceivers cannot cope with such a large amount of data. Thus, IEEE 802.11 transceivers are the best alternative (if not only) for real-time video streaming. However, the power dissipation of an IEEE 802.11b transceiver is more than an order of magnitude larger than the power dissipation on an IEEE 802.15.4 transceiver.

5.5 Operating system and software architecture

We can classify Operating Systems (OS) utilized in VSN platforms into three categories: (a) general purpose OS; (b) OS specifically designed for sensor networks; and (c) Finite State Machine (FSM) based resource management (no OS). Each category of OS has its benefits and drawbacks.

Panoptes, Meerkats, CITRIC, and Vision Mote platforms use Linux operating system. Linux provides the flexibility to alter and modify the part of the system according to the specific needs of the applications. Furthermore, there are many available vision computing algorithms implemented in C/C++ (e.g., OpenCV library, IJG library) and routing protocols available for Linux (e.g., DSR routing protocol). Thus, for ease of programming, interfacing and rapid prototyping utilization of a general purpose OS (e.g., Linux) is highly advantageous.[4] For example, in Panoptes platform specific functions (JPEG compression routines) are implemented in high-level languages (e.g., C) and natively compiled. And the modularized code pieces are connected via function calls by using an interpreted programming language (i.e., Python). This approach eliminates the need for recompiling all the code. Nevertheless, using a general purpose OS brings extra overhead when compared to an application specific OS (e.g., TinyOS). VSN platforms using a general purpose OS instead of an application-specific OS are all equipped with PDA-class processors, thus, the energy dissipation for these platforms are significantly higher than the other platforms.

Three of the VSN platforms use application specific operating systems—Cyclops, FireFly Mosaic, and XYZ-ALOHA platforms use TinyOS [49], Nano-RK [33], and SOS [45] operating systems, respectively. TinyOS is an event based operating system designed to support the concurrency intensive operations required by sensor network platforms with minimal hardware requirements. TinyOS has a synchronous execution model, where a synchronous task blocks the computational resources unless it is not preempted by an interrupt handler. Vision computing can take extended computation times, hence, in Cyclops there are two processors—one for image processing (on the imaging board) and the other is for networking (on the networking board). Nano-RK is a real-time resource centric operating system, which provides interfaces for globally synchronized task processing. This framework supports synchronous image capturing and transmission scheduling without sacrificing the overall timing constraints. SOS is an operating system for mote-class wireless sensor networks, which follows an event driven design similar to TinyOS. The kernel used in SOS

[4] There are some real-time embedded operating systems that provide a full-compliance POSIX interface (e.g., LynxOS, QNX), the same functionalities as Linux, but with less overhead. However, they were not used in VSN platforms presented in this paper.

implements messaging, dynamic memory, module loading and unloading, along with other services. Unlike TinyOS, which compiles a static OS image, SOS uses dynamically loaded software modules to create a system supporting dynamic addition, modification, and removal of network services, which supports the installation and modification of programs at runtime.

TinyOS is the most popular OS for networked embedded sensor network platforms. However, its use may require extra development time. For example, in Cyclops device drivers are implemented specifically for TinyOS and require extra efforts. SOS on the other hand is not under active development and at its web site it is recommended to use one of the more actively supported alternatives. On the other hand, there are many studies in the literature [31] (e.g., Contiki [13]) that aims to design an operating system for sensor nodes having lightweight mechanisms and abstractions that provide a rich enough execution environment while staying within the limitations of the constrained devices. Contiki OS [13] enables sensor nodes to communicate over IP (both IPv4 and IPv6) and has been used in a variety of projects which involves scalar sensor nodes. Up to our best knowledge, it has not been ported to any VSN platform yet.

MicrelEye and MeshEye platforms does not use an operating system instead resource management and process scheduling are performed through FSM-based approaches. Such an approach have advantages in terms of speed and energy efficiency, however, the user would need to program in a hardware description language, making algorithm implementation and debugging a time-consuming process [6].

Vision computing is an established research area; however, utilization of high level vision algorithms requires intensive efforts on currently available VSN platforms. Several architectures are introduced to utilize high-level vision algorithms effectively and efficiently [9, 18]. WiSNAP (Wireless Image Sensor Network Application Platform) is a Matlab-based application development architecture for VSN platforms. WiSNAP is intended to accelerate algorithm development by providing standardized and easy-to-use APIs to interface various components (e.g., image sensor, wireless radios) of a VSN platform. WiSNAP framework currently includes device libraries for Agilent ADCM-1670 and ADNS-3060 image sensors and ChipCon CC2420 radio. Sensor Data Library (sdlib) [9] is a library of well-tested and reusable nesC components that capture the fundamental high-level operations (similar to the well-known communication abstractions in TCP/IP service model). This library allows a convenient toolbox for rapid development of sensor applications at nesC language.

5.6 Hierarchy

The concept of hierarchy in the context of VSNs can be explained best by considering a visual surveillance application [25] where the sensors implement three tasks in multiple tiers of a hierarchical structure: object detection in the first tier, object recognition in the second tier and finally object tracking in the third tier. A hierarchical network structure (see Fig. 12) enables both cost savings and energy efficiency for aforementioned tasks. It is widely accepted that for optimal energy-efficient operation of a scalable VSN application, the network should be structured functionally similar to the hierarchical network organization proposed in [25], which consists of three levels (tiers) of platforms. Object detection in a visual surveillance application can be achieved by low cost and low energy dissipation lightweight VSN platforms (e.g., Cyclops uses frame differencing and background subtraction for detecting moving objects, XYZ-ALOHA platform utilizes AER concept for motion detection, MicrelEye uses an SVM-like technique for people detection, and MeshEye uses frame differencing and stereo matching for moving object detection). Once an object is detected, second tier nodes with higher computation capabilities perform object recognition.

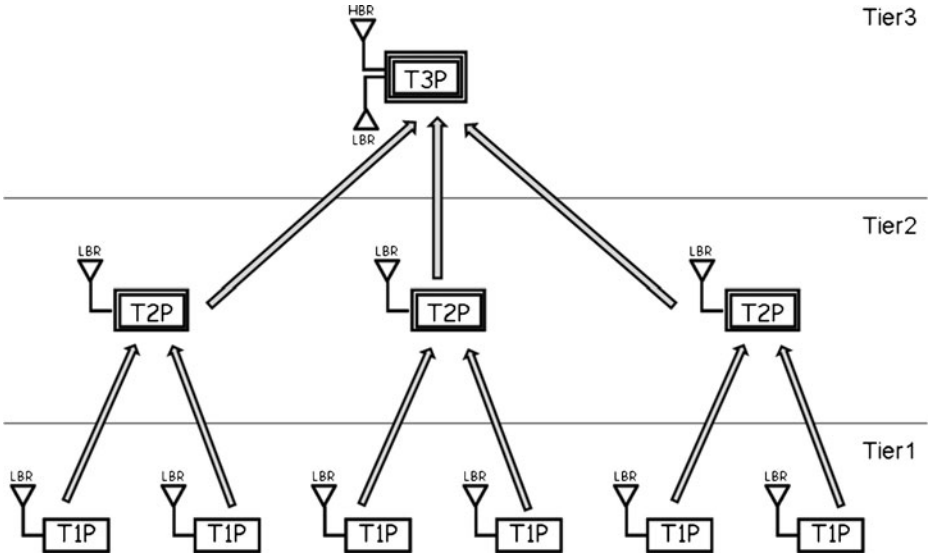


Fig. 12 Hierarchical organization of a VSN (T1P: Tier 1 Platform, T2P: Tier 2 Platform, T3P: Tier 3 Platform, LBR: Low Bandwidth Radio, HBR: High Bandwidth Radio)

MicrelEye, MeshEye, FireFly Mosaic, CITRIC and Vision Mote platforms are capable of acting as second tier nodes. The third task involves higher resolution cameras, which track the object, send streaming video, and hand over the tracking task to next set of cameras in the path of the object. Panoptes and Meerkats platforms are suitable for third tier utilization. This example poses the main computation challenges that a VSN platform has to deal with. Table 4 presents a comparison of VSN platforms with respect to their typical tasks in a hierarchical network organization.

5.7 Applications

There are many applications envisioned for VSN platforms (Table 3 lists only the applications specifically targeted by each platform). Application scenarios include vehicle

Table 4 Hierarchy, Typical Tasks and corresponding VSN Platforms

Hierarchy	Typical task	VSN platforms
Tier 1	Object detection	Cyclops
		XYZ-ALOHA
		(MeshEye MicrelEye)
Tier 2	Object recognition	MeshEye[a]
		Firefly Mosaic
		MicrelEye[a]
		CITRIC
		Vision Mote
Tier 3	Object tracking	Panoptes
		Meerkats

[a] Can also be used as Tier 1 platform

tracking applications, environmental observation systems [14], security and surveillance applications [14], emergency response systems [14], assisted living applications and smart homes [5, 42, 44], water conservatory engineering [57], virtual reality and telepresence systems [5, 44] and smart meeting rooms [44]. By taking a holistic view, we can see that all these applications can be realized by implementing three sets of tasks: (a) object and event detection and localization; (b) recognition of these objects and events; (c) tracking, monitoring and real-time video delivery. It is of no surprise that these tasks match with those already mentioned when we introduce the concept of hierarchical network organization in previous subsection.

Panoptes and Meerkats platforms are the best matches for use in the highest tier (tier3) and for real-time video surveillance and object tracking purposes due to their relatively high bandwidth radios (i.e., IEEE 802.11) and ample processing and storage capabilities.

MeshEye, FireFly Mosaic, MicrelEye, CITRIC, and Vision Mote platforms can be used as tier2 platforms and for object and event recognition. MeshEye has the lowest energy dissipation among these platforms. With its multiple types of cameras on the same platform, it can also be used as a tier-1 platform (low resolution cameras for object detection and high resolution camera for high resolution image acquisition for object recognition).

Cyclops and XYZ-ALOHA are two platforms most suitable as tier1 platforms for object and event detection and object localization applications. In a multi-tiered architecture, sensor nodes which do not have the ability to capture visual information can also be used for detection purposes. For instance, upon sensing an event with a PIR sensor a low-cost sensor node can trigger a tier2 platform which takes high resolution pictures and performs complex object recognition processing. PIR sensors are up to two orders of magnitude more energy-efficient than cameras [42].

6 Critical evaluation and open problems

Direct comparison of the nine VSN platforms presented is not meaningful. Because as it is stated in [3] and [40] different platforms are designed with functionalities complementing each other. However, we can compare platforms in the same functional class.

Energy dissipation of tier3 platforms is roughly at least an order of magnitude higher than the rest of the platforms. Meerkats dissipates approximately 30% less power than Panoptes when both devices are in their highest energy dissipation mode. Meerkats' energy dissipation at the lowest energy dissipation state is approximately 14% less than that of Panoptes in its lowest energy dissipation state. The difference is due to Meerkats' use of more recent components (e.g., Panoptes uses Applied Data Bitsy board and later StrongARM, whereas Meerkats uses XScale). Both Panoptes and Meerkats use Linux OS and they are designed with a device-level integration approach.

Among all the tier2 platforms, CITRIC has the highest computational resources and its energy dissipation is also the highest. Since there is no OS running on MeshEye, it requires special effort to reprogram this platform for different applications. MicrelEye is also a platform without an OS governing the operation of the platform. MeshEye and MicrelEye are single board designs, thus, they are designed with a cross-layer methodology. FireFly Mosaic energy dissipation is higher than MicrelEye and lower than CITRIC. FireFly Mosaic's OS (Nano-RK) and link protocol (RT-link) are custom-designed. Both CITRIC and FireFly Mosaic platforms are designed with multiple-board-level integration approach. Vision Mote, CITRIC, MeshEye, and FireFly Mosaic are equipped with 250 Kbps CC2420 or CC2430 radios. Although MicrelEye is equipped with a Bluetooth radio, it is used with 230 Kbps data rate.

Cyclops has the lowest energy dissipation characteristics among all the platforms. It also has the lowest data rate radio (CC1000—38.4 Kbps). Yet, it has sufficient amount of computational power for tasks envisioned for tier1. XYZ-ALOHA energy dissipation is more than double the energy dissipation of Cyclops; however, most of the energy dissipation is due to the networking board. AER concept demonstrated on XYZ-ALOHA platform has potential to reduce the energy dissipation below the Cyclops energy dissipation level for tier1 operations if integrated with a more energy efficient networking board.

Each VSN platform introduced in Section 4 is designed with a consideration of specific functionalities and application(s) (see Tables 3 and 4). Since these platforms are mainly implemented as proof-of-concept and not commercialized, many details that could be useful for application developers such as cost information are not available. While many commercial versions of WSN platforms are available [52], none of the VSN platforms presented in this paper can be purchased from vendors. If application developers aim to decrease design effort by purchasing off-the-shelf components, they can choose among three alternatives:

- Buying a general-purpose sensor platform and attaching imagers to it. This is the most appropriate choice if the focus is more on system aspects but not on hardware design [3]. The designers of Panoptes [14] and Meerkats [3] have preferred this approach. The platforms they use utilize a webcam attached to the Stargate board from Crossbow Technology [10].
- Purchasing a vision sensor platform and integrating it with a networking board. FireFly Mosaic [42] team has designed a sensor board named FireFly having a CC2420 radio and integrated it with CMUcam3 [43]. CMUcam3 is a vision sensor platform without networking functions available from multiple international commercial vendors for a cost of approximately US$239 [43].
- Buying a general-purpose sensor platform and integrating it with a custom-designed image acquisition and processing board (e.g., Cyclops [40] uses Mica2 [10] and CITRIC [6] uses Tmote Sky [30]).

There are various interesting open problems to be studied ahead. We present some of these in the rest of this section.

The main motivation for implementing in-network processing techniques and compression/coding in VSN platforms is that the complexity involved with acquiring raw images and transmitting large volumes of data impose limitations on cost, scalability, network lifetime and robustness of the sensor networks. On the other hand, in the context of VSN applications it would not be correct to assume that computation is always cheaper than communication and thus preferred. Hence, it is important to balance the tradeoff between amount of reduction achieved with computation on VSN platforms and amount of data transmitted to the base station(s) [47, 56].

In a recent study [46], using a linear programming framework, it was shown that neither compressing all data nor completely avoiding data compression achieve the longest possible network lifetime. Dynamic data transformation is shown to achieve significantly longer network lifetimes than the lifetimes obtained with pure strategies. Modeling communication/computation tradeoffs as optimization problems could provide insights for the energy-optimal operation of VSNs. But these models become more useful if they are fed by elaborate and realistic performance data obtained experimentally for VSN platforms. Therefore we think that more energy-efficient VSNs could be attained by a close cooperation between practitioners and system optimizers. A promising future work on this

area could be investigating whether general cost models covering both computation and communication aspects could be developed or these models should be customized for each platform separately.

Performance results of video transfer (as opposed to still images) were reported only for FireFly Mosaic and Panoptes. In the experiment of assisted living application with FireFly Mosaic, the nodes run just five days from AA batteries [42]. The reported value of the power required for Panoptes [14] is even higher; an order of magnitude greater than FireFly Mosaic due to its use of higher-bandwidth and more energy-hungry 802.11 based networking card. These results indicate that extending the lifetime of battery-powered real-time video applications continues to be one of the greatest challenges that lie ahead of VSN researchers.

In our view, maintaining privacy may be critical to the success of future VSN applications in real-life settings considering that camera violates privacy more than other sensors [48] and privacy-enhancing techniques like video-masking may introduce extra overhead [16]. An efficient and promising strategy is to take privacy issues into consideration not as an after-thought but from the start. For instance XYZ-ALOHA could satisfy both privacy and efficiency requirements by the use of address-event imagers [48].

For ease of deployment and increased flexibility of VSN platforms, programming should be simple enough to be used by application developers who are experts in their domains but not in technology [31]. For achieving the ultimate goal of plug-and-play functionality, more effort is definitely required. With the next-generation of programming solutions, we expect to see more VSNs deployed in real-life.

7 Conclusions

We have provided an overview of current platforms developed for VSNs according to characteristics such as level of integration, data processing hardware, energy dissipation, and radios. Vision applications necessitate the use of specialized platforms that can provide: (a) enough bandwidth to support real-time vision data; (b) enough memory to buffer large amount of data packets and visual data; (c) processing hardware that can handle visual data acquisition and information processing. Powerful processors, large memory, and high bandwidth are necessary for the better performance of vision computing algorithms, which contradict the energy efficiency requirements on the VSN platforms. Hence, there are inherent trade-offs between energy efficiency and performance depending on resolution, vision-computing, compression and coding algorithms, performed over visual data. Our conclusions on the current state-of-the-art of VSN platforms are itemized as follows:

- There are three categories of VSN platforms according to the computation power: lightweight platforms, intermediate platforms, and PDA-class platforms. Only PDA-class platforms can support real-time video streaming, which dissipate several Watts of power. Intermediate platforms dissipate less than one Watt power. They cannot support real-time video (i.e., the frame rate supported by these platforms are a few fps). However, they can provide real-time visual information (e.g., the trajectory of a moving object). Lightweight platforms dissipate in the order of a few hundred mW. However, they are highly constrained in terms of both computational capabilities and the bandwidth they can support.
- Application-specific cross-layer design is of utmost importance, given the severe limitations imposed on the VSN platforms. Hence, the efficiency of a VSN application is optimized if the utilized platforms are specifically designed for the application.

- Hierarchical organization of VSN with a tight coupling between the heterogeneous platforms in different layers of the hierarchy is the optimal organizational decision for both energy efficiency and effectiveness.
- Compression, coding, and vision computing algorithms designed for general purpose computing platforms which are not as constrained as VSN platforms cannot be directly utilized without sacrificing energy efficiency in VSN platforms. Therefore, optimizations of these techniques for VSN platforms are crucial.

The success of VSNs depends on the efficiency of the enabling technologies. Design and development VSN platforms will determine the future of VSN applications.

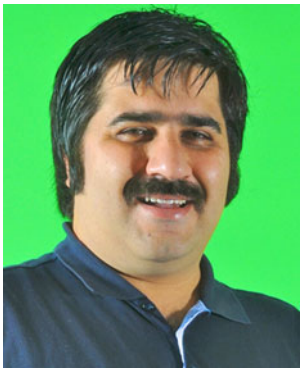
Acknowledgements This work was partially supported by Spanish projects TIN2010-21378-C02-01 and 2009-SGR-1167. We would like to thank Ugur Cil for his help on figure drawings. We would like to thank the anonymous reviewers for their helpful comments, feedbacks, and suggestions.

References

1. Atmel AT 91 Sam9261 Datasheet, http://www.keil.com/dd/docs/datashts/atmel/at91sam9261_dc.pdf
2. Bajwa W, Haupt J, Sayeed A, Nowak R (2006) Compressive wireless sensing. In Proc. International Conference on Information processing in Sensor Networks
3. Boice J, Lu X, Margi C, Stanek G, Zhang G, Manduchi R, Obraczka K (2006) Meerkats: a power-aware, self-managing wireless camera network for wide area monitoring. In: Proc. Workshop on Distributed Smart Cameras
4. Candes EJ (2006) Compressive sampling. In: Proc. International Congress of Mathematicians
5. Charfi Y, Wakamiya N, Murata M (2009) Challenging issues in visual sensor networks. IEEE Wirel Commun 16:44–49
6. Chen P, Ahammed P, Boyer C, Huang S, Lin L, Lobaton E, Meingast M, Oh S, Wang S, Yan P, Yang AY, Yeo C, Chang LC, Tygar D, Sastry SS (2008) CITRIC: a low-bandwidth wireless camera network platform. In: Proc. International Conference on Distributed Smart Cameras, pp 1–10
7. Chong CY, Kumar SP (2003) Sensor networks: evolution, opportunities, and challenges. Proc IEEE 91:1247–1256
8. Chou PA, Wu Y (2007) Network coding for the internet and wireless networks. IEEE Signal Proc Mag 5:77–85
9. Chu D, Lin K, Linares A, Nguyen G, Hellerstein JM (2006) Sdlib: a sensor network data and communications library for rapid and robust application development. In: Proc. International Conference on Information processing in Sensor Networks, pp 432–440
10. Crossbow Technology, http://www.xbow.com.
11. Donoho DL (2006) Compressed sensing. IEEE Trans Inf Theory 52:1289–1306
12. Duarte MF, Wakin MB, Baron D, Baraniuk RG (2006) Universal distributed sensing via random projections. In: Proc. International Conference on Information processing in Sensor Networks
13. Dunkels A, Gronvall B, Voigt T (2004) Contiki—a lightweight and flexible operating system for tiny networked sensors. In: Proc. IEEE International Conference on Local Computer Networks, pp 455–462
14. Feng WC, Kaiser E, Shea M, Feng WC, Baillif L (2005) Panoptes: scalable low-power video sensor networking technologies. Trans Multimed Comput Comm Appl 1:151–167
15. Girod B, Aaron A, Rane S, Rebollo Monedero D (2005) Distributed video coding. Proc IEEE 93:71–83
16. Guerrero-Zapata M, Zilan R, Barcelo Ordinas JM, Bicakci K, Tavli B (2010) The future of security in wireless multimedia sensor networks: a position paper. Telecommun Syst 45:77–91
17. He Z, Wu D (2006) Resource allocation and performance analysis of wireless video sensors. IEEE Trans Circ Syst Video Tech 16:590–599
18. Hengstler S, Aghajan H (2006) WiSNAP: a wireless image sensor network application platform. In: Proc. International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities
19. Hengstler S, Prashanth D, Fong S, Aghajan H (2007) MeshEye: a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. In: Proc. International Conference on Information processing in Sensor Networks
20. Hill J, Horton M, Kling R, Krishnamurthy L (2004) The platforms enabling wireless sensor networks. Commun ACM 47:41–46

21. Hu W, Tan T, Wang L, Maybank S (2004) A survey on visual surveillance of object motion and behaviors. IEEE Trans Syst Man Cybern C Appl Rev 34:334–352
22. Kang LW, Lu CS (2007) Multi-view distributed video coding with low-complexity inter-sensor communication over wireless video sensor networks. In: Proc. IEEE International Conference on Image Processing, vol. 3, pp 13–16
23. Kerhet A, Magno M, Leonardi F, Boni A, Benini L (2007) A low-power wireless video sensor node for distributed object detection. J Real-Time Image Process 2:331–342
24. Kleihorst R, Abbo A, Schueler B, Danilin A (2007) Camera mote with a high-performance parallel processor for realtime frame-based video processing. In: Proc. International Conference on Distributed Smart Cameras, pp 109–116
25. Kulkarni P, Ganesan D, Shenoy P, Lu Q (2005) SensEye: a multi tier camera sensor network. In: Proc. ACM International Conference on Multimedia, pp 229–238
26. Lu Q, Luo W, Wang J, Chen B (2008) Low-complexity and energy efficient image compression scheme for wireless sensor networks. Comput Netw 52:2594–2603
27. Margi CB, Petkov V, Obraczka K, Manduchi R (2006) Characterizing the energy consumption in a visual sensor network testbed. In: Proc. International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities
28. Mica High Speed Radio Stack, http://www.tinyos.net/tinyos-1.x/doc/stack.pdf.
29. Misra S, Reisslein M, Xue G (2008) A survey of multimedia streaming in wireless sensor networks. IEEE Commun Surv Tutorials 10:18–39
30. Moteiv Corporation Tmote Sky Datasheet, http://www.snm.ethz.ch/Projects/TmoteSky.
31. Mottola L, Picco GP (2011) Programming wireless sensor networks: fundamental concepts and state of the art, (to appear in) ACM Computing Surveys
32. Murat UM, Sclaroff S (2004) Optimal placement of cameras in floorplans to satisfy task requirements and cost constraints. In: Proc. Workshop on Omnidirectional Vision, Camera Networks, and Non-classical Cameras
33. Nano-RK: A Wireless Sensor Networking Real-Time OS—http://www.nano-rk.org/
34. Nguyen H, Bhanu B, Patel A, Diaz R (2009) VideoWeb: design of a wireless camera network for real-time monitoring of activities. In: Proc. ACM/IEEE International Conference on Distributed Smart Cameras, pp 1–8
35. OpenCV, http://www.opencv.org
36. Park C, Chou PH (2006) eCAM: ultra compact, high data-rate wireless sensor node with a miniature camera. In: Proc. ACM Conference on Embedded Networked Sensor Systems, pp 359–360
37. Pereira F, Torres L, Guillemot C, Ebrahimi T, Leonardi R, Klomp S (2008) Distributed video coding: selecting the most promising application scenarios. Signal Process: Image Commun 23:339–352
38. Pottie GJ, Kaiser WJ (2000) Wireless integrated network sensors. Commun ACM 43:551–558
39. Puri R, Majumdar A, Ishwar P, Ramchandran K (2006) Distributed video coding in wireless sensor networks. IEEE Signal Process Mag 23:94–106
40. Rahimi M, Baer R, Iroezi OI, Garcia JC, Warrior J, Estrin D, Srivastava M (2005) Cyclops: in situ image sensing and interpretation in wireless sensor networks. In: Proc. ACM Conference on Embedded Networked Sensor Systems, pp 192–204
41. Rinner B, Winkler T, Schriebl W, Quaritsch M, Wolf W (2008) The evolution from single to pervasive smart cameras. In: Proc. International Conference on Distributed Smart Cameras, pp 1–10
42. Rowe A, Goal D, Rajkumar R (2007) FireFly Mosaic: a vision-enabled wireless sensor networking system. In Proc. IEEE International Real-Time Systems Symposium, pp 459–468
43. Rowe A, Goode A, Goel D, Nourbakhsh I (2007) CMUcam3: an open programmable embedded vision sensor. Carnegie Mellon Robotics Institute Technical Report, RI-TR-07-13
44. Soro S, Heinzelman W (2009) A survey of visual sensor networks. Adv Multimed 2009, Article ID 640386
45. SOS: An OS for mote-class wireless sensor networks—https://projects.nesl.ucla.edu/public/sos-2x/doc/
46. Tavli B, Bagci IE, Ceylan O (2010) Optimal data compression and forwarding in wireless sensor networks. IEEE Commun Lett 14:408–410
47. Tavli B, Kayaalp M, Ceylan O, Bagci IE (2010) Data processing and communication strategies for lifetime optimization in wireless sensor networks. AEU Int J Electron Commun 64:992–998
48. Teixeira T, Culurciello E, Park JH, Lymberopoulos D, Sweeney AB, Savvides A (2006) Address event imagers for sensor networks: evaluation and modeling. In Proc. International Conference on Information processing in Sensor Networks, pp 458–466
49. TinyOS: An OS for networked sensors—http://tinyos.millennium.berkeley.edu
50. Wang Z, Sheikh HR, Bovik AC (2003) Objective video quality assessment. In: Furht B, Marques O (eds) The handbook of video databases: design and applications. CRC Press, Boca Raton
51. Wang H, Wang W, Wu S, Hua K (2010) A survey on the cross-layer design for wireless multimedia sensor networks. Social Informatics and Telecommun Eng, LNCS 48:474–486
52. WSN Platforms, http://wsn.oversigma.com/wiki/index.php?title=WSN_Platforms.

53. Wu M, Chen CW (2007) Collaborative image coding and transmission over wireless sensor networks. EURASIP Journal on Advances in Signal Processing, Article ID: 70481
54. Wu Y, Stankovic V, Xiong Z, Kung SY (2005) On practical design for joint distributed source and network coding. In: Proc. Workshop on Network Coding, Theory and Applications
55. Yick J, Mukherjee B, Ghosal D (2008) Wireless sensor network survey. Comput Netw 52:2292–2330
56. Yu Y, Krishnamachari B, Prasanna VK (2008) Data gathering with tunable compression in sensor networks. IEEE Trans Parallel Distr Syst 19:276–287
57. Zhang M, Cai W (2010) Vision mesh: a novel video sensor networks platform for water conservancy engineering. In: Proc. IEEE International Conference on Computer Science and Information Technology, pp 106–109



Bulent Tavli is an Associate Professor in the Department of Electrical and Electronics Engineering at the TOBB University of Economics and Technology, Ankara, Turkey. He received the B.S. degree in Electrical and Electronics Engineering in 1996 from the Middle East Technical University, Ankara, Turkey. He received the M.S. and Ph.D. degrees in Electrical and Computer Engineering in 2001 and 2005 from the University of Rochester, Rochester, NY, USA. Telecommunications, networking, signal processing, and embedded systems are his current research areas.

Kemal Bicakci is currently an associate professor at the Department of Computer Engineering, TOBB University of Economics and Technology. He has obtained his Ph.D. degree from Middle East Technical University, Ankara, Turkey in 2003. Between 2004 and 2006, he was a postdoc researcher in Vrije Universiteit Amsterdam working with Prof. Tanenbaum in EU FP6 project named SecurE-Justice. His

previous research experience includes several NSF funded projects in which he participated as a research assistant during his M.S. studies in University of Southern California, Los Angeles, USA. His research interests include information security, applied cryptography, wireless and sensor networks, health informatics, human–computer interaction and usability.

Ruken Zilan received the BS and MS degrees in Physics respectively, in 1997 from Gazi University and in 2001 from Middle East Technical University, Ankara Turkey. She received her 2nd MS degree in Computer Engineering from TOBB University of Economics and Technology, Ankara Turkey, in 2007. She is currently pursuing her PhD in Computer Architecture Department, Technical University of Catalonia (UPC), Barcelona, Spain, as a member of CompNET Research Group at UPC. She is also a member of Barcelona SuperComputing (BSC) Center, as a CISCO scholar. Her research interests are network processors, internet traffic, wireless networks, wireless multimedia sensor networks.

Jose M. Barcelo-Ordinas is an Associate Professor in the Computer Architecture Department at Universidad Poltècnica de Catalunya (UPC), Barcelona, Spain. He received his PhD and MsC in Telecommunications Engineering at UPC in 1998 and 1991 respectively. He joined the CompNet group at the Computer Architecture Department in 1993. He has participated in several European projects such as EXPLOIT, BAF, EXPERT, NETPERF, MOEBIUS, WIDENS projects, and EuroNGI, EuroNFI and EuroNF (VII FP) Networks of Excellence (NoE). His currently research areas are DTN (Delay Tolerant Networks), VANETs (Vehicular Ad Hoc Networks) and WMSN (Wireless Multimedia Sensor Networks).