MeshEye: A Hybrid-Resolution Smart Camera Mote for Applications in Distributed Intelligent Surveillance

Stephan Hengstler, Daniel Prashanth, Sufen Fong, and Hamid Aghajan
Wireless Sensor Networks Lab
Department of Electrical Engineering
Stanford University, Stanford, CA 94305, United States
{*hengstler*, *daniel.prashanth*, *sffong*, *aghajan*}@stanford.edu

ABSTRACT
Surveillance is one of the promising applications to which smart camera motes forming a vision-enabled network can add increasing levels of intelligence. We see a high degree of in-node processing in combination with distributed reasoning algorithms as the key enablers for such intelligent surveillance systems. To put these systems into practice still requires a considerable amount of research ranging from mote architectures, pixel-processing algorithms, up to distributed reasoning engines. This paper introduces MeshEye, an energy-efficient smart camera mote architecture that has been designed with intelligent surveillance as the target application in mind. Special attention is given to MeshEye's unique vision system: a low-resolution stereo vision system continuously determines position, range, and size of moving objects entering its field of view. This information triggers a color camera module to acquire a high-resolution image sub-array containing the object, which can be efficiently processed in subsequent stages. It offers reduced complexity, response time, and power consumption over conventional solutions. Basic vision algorithms for object detection, acquisition, and tracking are described and illustrated on real-world data. The paper also presents a basic power model that estimates lifetime of our smart camera mote in battery-powered operation for intelligent surveillance event processing.

Categories and Subject Descriptors
B.0 [**Hardware**]: General; C.4 [**Performance of Systems**]: Design Studies; I.4.9 [**Image Processing and Computer Vision**]: Applications

General Terms
Algorithms, Design, Experimentation, Measurement, Performance

Keywords
Distributed Intelligence, Mote Architecture, Power Efficiency, Smart Cameras, Wireless Sensor Networks

IPSN'07, April 25–27, 2007, Cambridge, Massachusetts, USA.

1. INTRODUCTION
Distributed smart cameras have received increased focus in the research community over the past several months. The notion of cameras combined with embedded computation power and interconnected through radio links opens up a new realm of intelligent vision-enabled applications. Real-time image processing and distributed reasoning made possible by smart cameras can not only enhance existing applications but also motivate new applications. Potential application areas range from home monitoring, elderly care, and smart environments to security and surveillance in public or corporate buildings. Critical issues influencing the success of smart camera deployments for such applications include reliable and robust operation with as little maintenance as possible.

In comparison to scalar sensors, such as temperature, pressure, humidity, velocity, and acceleration sensors, vision sensors generate much higher bandwidth data due to the two-dimensional nature of their pixel array. The sheer amount of raw data generated precludes it from human analysis in many applications. Hence distributed intelligent algorithms supported by in-node image processing are required to successfully operate scalable networks of wireless smart cameras. We see the combination of local processing and distributed reasoning as the key challenge in making intelligent vision-enabled applications a reality. As outlined in [1]–[3], local processing calls for adequate low-level pixel processing, which enhances image content, and intermediate-level object processing, which detects and tracks objects and extracts their properties. On the other hand, distributed reasoning requires high-level algorithms, which exchange and compare object information among nodes and make joint decisions [2]. Several examples of low- and intermediate-level algorithms can be found in [1]–[6].

In this paper, we introduce a smart camera mote architecture designed for in-node processing, with the aim of facilitating distributed intelligent surveillance. With this application in mind, our mote architecture targets the provision of sufficient processing power and an adequate vision system while minimizing component count and power consumption. Low power consumption is an important design objective to enable mobile surveillance applications using battery-powered camera motes.

Several mote architectures for wireless image sensor networks with similar objectives have been proposed in the past. In 2005, Cao et al. [7] proposed an image sensor mote architecture in which an FPGA connects to a VGA (640×480 pixel) CMOS imager to carry out image acquisition and compression. An

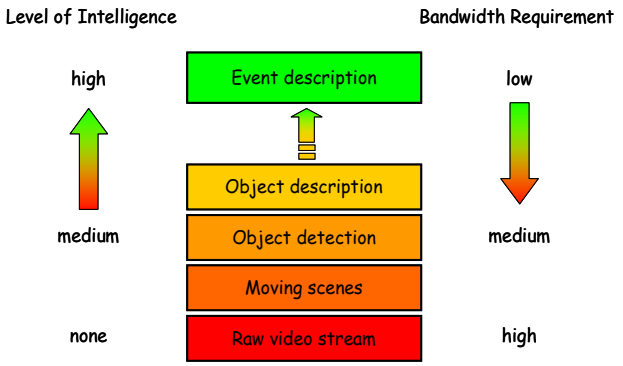

Figure 1. Intelligence-added surveillance.

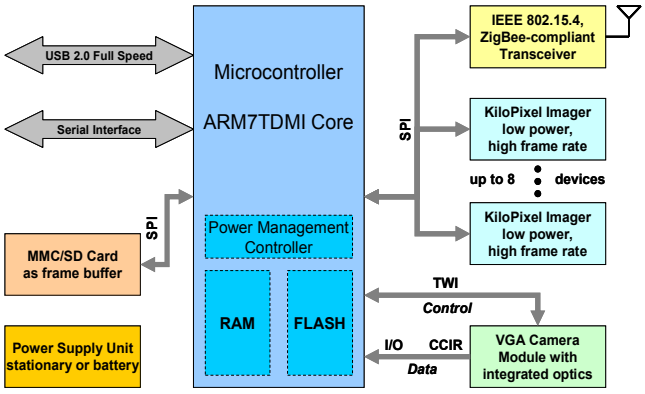

Figure 2. Block diagram of our MeshEye™ architecture.

ARM7 microcontroller processes images further and communicates to neighboring motes via an ultra-low-power transceiver at data rates up to 76.8 KBaud per second. Rahimi et al. [4] suggested another powerful image sensor mote, which combines Agilent Technologies' Cyclops with Crossbow's Mica2 mote. Cyclops was developed as an add-on CIF (320×240 pixel) CMOS camera module board, which hosts an on-board 8-bit microcontroller and 64 Kbytes of static and 512 Kbytes of flash memory for pixel-level processing and storage. The authors of [5] presented a smart camera mote architecture that uses an FPGA as its central processing unit, a VGA CMOS imager, and 10 Mbytes of static and 64 Mbytes of flash memory to perform early-vision, i.e. pixel-level, tasks. Note that all three mote architectures use FPGA or CPLD devices for interfacing with just a single image sensor. As will be discussed in this paper, we believe that this may not yield the most power-efficient nor optimal performance solution for many distributed vision-enabled applications.

A mote architecture with minimal component count was introduced by Downes et al. [6] at Stanford's Wireless Sensor Networks Lab. It deploys an ARM7 microcontroller at its core, up to 2 Mbytes of flash memory, and a 2.4 GHz IEEE 802.15.4 radio. Unlike the motes mentioned above, this mote pioneers the concept of multiple vision sensors as it can host up to four low-resolution (30×30 pixel) image sensors and two CIF CMOS camera modules. Both types of vision sensors feature a serial interface thus eliminating the need for additional FPGA or CPLD devices. In 2006, Kleihorst et al. [8] presented WiCa—a smart camera mote with a high-performance vision system. Its radio section features a 2.4 GHz IEEE 802.15.4 radio controlled by an Atmel AVR microcontroller. Its vision system consists of two VGA camera modules, which feed video to Xetal, a remarkable dedicated parallel processor based on a vector single-instruction multiple-data (SIMD) architecture. Xetal exchanges image processing results with an 8051-based host processor through a 128 Kbytes dual-port RAM.

The remainder of this paper is organized as follows. Section 2 outlines the target application and derives design objectives for the mote architecture. In Section 3, we introduce our smart camera mote architecture called MeshEye and discuss its constituent components in more detail. Section 4 discusses our implementation of the medium access control and data link layers for low-rate wireless personal area networks. Section 5 describes the configuration of MeshEye's hybrid-resolution vision system whereas Section 6 explains its underlying algorithms that perform object detection, acquisition, and tracking as building blocks for higher-level reasoning algorithms. In Section 7, we present a basic power model of our battery-powered mote architecture that yields lifetime predictions for surveillance operation under varying event rate. Finally, Section 8 summarizes our smart camera mote architecture and outlines directions for further work.

2. SURVEILLANCE APPLICATION

Our design of a smart camera mote has been pursued with a specific application in mind: distributed intelligent surveillance. This guides our design decisions and helps us in specifying critical mote functionality. We believe that surveillance will be one of the first areas to benefit from emerging wireless sensor networking technology. Especially low per-node cost, ease of deployment, scalability, and in-network distributed processing are factors that make this technology ideal for intelligent surveillance.

Intelligent surveillance may have different meaning to different people. Let us first consider how surveillance is typically realized today. Pan-tilt-zoom cameras are distributed across the deployment area and their raw video output is streamed to a surveillance center, in which a panel of monitors displays the video streams. Obviously, this implementation requires sufficient bandwidth for video streaming, has high installation cost, and most of all is hardly scalable. We consider any surveillance solution that performs processing of the video stream right at the camera and hence reduces bandwidth requirements as an intelligent system.

As a first level of intelligence, the camera nodes use a motion detection scheme such that only moving scenes are streamed to the surveillance center. At a second level, the camera nodes could perform object detection and classification such that only moving scenes containing persons or more general objects of interest are forwarded. Going even further, the smart camera nodes could collaborate to identify objects and only transmit their textual description along with a snapshot. Continuing this train of thought of adding intelligence to surveillance, a network of smart cameras could possibly just notify the surveillance center in case of events of interest by providing a hybrid textual/visual or fully textual description of the event. Figure 1 illustrates this concept of levels of intelligence in surveillance. As the level of intelligence increases, bandwidth requirements on the underlying data transmission network decrease accordingly.

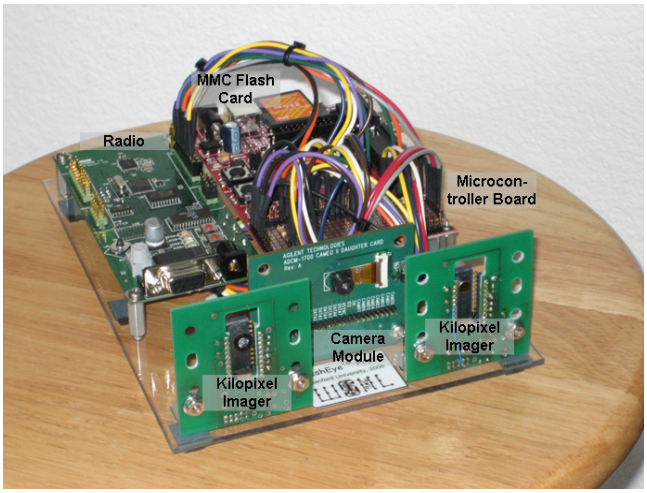

Figure 3. Photograph of a MeshEye™ prototype.

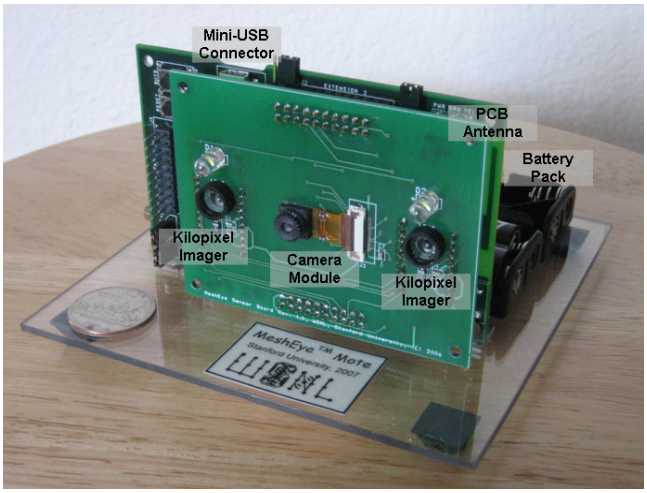

Figure 4. Photograph of the MeshEye™ mote.

A basic list of requirements for a smart camera mote architecture can be formulated from this general description of intelligent surveillance although arguably an optimized architecture would require a much more detailed specification of the entire surveillance system:

- The mote needs to monitor its observation area fast enough to detect all objects of interest. These objects vary by surveillance deployment and can include, for example, humans, vehicles, animals, airplanes or a combination thereof.
- The mote must be able to process and analyze multiple detected objects in real time. Of course, actual real-time specifications are driven by latency requirements of the application at hand.
- Wired or wireless links between motes need to exist for information exchange of the high-level reasoning algorithms. In this paper, we consider wireless links due to their rapid and scalable deployment.
- The surveillance network can either consist of homogeneous or heterogonous motes. Our approach is a homogenous mote network since we believe that it minimizes system cost and simplifies mote capability management.
- For mobile applications, mote design needs to pay special attention to energy consumption such that lifetime expectations can be satisfied using an appropriately sized battery pack. This case applies to our smart camera architecture.

3. MOTE ARCHITECTURE

The block-level architecture of our smart camera mote called MeshEye™ mote is shown in Figure 2. An Atmel AT91SAM7S family microcontroller [10] forms the core of our mote architecture. It features a USB 2.0 full-speed port and a serial interface for wired connection. The mote can host up to eight kilopixel imagers and one VGA camera module, for which we chose Agilent Technologies' ADNS-3060 high-performance optical mouse sensor [11] (30×30 pixel, 6-bit grayscale) and Agilent Technologies' ADCM-2700 landscape VGA resolution CMOS camera module [12] (640×480 pixel programmable, grayscale or 24-bit color), respectively. An MMC/SD flash memory card provides sufficient and scalable non-volatile memory for temporary frame buffering or even image archival. Wireless connection to other motes in the network can be established through a Texas Instruments CC2420 2.4 GHz IEEE 802.15.4/ZigBee-ready RF transceiver [13], which will be discussed in the following section in more detail. The mote can either be powered by a stationary power supply if available or battery-operated for mobile applications or ease of deployment.

The objectives guiding the electrical design of the MeshEye architecture have been the integration of low-power, readily available parts, use of standard interfaces, and most of all minimization of total component count. The main motivation in keeping the component count small lies in reduced power consumption and mote cost. The use of standard interfaces manifests itself in that a single SPI interface connects flash memory card, kilopixel imagers, and radio to the microcontroller. A TWI interface controls the camera module while its CCIR video is read out through general-purpose I/O pins. Note that this CCIR read-out method is not common but allows us to avoid additional interface components at the expense of a reduced frame rate of about 3 frames per second. Most other solutions interface CCIR image sensors to microcontrollers through a combination of an FPGA or CPLD and static memory for frame buffering. While such solutions may enable video streaming, it oftentimes adds significantly to the mote cost and the power budget.

Initially we built a prototype of a MeshEye mote with two kilopixel imagers. Low-level device drivers have been written and tested. Figure 3 shows a photograph of a prototype with the microcontroller development board (back, right), radio development board (back, left), two kilopixel imagers (front, left and right), and the camera module (front, center). Recently we completed the design of a fully integrated Printed Circuit Board (PCB) MeshEye mote, which is shown in Figure 4. It consists of a base board and a sensor board. The base board hosts the voltage regulators, microcontroller, radio, MMC/SD flash card and external interface connectors. Power can be supplied through an external source, the Mini-USB port, or at least one 2 AA battery

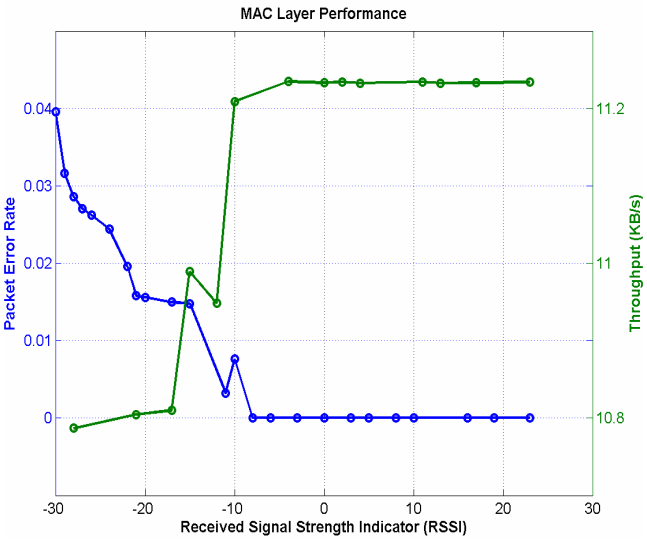

Figure 5. MAC layer performance (indoors).

set. The sensor board, which sits on top of the base board, contains the two kilopixel imagers and the VGA camera module.

3.1 Processing Unit

The mote architecture is centered around the Atmel AT91SAM7S family of microcontrollers, which incorporate an ARM7TDMI ARM Thumb processor. We prefer this microcontroller family for their leading power-efficient 32-bit RISC architecture that can be clocked up to 55 MHz. This high-performance architecture appears adequate to carry out from low- to high-level processing in real time if image resolutions and frame rates are chosen appropriately. Note that other processor architectures may also be suitable and even offer advantages in certain cases.

To meet varying memory requirements, the AT91SAM7S family offers internal SRAM up to 64 Kbytes and internal flash memory up to 256 Kbytes. Its built-in power management controller can not only put the processor into different power-saving modes, it can also power down peripherals by disabling their clock source. Additionally, an internal, programmable PLL allows the ARM7TDMI core to be operated at clock frequencies below the maximum; for example during periods of little processing load.

4. WIRELESS LINK

The Texas Instruments CC2420 transceiver provides wireless connectivity to other motes in the sensor network. It implements the IEEE 802.15.4 standard, which targets low-rate wireless personal area networks (LR-WPAN). Thus this transceiver supports data transmission at 250 Kbps (Kbits per second) with a maximum transmit power of 1mW in the unlicensed 2.4 GHz ISM band.

Obviously such a wireless link cannot support video streaming in real-time although transmission of small or highly compressed images is possible [14]. But we deem it suitable for intelligent surveillance if in-node, intermediate-level processing condenses an object's pixel array into more compact or descriptive representations (axis projection, color histogram, or object shape for instance).

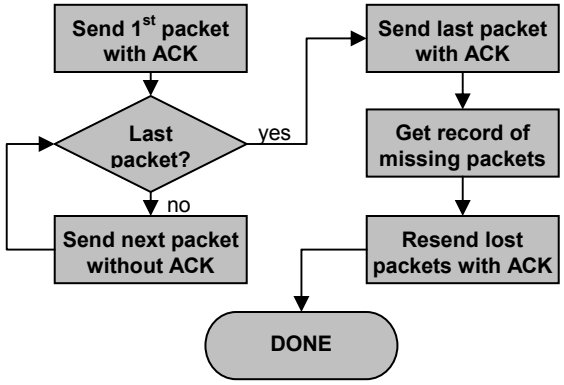

Figure 6. Implementation of efficient data link protocol.

4.1 Medium Access Control

Our current MAC layer implementation utilizes the CSMA-CA MAC protocol. Hence, before transmission the radio monitors the assigned channel and transmits data only when the signal on the channel is below a particular threshold level, which indicates that the channel is free. This minimizes collisions between packets being sent over the channel from different transmissions. The threshold level is user-defined.

For the reception of packets, the radio constantly monitors the channel traffic. When a packet is detected, the receiver places the packet in a 127-byte receive (RX) FIFO buffer. The Texas Instruments CC2420 radio also allows for hardware-based address recognition. When this function is used, which is the case for our implementation, the radio monitors the destination field of incoming packets and places the data into the RX FIFO only when the destination address of the packet matches the address assigned to the receiver. Upon reception of a packet, the receiver checks for error via cyclic redundancy checking (CRC) and checks for missing packets by comparing its expected packet sequence number with the sequence number in the packet's header.

To allow for quick response to and processing of incoming packets, an interrupt is used to notify the microcontroller when a packet is received. This ensures that the RX FIFO is cleared for further reception of packets. The interrupt occurs when a complete packet has been received or when the number of bytes received exceeds the preset threshold. Again, the threshold can be modified by the user. After the interrupt has completed, the receiver is ready to receive the next packet.

To avoid spurious packets (especially interference from nearby IEEE 802.11 radios), a simple check of the size of the packet is performed. All very small packets that cannot contain valid headers are discarded. If the sender requests an acknowledgment, the receiver automatically replies with an acknowledgment packet if it has received a packet correctly and in order. If an acknowledgment has not been received by the sender after a set timeout, the sender retransmits the packet.

When a data packet has been received correctly, the receiver extracts data from the RX FIFO to be stored internally, and then flushes its buffer to prepare for reception of the next packet.

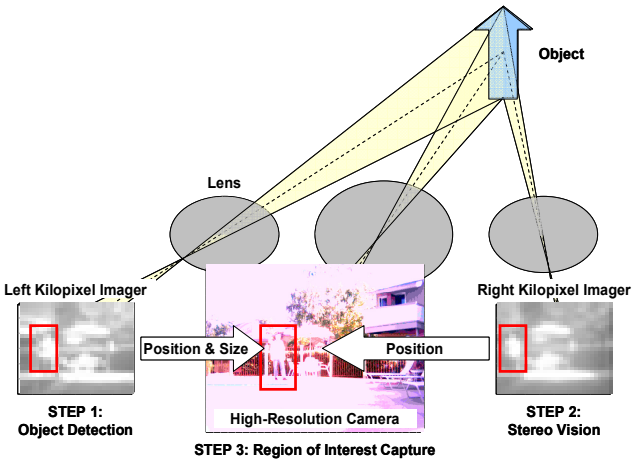

Figure 7. Hybrid-resolution vision system.

We carried out indoor measurements of our IEEE 802.15.4 wireless link to validate its correct operation. As would be expected, the observed packet error rate (PER) decreases with an increase in the received signal strength indicator (RSSI) (left-hand axis in Figure 5). The PER approaches zero for RSSI values larger than -8, and only becomes observable for RSSI values below -8. This indicates that the MAC layer provides for a highly reliable wireless link.

The MAC layer is able to sustain a maximum data throughput of about 11.2 KB/s (KBytes/second) for a RSSI of above -5 (right-hand axis in Figure 5), which falls less than 4 KB/s short of the maximum possible effective data rate inferred from Fig. 9 of [15] for direct data transmission in a nonbeacon-enabled network. Note that the throughput performance reported here applies to single point-to-point links and will significantly decrease in multi-node or multi-hop transmissions. The throughput drops to about 10.8 KB/s as the RSSI decreases to -28. This degradation of throughput is very gradual with large changes in RSSI since across the range from -28 to 25, the throughput changes only by about 3.7%.

4.2 Data Link Layer

For transmission of kilopixel images, the 900-byte image is broken up into several 110-byte long data frames. Each of these frames is sent with request for acknowledgment (ACK) from the receiver. If an ACK is not received within a certain timeout, the sender retransmits the current data frame. Once the sender receives the ACK correctly, it transmits the next frame. One limitation of this data link implementation occurs when bursts of error exist and the same frame needs to be retransmitted many times until the error burst has disappeared. This degrades the efficiency and throughput of this basic data link protocol. In addition, the delay in waiting for the acknowledgment to arrive after each data frame has been sent reduces efficiency and throughput even further.

An alternative, more efficient protocol implementation, which mitigates the above mentioned limitations, is shown in Figure 6. The sender transmits all data frames without waiting for an acknowledgment while the receiver keeps track of the correctly received frames. Only the first and last frame sent require an acknowledgment to delimit the start and end of the image

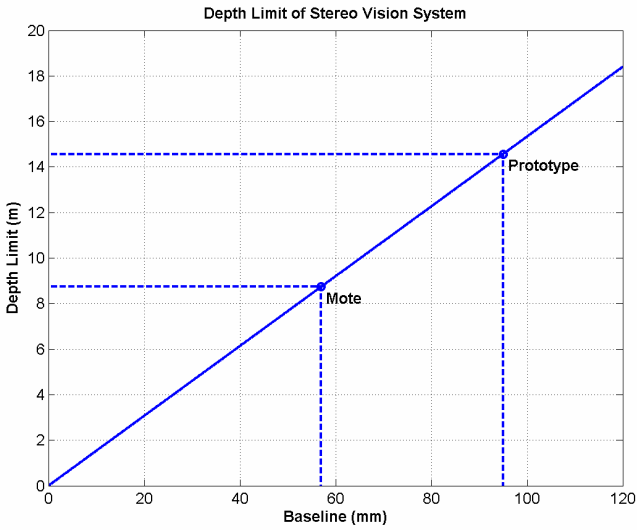

Figure 8. Depth limit of MeshEye's stereo vision system.

transmission cycle. Once the last frame has been received, the receiver returns a special control frame to the sender side with the sequence numbers of all lost data frames. In turn the sender retransmits all lost frames; this time with acknowledgment for each frame. In order to implement such a scheme, enough memory must be available to store the frames which have been sent but not yet acknowledged. Also, the receiver needs to have enough memory to store the correctly received (but out of order) frames and to store the incoming retransmitted frames (after the special control frame has been sent at the end of the normal transmission cycle) before further processing can be done on the image.

For medium to high link qualities, both data link protocols support a maximum throughput of 11 KB/s. Thus, both protocols have minimal overhead so that their throughput only falls slightly below the MAC layer's maximum throughput of 11.2 KB/s. The maximum frame rate for wireless kilopixel transmission is 10 frames per second.

5. VISION SYSTEM

The vision system forms the key sensing element of our smart camera mote. In a first implementation, we will use two kilopixel imagers with the VGA camera module centered between them. All three pixel arrays are parallel and thus facing the same direction. The three image sensors are focused to infinity and their field of view (FoV) angle should be approximately the same although ideally the kilopixel imagers should have a slightly larger FoV angle. Hence the three imagers have an overlapping FoV only offset by the small distance—the baseline—in between them.

We envision the following usage model for the three image sensors during intelligent surveillance operation. One of the kilopixel imagers will be used to continuously poll for moving objects entering its FoV. Once one or possibly more objects have been detected, position and size within a kilopixel image can be determined for each object. Basic stereo vision of the two kilopixel imagers yields the distance to the object. This information allows us to calculate the region of interest (RoI) containing the object within the VGA camera's image plane. Subsequently the microcontroller triggers the VGA camera

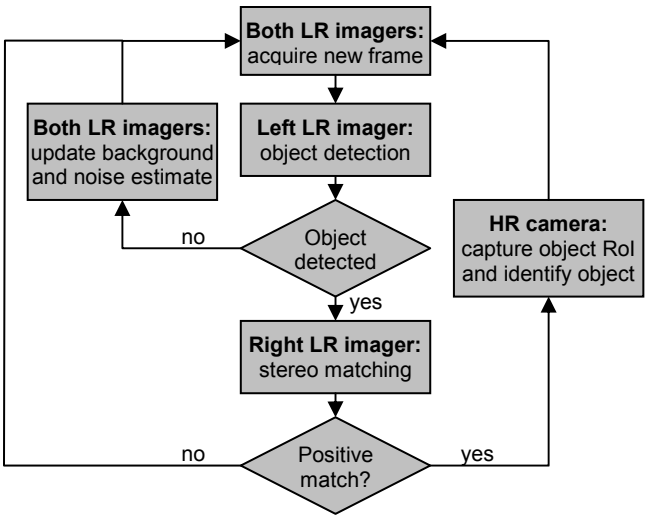

Figure 9. Flowchart of the vision processing system.

module to capture a high-resolution grayscale or color RoI including the detected object. Figure 7 illustrates this hybrid-resolution vision system overlaid with real images acquired by our mote prototype. After possibly additional low-level processing, the object's RoI will then be handed over to intermediate-level processing functions.

An obvious question frequently asked at this point is: "Why do you not just use two VGA camera modules instead and have them downsample or downsize their images to obtain kilopixel images?" While this approach is certainly feasible [8], it leads to longer image acquisition time and increased power consumption. Typically, camera modules still acquire the entire pixel array internally and downsample or downsize it digitally. Hence low-resolution frame capture with a kilopixel imager is more power-efficient even if it has about the same power consumption per pixel as the camera module.

The depth limit of our low-resolution stereo vision system can be calculated from the disparity-baseline relation

$$disparity = \frac{Bf}{Z}, \tag{1}$$

where B denotes the baseline between left and right low-resolution imager sensors, f the focal length, and Z the object's depth. To determine the depth limit of MeshEye's stereo vision system as plotted in Figure 8, we lower-bound the disparity to the 30 μm pixel pitch of the kilopixel image arrays; f equals 4.6 mm for the plastic aspheric lenses used. The 57 mm (95 mm) baseline presently set in our MeshEye mote (prototype) allows for a maximum perceived depth of 8.74 m (14.5 m) in theory, which we deem adequate for indoor and limited outdoor usage. At fixed resolution of the image sensors, one may increase the focal length to increase the depth limit at the expense of narrower FoV angle.

6. VISION ALGORITHMS

The image processing algorithms behind the hybrid-resolution vision system are designed to detect, acquire, and track objects entering its FoV. Since we plan to implement and execute these algorithms on a generic 32-bit RISC architecture without dedicated DSP engines, the algorithms are intentionally kept at lower computational complexity.

The overall vision processing flow is shown in the flowchart of Figure 9. Both low-resolution (LR) imagers continue updating their background image and estimate of temporal pixel noise when no objects are present. Upon detection of a moving object in the left kilopixel image, the vision system determines the bounding box and stores the object's RoI. This RoI serves as a template to locate the object's position within the FoV of the right kilopixel imager. If this stereo matching cannot establish a positive match to the template, the left LR imager will continue detecting the object. This case occurs for example when the object lies outside the overlapping FoV of the two kilopixel imagers. Knowing the object's position within both LR image arrays and its size, the vision system triggers the VGA camera module to acquire a high-resolution (HR) snapshot of the object. The left kilopixel imager can continue tracking the object until it leaves its FoV and initiate additional HR RoI acquisitions of the object as required by the application.

6.1 Object Detection

Prior to any further processing, the raw image arrays from both kilopixel imagers are normalized to each frame's average pixel value. This mitigates changes in brightness and exposure time. We found this normalization especially effective in coping with oscillations of the digital shutter control loop.

The kilopixel imager performs object detection through background subtraction on a frame-by-frame basis. That is it calculates the frame difference between the current frame $I^t_{left}(x,y)$, $x, y = 1,2,\ldots,30$, and the latest background $B^t_{left}(x,y)$ as given by

$$D^t_{left}(x,y) = \left| I^t_{left}(x,y) - B^t_{left}(x,y) \right|. \tag{2}$$

All pixels, whose frame difference $D^t_{left}(x,y)$ exceed a preset multiple k of the temporal noise standard deviation η, are set in a binary mask $M^t_{left}(x,y)$ as potential candidates of motion,

$$M^t_{left}(x,y) = \begin{cases} 1 & D^t_{left}(x,y) \geq k\eta \\ 0 & otherwise \end{cases}. \tag{3}$$

In our experimental vision system, η typically equals 1.75 and k is set to about 6. To eliminate objects smaller than 2×2 pixel, we low-pass filter the binary frame mask according to

$$F^t_{left}(x,y) = conv_2\left(\frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, M^t_{left}(x,y) \right), \tag{4}$$

where $conv_2$ denotes two-dimensional convolution.

In the final processing step, a blob search algorithm identifies all regions as moving objects, which consist of connected groups of unity pixels within the binary mask $M^t_{left}(x,y)$ that contain at least one unity pixel of the filtered mask $F^t_{left}(x,y)$. It is then straightforward to determine the bounding box of the object and

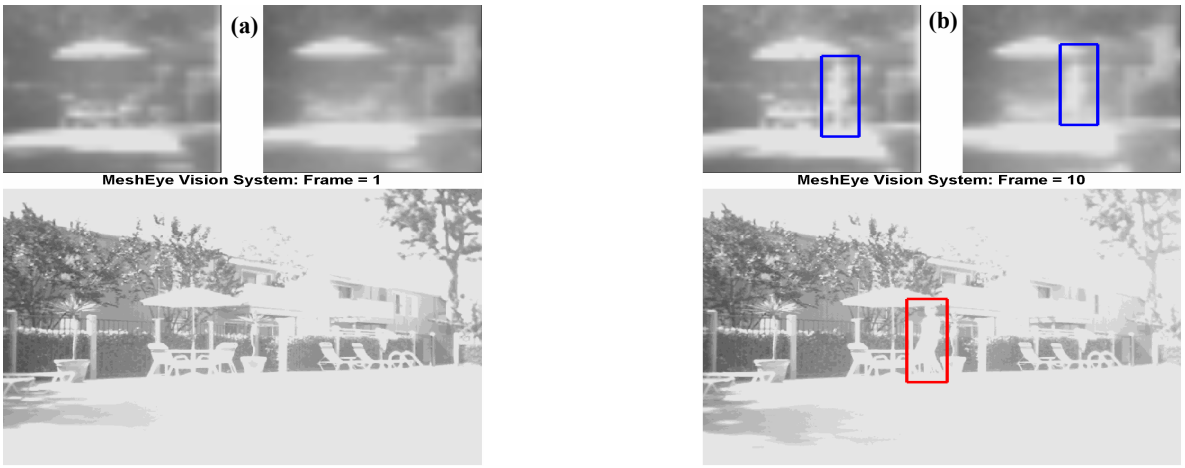

Figure 10. Hybrid-resolution vision system during operation: (a) stationary background and (b) moving person.

extract the object's difference RoI $D^t_{object}(x,y)$ as a subset of the current frame difference $D^t_{left}(x,y)$.

6.2 Stereo Matching

The objective of the stereo matching algorithm lies in locating the object's RoI within the right kilopixel image array. Since the pixel arrays of both LR imagers are aligned in parallel, the object will appear as a shifted version in the right kilopixel array if it is located within their joint FoV. This satisfies the requirements for template matching based on cross-correlation. Therefore, the vision system computes the cross-correlation of the object's array $D^t_{object}(x,y)$ over the right kilopixel difference array $D^t_{right}(x,y)$,

$$C^t_{right}(u,v) = xcor_2\left(D^t_{right}(x,y), D^t_{object}(x,y)\right), \quad (5)$$

where $xcor_2$ denotes two-dimensional unbiased cross-correlation without boundary padding such that $u,v = 1,2,\ldots,30$.

Lastly, the (u,v) coordinate with the largest cross-correlation value—or the average (u,v) coordinate in case of multiple largest cross-correlation values—is assumed as the object's position within the right LR image array. However, to qualify a positive match and remove objects outside the joint FoV, the maximum cross-correlation value has to be sufficiently close to the center autocorrelation value of the object's ROI, which may be expressed as

$$\max_{u,v} C^t_{right}(u,v) \geq 0.75 A^t_{object}(0,0). \quad (6)$$

$A^t_{object}(u,v)$ denotes the unbiased autocorrelation of the object's difference array

$$A^t_{object}(u,v) = xcor_2\left(D^t_{object}(x,y), D^t_{object}(x,y)\right).$$

If this condition is not met, no positive match can be established and the object is discarded.

6.3 Object Acquisition

The top two frames of Figure 10a show the stationary background for both kilopixel imagers in an outdoor scene. The bounding box in the top left frame of Figure 10b indicates successful detection of an object—in this case a person walking through the scene. The bounding box in the top right frame of Figure 10b confirms correct stereo matching of this person. The bounding box in the bottom image of Figure 10b results from mapping the object's bounding box from both LR imagers into the image array of the HR camera. The vision system performs this mapping by first calculating the average position of the box for the two kilopixel imagers. Recall that this averaging step is valid here since the HR camera sits right in the center of the two LR imagers. In turn, position and size of the bounding box are scaled to account for the increase in resolution of the VGA camera. For our MeshEye mote prototype, the scaling factor equals 10 in both, horizontal and vertical, directions.

Finally, the VGA camera captures a HR snapshot of the object, which is stored for further processing or exchange with neighboring smart cameras. Such processing can include object classification or even distributed, multi-camera view identification based on its shape, orientation, aspect ratio, or color histogram for example.

6.4 Computational Efficiency

To conclude this section, let us consider the computational savings that our hybrid-resolution vision system achieves over the WiCa smart camera mote [8]. WiCa's dual high-resolution (VGA format) camera system combined with the Xetal SIMD (Single Instruction Multiple Data) dedicated video processor establishes a fair base of comparison among published smart camera vision systems to carry out range estimation and moving object detection and RoI extraction.

For simplicity, this consideration of computational complexity is limited to one moving object within the vision system's FoV. The MeshEye vision system carries out the algorithms described in the previous subsections. For the dual-camera SIMD system, our calculation assumes that it performs the same computations on its two HR cameras rather than two LR imagers, which result in a moving object's pixel array and estimated range. Whenever possible the computations utilize Xetal's line-parallel processor architecture, which executes an instruction across an entire line of pixel data within one instruction cycle. Of course, an alternative approach may downsize incoming frames from VGA to kilopixel

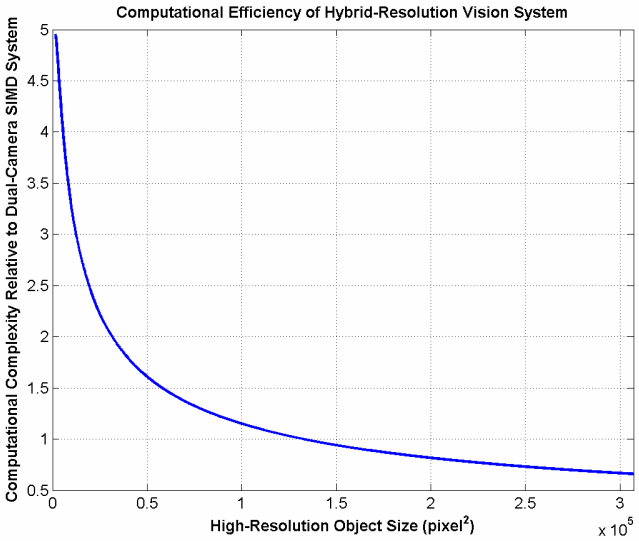

Figure 11. Computational efficiency of the hybrid-resolution vision system over the dual-camera SIMD system.

resolution prior to object detection and extraction. This however would underutilize the line-parallel processor and not take full advantage of its 640-pixel wide line buffers.

The computational efficiency of our MeshEye vision system relative to the WiCA vision system is shown in Figure 11. More specifically, it graphs the ratio of computational complexity (using Big O notation for each image processing step) of the dual-camera SIMD system over the hybrid-resolution vision system as a function of object size in high resolution when both systems perform moving object detection, ranging, and RoI extraction. The MeshEye system achieves a fivefold reduction in complexity for small object sizes, but the gains in efficiency diminish down to 0.7 as object size increases. Smaller objects require fewer computations and hence the hybrid-resolution processing outperforms the dual-camera SIMD system. Large objects cause a heavier processing load and hence the SIMD processor excels over hybrid-resolution vision.

For objects sized around 132,800 high-resolution pixel^2, which amounts to rather large objects of for example 364×364 pixel—or more than half the VGA frame, both vision systems have equal computational complexity. Note that it is for the significant reductions in computational complexity for moderately sized objects that the MeshEye architecture is viable without the need for a dedicated, high-performance DSP engine.

7. POWER MODEL

An important performance metric for battery-operated motes is their lifetime during deployment. The mote can be powered by batteries in mobile applications for instance. Batteries may also serve as backup energy sources in case the main power supply fails for example due to intruder attacks. Therefore we developed a power model for our smart camera mote architecture during basic surveillance operation.

The power model assumes that the mote is powered by two non-rechargeable AA batteries (capacity 2850 mAh) at a conversion efficiency of 90%. It accounts for current consumption of the following main mote components: Atmel AT91SAM7S64 microcontroller running at a processor clock of 47.92 MHz, PQI 256 MB MultiMediaCard flash memory, Agilent ADNS-3060 kilopixel imager, Agilent ADCM-2700 VGA camera module, Texas Instruments CC2420 IEEE 802.15.4 transceiver. Table 1 summarizes the components' typical current and runtime values used in the power model. Estimated runtimes and current draw values quoted in each component's data sheet are shown in regular font style. Italicized values have actually been measured on our mote implementation. For the most part, the initially estimated values are in good agreement with the measurements although the estimated active currents for the flash card and the image sensors turn out to be rather conservative. The power model uses measured values whenever available. Note that we have not yet entirely validated the mote lifetimes estimated by the power model although our MeshEye mote can be powered from two AA batteries.

Table 1. Estimated (regular) and *measured (italicized)* component values used in the power model.

Component	Typical Current (mA)		Runtime (ms)	
	Active	Sleep	Poll	Event[a]
Microcontroller	29.40 *29.72*	0.034	200 *125*	2000
MMC Flash Card	34.00 *14.03*	0.050 *0.116*	—	2000
Kilopixel Imager	30.6 *14.89*	0.005 *<0.010*	95 *24*	—
Camera Module	48.00 *9.65*	0.005 *0.015*	—	1000 *600*
Radio Transceiver	18.10 *19.81*	0.426 *0.419*	—	500

[a] Event assumes an average object region of interest size of 64×64 pixel.

The mote operates as follows during a basic surveillance application. In periodic poll intervals, the microcontroller wakes up, acquires a kilopixel image, and determines whether an object has entered its FoV. Once an object has been detected, the microcontroller captures a high-resolution RoI of the object following the procedure of hybrid-resolution vision described in the preceding section. For simplicity our model treats the random occurrence of object detection as an event, which occurs in periodic event intervals. Intermediate level processing reads the object's RoI from flash memory and extracts its descriptive representation. The radio transceiver communicates this representation to neighboring motes. We either measured or approximated the duration of poll and event activity for every main component as shown in Table 1. The worst-case end-to-end detection latency—an important performance metric in surveillance networks, e.g. the delay from a moving object entering the FoV until it is "known" within the network, equals the poll period plus the microcontroller's event runtime of 2 seconds.

According to this power model, Figure 12 presents a graph of mote lifetime vs. event interval for poll periods ranging from 0.5 to 3 seconds. Event processing dominates the power consumption for very frequent events and reduces lifetime considerably. When

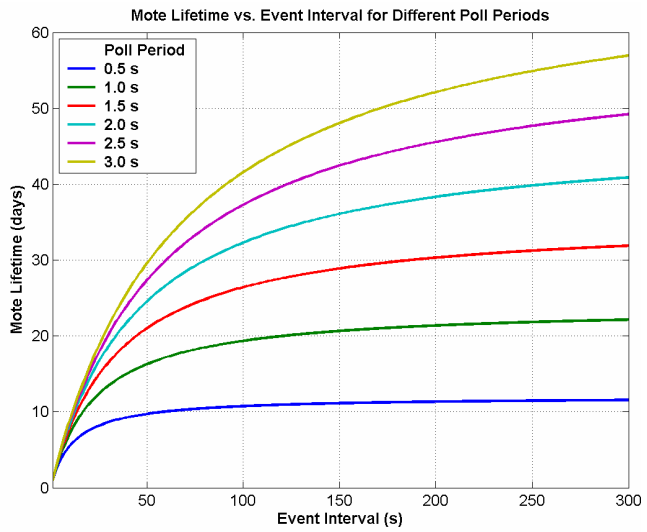

Figure 12. Mote lifetime vs. event interval.

events occur less often, poll processing governs the power consumption and the mote lifetime increases approaching lifetime limited only by poll activity. For a moderately fast poll period of 1 second, the lifetime of our smart camera mote approaches 22 days in this basic surveillance application.

As for the comparison of computational efficiency, it would be valuable to compare the lifetime of our MeshEye mote against the WiCa dual high-resolution smart camera mote [8]. However, insufficient data on power consumption of the WiCa mote has been published to date to compile a directly comparable power model.

Further reductions in power consumption of the microcontroller are attainable by deploying even more power-efficient, high-performance processor cores like Intel's StrongARM SA-1110 used in the Consus platform [16] or more recently Intel's PXA271 XScale used in the Intel Mote 2 [17].

8. CONCLUSION

In this paper, we introduced a low-power ARM7-based smart camera mote architecture. We designed it for real-time object detection and in-node processing for applications in distributed intelligent surveillance. Its hybrid-resolution vision system deploys two kilopixel imagers to trigger region of interest acquisition through a high-resolution camera module. The underlying image processing algorithms for object detection, tracking, and acquisition were discussed and illustrated on an outdoor scene. Our efficient data link layer utilizing the IEEE 802.15.4 MAC enables data exchange between smart cameras with little overhead. We presented a basic power model, which estimates battery-powered mote lifetime under varying operating conditions in a surveillance application. Future work will be directed towards continued development of mote boards, refining the vision algorithms, deploying them on our smart camera mote, and running in- and outdoor trials.

9. ACKNOWLEDGMENTS

We gratefully acknowledge Agilent/Avago Technologies for funding this research in part and supplying camera modules and optical mouse sensors. Funding support provided by Micron Foundation is gratefully acknowledged. Special thanks go to Ian Downes with Stanford University for providing the schematics of his image sensor mote design and to Benny Lai with Avago Technologies for offering advice on the boost converter circuit.

10. REFERENCES

[1] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl, S. Pankanti, S., "Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking," *IEEE Signal Processing Mag.*, vol. 22, no. 2, pp. 38–51, Mar. 2005.

[2] G. L. Foresti, C. Micheloni, L. Snidaro, P. Remagnino, T. Ellis, "Active video-based surveillance system: the low-level image and video processing techniques needed for implementation," *IEEE Signal Processing Mag.*, vol. 22, no. 2, pp. 25–37, Mar. 2005.

[3] R. Kleihorst, A. Abbo, V. Choudhary, B. Schueler, "Design Challenges for Power Consumption in Mobile Smart Cameras," *in Proc. COGnitive systems with Interactive Sensors (COGIS 2006)*, Mar. 2006.

[4] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, M. Srivastava, "Cyclops: In Situ Image Sensing and Interpretation in Wireless Sensor Networks," in *Proc. 3rd International Conference on Embedded Networked Sensor Systems (SenSys 2005)*, Nov. 2005, pp. 192–204.

[5] F. Dias Real de Oliveira, P. Chalimbaud, F. Berry, J. Serot, F. Marmoiton, "Embedded Early Vision Systems: Implementation Proposal and Hardware Architecture," in *Proc. COGnitive systems with Interactive Sensors (COGIS 2006)*, Mar. 2006.

[6] I. Downes, L. Baghaei Rad, H. Aghajan, "Development of a Mote for Wireless Image Sensor Networks," in *Proc. COGnitive systems with Interactive Sensors (COGIS 2006)*, Mar. 2006.

[7] Z.-Y. Cao, Z.-Z. Ji, M.-Z. Hu, "An image sensor node for wireless sensor networks," in *Proc. International Conference on Information Technology: Coding and Computing (ITCC 2005)*, Apr. 2005, vol. 2, pp. 740–745.

[8] R. Kleihorst, B. Schueler, A. Danilin, M. Heijligers, "Smart Camera Mote with High Performance Vision System," *ACM SenSys 2006 Workshop on Distributed Smart Cameras (DSC 2006)*, Oct. 2006.

[9] M. Bramberger, A. Doblander, A. Maier, B. Rinner, H. Schwabach, "Distributed embedded smart cameras for surveillance applications," in *IEEE Computer Mag.*, vol. 39, no. 2, pp. 68–75, Feb. 2006.

[10] Atmel Corporation, "AT91SAM7Sxxx AT91 ARM Thumb-based Microcontrollers," *Data Sheet*, Revision 6175E-ATARM-04-Apr-06, Apr. 2006.

[11] Agilent Technologies, "Agilent ADNS-3060 High-Performance Optical Mouse Sensor," *Data Sheet*, Oct. 2004.

[12] Agilent Technologies, “Agilent ADCM-2700-0000 Landscape VGA Resolution CMOS Camera Module,” *Data Sheet*, Draft 0.20, Jan. 2005.

[13] Texas Instruments, “Chipcon CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver,” *Data Sheet*, Rev. 1.4, 2006.

[14] G. Pekhteryev, Z. Sahinoglu, P. Orlik, G. Bhatti, “Image transmission over IEEE 802.15.4 and ZigBee networks,” *IEEE International Symposium on Circuits and Systems (ISCAS 2005)*, May 2005, vol. 4, pp. 3539–3542.

[15] J.-S. Lee, “An Experiment on Performance Study of IEEE 802.15.4 Wireless Networks,” *10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2005),* Sept. 2005, vol. 2, pp. 451–458.

[16] V. Raghunathan, T. Pering, R. Want, A. Nguyen, P. Jensen, “Experience With A Low Power Wireless Mobile Computing Platform,” in *Proc. International Symposium on Low Power Electronics and Design (ISLPED 2004),* Aug. 2004, pp. 363–368.

[17] R. Adler, M. Flanigan, J. Huang, R. Kling, N. Kushalnagar, L. Nachman, C. Y. Wan, M. Yarvis, “Intel mote 2: an advanced platform for demanding sensor network applications,” in *Proc. 3rd International Conference on Embedded Networked Sensor Systems (SenSys 2005)*, Nov. 2005, pp. 298–298.