Meerkats: A Power-Aware, Self-Managing Wireless Camera Network for Wide Area Monitoring

C. B. Margi[1], X. Lu[1], G. Zhang[1], G. Stanek[2], R. Manduchi[1], K. Obraczka[1]
[1]Department of Computer Engineering, University of California, Santa Cruz
[2]Volkswagen of America, Palo Alto
{cintia,xylu,gefan,manduchi,katia}@soe.ucsc.edu, ganymed@stanek.ch

Abstract

We introduce Meerkats, a wireless network of battery-operated camera nodes that can be used for monitoring and surveillance of wide areas. One distinguishing feature of Meerkats (when compared, for example, with systems like Cyclops [10]) is that our nodes are equipped with sufficient processing and storage capabilities to be able to run relatively sophisticated vision algorithms (e.g., motion estimation) locally and/or collaboratively.

In previous work [9, 8, 7] we analyzed the energy consumption characteristics of the Meerkats nodes under different duty cycles, involving different power states of the system's components. In this paper we present an analysis of the performance of the surveillance system as a function of the image acquisition rate and of the synchronization between cameras nodes. Our ultimate goal is to optimally balance the trade-off between application-specific performance requirements (e.g., event miss rate) and network lifetime (as a function of the energy consumption characteristics of each node).

1 Introduction

There are many applications of scientific, social, and strategic relevance that require monitoring of events in wide areas over long periods of time. Continuous and pervasive monitoring often necessitates a large number of networked sensors. Wiring the sensor network for power and communication is, in most cases (e.g., outdoors), too expensive and not practical, hence the need for battery-operated, wireless deployments.

There is widespread agreement within the sensor network community that multi-tier deployments, comprising both low- and high-level sensors, such as cameras, have great potential for a wide-range of current and upcoming applications. Visual sensors can cover larger fields of view and extract more substantial information about the scene than other simpler sensors such as temperature, humidity, and pressure [4, 10, 2, 11, 6]. Given that cameras are considerably more power-hungry than simpler, lower-level sensors, visual sensor systems push the envelop of energy conservation in sensor networks even further.

This paper concentrates on a specific wireless networks (*Meerkats*) developed at UCSC, consisting of battery–operated sensor nodes equipped with webcams. Each node is based on the Stargate board, and the nodes communicate using 802.11b. Even though technology advances enable more processing power and storage capability with smaller form factors, we contend that application requirements increase at the same or higher rates. Therefore, our premise is that efforts in developing low-power platforms (e.g., Cyclops [10]) and research in efficient resource management such as Meerkats are complementary to one another.

Our main challenge in Meerkats is to optimally balance the trade-off between application-specific performance requirements (e.g., event miss rate) and network lifetime. For example, a higher image acquisition rate leads to better performances but shorter lifetime, due to increased energy consumption. Different strategies for data processing and transmission also influence this trade-off. Processing an image before transmission, in order to perform event detection, isolate a region of interest, and extracting features such as motion flow, may reduce the amount of data being transmitted. But local processing is energy–consuming in itself, and the savings in transmission energy may be offset by the additional energy required for processing.

Hence, the main focus of our work is in the characterization of performance and energy consumption for a given resource management strategy. Each node operates according to a specific duty cycle, switching its components (camera, processor, radio) into different operational states (sleep, idle, active), and performing specified tasks. Each specific duty cycle requires a certain time to complete and uses a certain amount of energy. Likewise, for a certain camera placement, a specific duty cycle determines the probability that a moving object in the environment (e.g., and intruder) is missed, meaning that it transited through the camera field of view without a picture being taken of it. A thorough characterization of energy consumption and execution times for different operational duty cycles was given in [9, 8, 7]. In this work, we concentrate on performance analysis.

This paper is organized as follows. Sec. 2 describes our testbed, highlighting its hardware and software components. Sec. 3 introduces a model for performance analysis for a single node as well as for cooperating nodes. Sec. 4 has the conclusions.

2 The Meerkats Testbed

Currently, the Meerkats testbed is composed of eight visual sensor nodes, each of which consists of a Stargate, battery, webcam, and IEEE 802.11b wireless card. A laptop acts as the information sink.

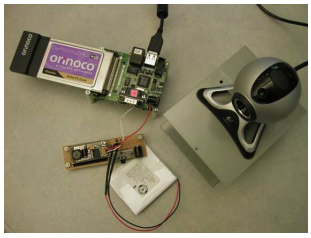


Figure 1. Visual sensing node in the Meerkats testbed

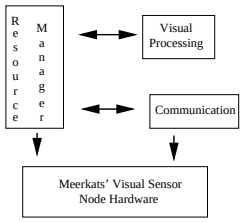


Figure 2. Meerkats software organization.

2.1 Hardware

The Meerkats node (shown in Fig. 1) is based on the Crossbow's Stargate[1] platform, which has an XScale PXA255 CPU (400 MHz) with 32MB flash memory and 64MB SDRAM. PCMCIA and Compact Flash connectors are available on the main board. The Stargate also has a daughter board with Ethernet, USB and serial connectors. We equipped each Stargate with an Orinoco Gold 802.11b PCMCIA wireless card and a Logitech QuickCam Pro 4000 webcam connected via USB. The QuickCam can capture video with resolution of up to 640x480 pixels. We use a customized 7.4 Volt, 1 Ah, 2 cell Lithium-Ion (Li-Ion) battery and an external DC-DC switching regulator (with efficiency of about 80%). The operating system is Stargate version 7.3 which is an embedded Linux system (kernel 2.4.19).

The choice of Crossbow's Stargate as the Meerkat's node main component was based on a number of considerations. First, since our focus is not on hardware design, it made sense to use off-the-shelf components. Choosing a platform that runs an open source operating system was also an important factor. And, since we selected a webcam as the visual sensor, we needed a board with a USB connector. Finally, we needed a platform that provided reasonable processing and storage capabilities.

An important feature provided by the Stargate is its battery monitoring capability. This is achieved through a specialized chip (DS2438) on the main board. Two kernel modules provide access to the battery monitor chip and retrieve information about the battery's current state.

2.2 Software

The Meerkats nodes software organization, shown in Fig. 2, consists of three main components, namely the Resource Manager, Visual Processing, and Communication modules.

2.2.1 Resource Manager

The Resource Manager is the main thread of control running on the Meerkats node. It controls the activation of the webcam and wireless network card in order to perform image acquisition/processing and communication-related tasks (e.g., transmit an image), respectively, as needed.

For energy conservation, the Resource Manager has the Meerkats sensor node operating on a duty-cycle basis, i.e., the node periodically wakes up, performs some task as needed, and goes back to either *idle* or *sleep* mode. Whereas *sleep* is the mode with the lowest power requirements, *idle* mode has a number of variations. At a minimum, the processor is awake and ready to work, even though there are no active processes running. The other variations of *idle* are: processor and wireless network card ready; processor and webcam ready; and processor, wireless network card, and webcam ready. These variations correspond to the cases where the node is ready to engage in communication-related tasks, image acquisition/processing tasks, or both. An accurate power consumption analysis for the different elementary tasks forming a duty cycle, along with a number of different duty cycle configurations and related energy measurements, was presented in [8].

2.2.2 Visual Processing

The Visual Processing module performs all vision-related tasks, including image acquisition, compression, and processing. It is invoked by the Resource Manager after the webcam has been activated. The goal is to detect events, in the form of moving image. Upon completion, Visual Processing returns control to the Resource Manager with a parameter flagging whether an event has been detected, as well as a set of parameters including the number of moving blobs in the image and the velocity of each blob. If an event is detected, the relevant portion of the image is JPEG compressed and transmitted to the sink.

Moving blobs in the image are detected using a fast motion analysis algorithm described in [5]. The algorithm is comprised of three stages. First, local differential measurements are used to determine an initial labeling of image blocks, using a total least squares approach with fast implementation. Then, belief propagation is used to impose spatial coherence and resolve aperture effect inherent in textureless areas. Finally, the velocity of the resulting blobs is estimated via least squares regression. On the Meerkats node, the motion analysis algorithm, applied on a pair of consecutive images, takes about 0.9 s and consumes 0.16 C (Coulomb)[2].

2.2.3 Communication

Communication between nodes and the sink is based on 802.11b links. Multi–hop routing is performed using the Dynamic Source Routing (DSR) protocol [3]. This is an on-demand routing mechanism especially designed for multi-hop wireless ad-hoc networks. The version of DSR running on the Meerkats nodes was ported from the DSR kernel module available for the PocketPC Linux [12].

Two types of data are handled by the application layer in our current implementation: control packets (exchanged between nodes via UDP for synchronization and alerting); and image data (transmitted from nodes to the sink via TCP). The sink runs a multithreaded server program that listens for

[1] `www.xbow.com`

[2] These measurements were obtained using an HP 34001A digital multimeter connected to the board.

connection requests from sensor nodes, opens a connection, receives image files and renders images on the sink's console.

In our experiments, we observed sporadic instability problems using the 802.11b links. In order to minimize the effect of this instability, we implemented a simple fault recovery procedure. When control packets are transfered via UDP, the receiver is required to send an acknowledge message (ACK) back to sender. If within in a fixed period of time the sender doesn't receive an ACK from the receiver, the sender re-send the same control message during the next duty cycle. In the case of image data being sent from a camera node to the sink via TCP, a timer is set up at the sender to monitor the establishment of a TCP connection. If the TCP connection is not built within a fixed period of time, the sender considers that transmission failed, and tries to set up a TCP connection again in the next duty cycle. With these simple procedures, the reliability of data transmission over 802.11b is at an acceptable level for our experiments.

2.2.4 Master–Slave Coordination

Two nodes may coordinate when tracking a moving object in the scene. In our experiments we considered a simple master–slave scenario. The master node acquires and processes images on a regular basis. If it detects an event, it sends a short alert packet to the slave node. Unfortunately, Meerkats node are not interruptible while in sleep more, and therefore the slave node needs to periodically wake up and listen for messages from the master; if it receives an alert packet, it takes and compresses an image.

Fig. 3 shows the synchronization strategy used in our test. The master node acquires images at times kT, where T is the cycle period. If an event is detected at the n–th cycle, an alert message is sent at time $nT + T_2$, where T_2 is a known constant. The slaves listens for messages during a temporal window starting at times $kT + T_2$. The size of this window depends on the expected delay, as well as on the uncertainty of clock synchronization between the two nodes.

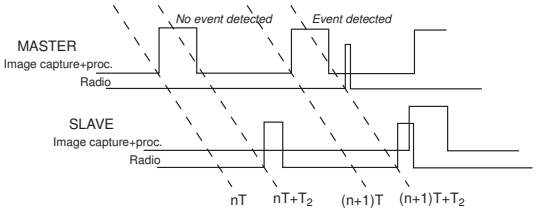


Figure 3. The synchronization scheme for master/slave organization.

Fig. 4 shows the current drawn by the master and slave nodes over two cycles, without and with an event detected. Note that in this particular experiment, the slave started in *idle* mode (rather than in sleep mode), but then is put to *sleep* at the end of each subsequent cycle.

3 Performance Analysis

The goal of the network is detect and track moving bodies within the area covered. Ideally, every time a body enters the field of view (FOV) of a camera, the camera would take one or more images of it. The visual data is used for event detection, data transmission in the chosen representation, and

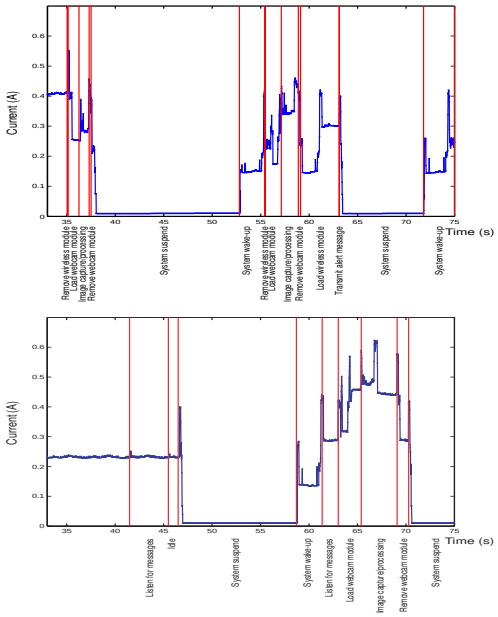


Figure 4. The time profile of current drawn for the master (top) and slave (bottom) measured over a cycle with no event detected and a cycle with an event detected.

activation of nearby nodes which are likely to see the body next. However, due to the finite acquisition rate of the cameras, it is possible that a moving body traverses a camera's FOV without being detected, and it is therefore important to assess the probability of this occurrence.

In our notation, the presence of a moving body in the network is denoted by the event X^1. If the body enters the i–th camera FOV (FOV_i), we will say that the event F_i^1 occurred. Every time a body circulating in the area covered by the network enters the i–th camera's FOV and is not detected, we will say that a "miss" event for camera i occurred, denoted by M_i^1. More in general, one may consider the case of n bodies circulating in the network (event X^n), r of which enter the FOV_i at some point (event F_i^r), with the i–th camera missing k of the bodies in its FOV (event M_i^k). We can safely assume that M_i^k is independent of X^n given F_i^r (since objects outside the camera's FOV cannot be detected anyway), so that $P(M_i^k|F_i^r, X^n) = P(M_i^k|F_i^r)$. We will further assume that $P(M_i^k|F_i^r)$ is binomial, meaning that each "miss" event is independent from the others. This makes sense in the case of "rare events", that is, when two bodies are unlikely to appear at the same time in the same FOV. We will also postulate that $P(F_i^r|X^n)$ is binomial, a reasonable assumption in the case of independently moving bodies.

A possible measure of performance of a camera node is the ratio of the expected numbers of "miss" events to the expected number of bodies in the network ("miss rate"): $\text{MR}_i = \text{E}[M_i]/\text{E}[X]$, where the $\text{E}[\cdot]$ represents the expectation operator. Let $\text{P}_{M|F} = \text{P}(M_i^1|F_i^1)$ and $\text{P}_{F|X} = \text{P}(F_i^1|X^1)$. Using the total probability theorem, and remembering that the conditional distributions of interest are binomial, it is not difficult to show that $\text{MR}_i = \text{P}_{M|F}\text{P}_{F|X}$.

In the next two subsections we will show how, in some

practical situations, the two factors $P_{M|F}$ and $P_{F|X}$ can be estimated for *end nodes* (nodes that are at the edge of the network, or near points of high flux, such as near an entry door) and *internal nodes*, which are normally in the neighborhood of one or more end nodes.

3.1 End Nodes

Consider the case of Fig. 5(a), with a camera placed near a door, or an area of relatively high flow. For simplicity's sake, we represent a FOV as a triangle, which approximates the trace of the actual FOV assuming that the camera is not too high on the ground. If the cameras are high (e.g. on the ceiling) pointing down, then the FOV traces will take different shapes.

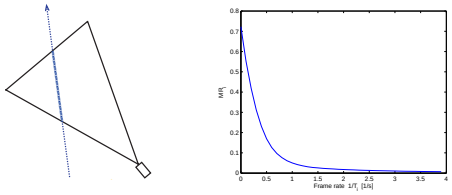
We will assume that persons walk through the door at times that are modeled by a Poisson point process of unknown density λ. We further assume that persons walk through the door in a rectilinear motion, with constant but unknown velocity v and orientation ϕ that can modeled by suitable probability distributions $p_v(v)$ and $p_\phi(\phi)$. For example, in our simulations we model v as a truncated Gaussian random variable, and ϕ as a uniform random variable. Note that prior information on the velocity is often available (e.g. the average speed of walking). We further assume that the orientation and the velocity of motion are statistically independent. As shown in Fig. 5, the direction of motion ϕ determines the length $l_1(\phi)$ of the path from the door to FOV_i, and the length $l_2(\phi)$ of the path overlapping FOV_i. Together with the velocity v, these path lengths determine the amount of time $t_1(\phi,v) = l_1(\phi)/v$ that it takes to go from the door to FOV_i, and the amount of time $t_2(\phi,v) = l_2(\phi)/v$ the moving person will be within FOV_i.

Let Φ be the set of orientations that overlap FOV_i. Then, $P_{F|X} = \int_{\phi\in\Phi} p_\phi(\phi)\, d\phi$. As for $P_{M|F}$, the probability of misdetection given that the person walks in the camera's FOV, it depends on the image acquisition policy of the camera. Under periodic image acquisition (with period T_i), a person walking through FOV_i is not detected if, for some m, $mT_i < t_{in} < t_{out} < (m+1)T_i$, where $t_{in} = t_0 + t_1(\phi,v)$ is the time at which the person enters FOV_i, $t_{out} = t_0 + t_1(\phi,v) + t_2(\phi,v)$ is the time at which the person exits FOV_i, and t_0 is the time at which the person walks through the door. Since t_0 is, by hypothesis, an outcome of a Poisson process, it is not difficult to see that the condition above is verified with probability $1 - \min(t_2(\phi,v), T_i)/T_i$. Hence:

$$P_{M|F} = \int_{\phi\in\Phi}\int_v p_\phi(\phi)p_v(v)\left(1 - \frac{\min(t_2(\phi,v),T_i)}{T_i}\right) dv\, d\phi \tag{1}$$

Fig. 5(b) shows the relationship between the image acquisition period T_i and the miss rate MR_i for the situation in Fig. 5(a). This information (possibly contained in a look–up table) can be used by the Resource Manager to decide a suitable image acquisition rate for the camera.

In practice, the parameters needed to estimate the miss rate are known only with a certain degree of approximation. These parameters include the location and orientation of the camera, as well as the statistical distribution of orientation and velocity. The hypothesis of rectilinear motion at constant speed may not always be accurate. However, uncertainty about the camera geometry can be taken into account by suitable modification of the expressions for $P_{F|X}$ and $P_{M|F}$. Likewise, uncertainty about the actual distributions of v and ϕ can be modeled by increasing the variance of the model distributions[3].



(a) (b)

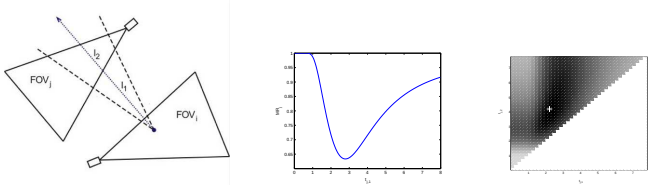
Figure 5. (a): A possible layout of an end node. The triangular shape represents the node's FOV, while the angular sector represents the possible directions of motion. (b): The miss rate as a function of the image acquisition rate $(1/T_i)$.

3.2 Internal Nodes

An internal camera node is normally alerted about the possible arrival of a moving body by one or more other end or internal nodes. Of course, in addition to this reactive behavior, an internal node may also follow a policy of regularly timed image acquisitions, to account for moving bodies that may have been missed by other nearby nodes.

An event detection by the i–th node at time t_0 is accompanied by some geometric information about the moving body. At a minimum, it is known that a moving body was located within FOV_i at time t. Different levels of geometric information may be extracted by visual analysis, including: the orientation of motion with respect to the camera axis; the actual position of the body in the ground plane; the direction of motion; the velocity of the body. Given the geometric information available (together with its uncertainty), and the location and orientation of nearby cameras (which may also be known to a degree of uncertainty), the control algorithm (distributed or localized at the i–th node) needs to decide: (1) Which (if any) nearby cameras need to be alerted; (2) How many images each of such cameras should take; (3) What are the optimal times for the image acquisition. Intuitively, if a very reliable prediction of the body's motion could be made, only the camera whose field of view will be intersected next by the body's path should be alerted, and just one image should be taken at any time the body is within this field of view. Due to uncertainty in the knowledge of the camera and moving body geometry, this prediction will be only approximate, meaning that more than one cameras may need to be alerted, and more than one image may have to be taken. Our strategy is to compute, for each nearby camera of index j, the miss rate MR_j as a function of the number of images N_j it may take, and of the times $\mathbf{t}_j = \{t_{j,1},\ldots,t_{j,N_j}\}$ at which the images are taken. For each value of N_j, the times $\mathbf{t}_j$ that minimize the corresponding miss rate can be

[3]Note that $p_\phi(\phi)$ and $p_v(v)$ can, in principle, be learned by analyzing the data collected by the camera.



(a) (b) (c)

Figure 6. (a): A possible layout of an internal node. An event has been detected at time t_0 by another sensor, which estimated that a body is moving within the specified angular sector. (b): The miss rate as a function of the time $t_{j,1}$ at which a single image is taken by the camera. (c): The miss rate as a function of the times at which two images $t_{j,1} < t_{j,2}$ are taken by the camera (the cross represent the pair of time instants that minimize the miss rate).

computed, resulting in the optimal (decreasing) sequence of values $\mathrm{MR}_j(N_j)$. Based on this knowledge, the control algorithm can allocate the number of images to be taken by each camera, balancing the need for a low miss rate with the available energy at each node.

A simple example of computation of the sequence $\mathrm{MR}_j(N_j)$ in the case of one nearby node detecting an event is shown in Fig. 6. For simplicity, we assume that the location of the body at time t_0 is known exactly, that the body is moving of rectilinear motion at constant speed, and that the distribution of velocities, $p_v(v)$ and of directions, $p_\phi(\phi)$ are modeled in the same way as in Sec. 3.1. Again, note that any level of uncertainty (about the location in which the body was seen, the motion direction and velocity) can be injected in our model by suitably modifying the probability distribution involved.

Fig. 6(b) shows the plot of $\mathrm{MR}_j(1)$ as a function of the image acquisition time $t_{j,1}$ for the case of Fig. 6(a). In this case, $\mathrm{MR}_j(1) = 0.63$. (For details about this computation, please see [1].) In order to reduce the miss rate, the node might take two snapshots (at an increased energy cost). The problem is then to compute the optimal times $t_{j,1} < t_{j,2}$. Fig. 6(c) shows the miss rate as a function of $(t_{j,1}, t_{j,2})$, with the optimal pair represented by a cross. The corresponding miss rate, $\mathrm{MR}_j(2)$, is equal to 0.43; as expected, it is smaller than $\mathrm{MR}_j(1)$.

This simple example shows how, at least in principle, it is possible to optimize the acquisition time when coordinating tracking of a moving body. However, in the practical implementation of this scheme, a number of system–related constraints must be taken into account. For example, as discussed in Sec. 2.2.4, a slave Meerkats node can receive control and alert packets from a master node only in specific time windows. The interval between two consecutive reception windows indirectly affects both the miss rate (the optimal image acquisition time may be missed due to a long interval) and the node life time (a short interval limits the amount of time in which a slave node can be kept in sleep mode).

4 Conclusions

This paper introduced Meerkats, a wireless network of battery-operated camera nodes that can be used for monitoring and surveillance of wide areas. Our work focuses on the analysis of the trade-offs between performance and network lifetime. In this paper, we concentrated on simple models that relate the miss rate with the image acquisition rate of the camera nodes, as well as on the synchronization between cooperating nodes. Our next step will be to integrate these models into the Resource Management module, whose purpose is to select suitable duty cycles for the different nodes in order to ensure the required miss rate while maximizing the network lifetime.

5 References

[1] J. Boice, X. Lu, C. B. Margi, G. Stanek, G. Zhang, R. Manduchi, and K. Obraczka. Meerkats: A Power-Aware, Self-Managing Wireless Camera Network for Wide Area Monitoring. Technical Report ucsc-crl-05-04, University of California Santa Cruz. `www.soe.ucsc.edu/research/reports/UCSC-CRL-05-04.pdf`.

[2] W.-C. Feng, B. Code, E. Kaiser, M. Shea, W.-C. Feng, and L. Bavoil. Panoptes: Scalable low-power video sensor networking technologies. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 562–571, New York, NY, USA, 2003. ACM Press.

[3] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[4] P. Kulkarni, D. Ganesan, and P. Shenoy. The case for multi-tier camera sensor networks. In *International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2005)*, 2005.

[5] X. Lu and R. Manduchi. Fast image motion computation on an embedded computer. In *2nd IEEE Workshop on Embedded Computer Vision*, New York, June 2006.

[6] C. M., J. Reich, and F. Zhao. Distributed attention in large scale video sensor networks. In *Proc. IEE Intelligent Distributed Surveillance Systems*, 2004.

[7] C. Margi. *Energy consumption trade-offs in power constrained networks*. PhD thesis, University of California, Santa Cruz, 2006.

[8] C. B. Margi, R. Manduchi, and K. Obraczka. Energy consumption tradeoffs in visual sensor networks. In *24th Brazilian Symposium on Computer Networks (SBRC 2006)*, Curitiba, Brazil, June 2006.

[9] C. B. Margi, V. Petkov, K. Obraczka, and R. Manduchi. Characterizing energy consumption in a visual sensor network testbed. In *2nd International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2006)*, Barcelona, Spain, March 2006.

[10] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava. Cyclops: In situ image sensing and interpretation in wireless sensor networks. In *SenSys 2005)*, 2005.

[11] A. Rajgarhia and F. Stann. Privacy sensitive monitoring with a mix of IR sensors and cameras. Technical Report ISI-TR-2003-582.

[12] A. Song. Piconet ii - a wireless ad hoc network for mobile handheld devices. `http://piconet.sourceforge.net/`, 2001.

Acknowledgments

This work was supported by NASA, Intelligent Systems Program, under contract NNA04CK89A.