

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

J. Boice, X. Lu, C. Margi, G. Stanek, G. Zhang, R. Manduchi, K. Obraczka
Department of Computer Engineering, University of California, Santa Cruz

Technical Report ucsc-crl-05-04

ABSTRACT

Meerkats is a wireless network of battery-operated camera nodes, that can be used for monitoring and surveillance of wide areas. This paper describes the Meerkats architecture, including: (1) a number of application-level visual sensor acquisition and processing techniques such as image acquisition policies (including cooperative, event-driven policies); visual analysis for event detection, parameter estimation, and hierarchical representation; (2) resource management strategies that dynamically assess the power versus application-level requirements to make decisions on the tasks to be performed by the system (e.g., what data representation level to use in transmitting data at a given point in time); and (3) network-level techniques for bandwidth- and power-adaptive routing as well as media scaling.

We also report results from the statistical models we developed for image acquisition scheduling which, given a probability of mis-detection, try to minimize power consumption by deciding when cameras should wake up and acquire images. We present results from our power consumption characterization of the Meerkats sensor node which is based on the Crossbow Stargate [13]. Finally, we describe our visual processing approach based on motion analysis to perform event detection, motion orientation, ground plane positioning, and velocity determination.

I. INTRODUCTION

Most wireless sensor networks to date employ nodes with limited power, processing, storage, and communication capabilities. Additionally, they typically include relatively simple sensors (e.g., light, temperature, pressure, magnetometer, etc.). In these deployments, energy consumed by sensing-related tasks is relatively small, which means that the communication subsystem (i.e., the radio) dominates energy consumption. On the other hand, in scenarios that employ more sophisticated sensors, usually power and bandwidth are not a concern. For instance, in most “traditional” camera-based surveillance

systems, nodes are interconnected through a wired network and plugged to continuous power sources. Another example of a sensor network utilizing powerful sensors is the Center for Collaborative Adaptive Sensing of the Atmosphere’s (CASA) [7] network connecting radars for atmospheric monitoring and prediction.

We introduce Meerkats, a wireless network of battery-operated camera-equipped nodes that can be used for monitoring and surveillance of arbitrarily large (indoor or outdoor) areas. In contrast to networks of simple sensors, power consumption due to sensing and processing is no longer negligible and is comparable to power consumed by other node components, in particular the communication subsystem. Therefore, new energy and information management strategies are needed to maximize sensor network lifetime, while guaranteeing adequate performance. In monitoring applications, performance essentially translates into the likelihood of detecting relevant events. Hence, in Meerkats, the driving goal is to balance the trade-off between maximizing the lifetime of the system while keeping event detection probability high. In the sensor networking research community, different definitions of system lifetime have been used. In dense sensor networks, for example, sensor network lifetime may refer to the time until the network gets disconnected and is no longer able to provide adequate coverage of the area being monitored. In sparse deployments, lifetime may relate to how soon any one node runs out of battery. This is reasonable since in sparse networks having a sensor fail usually means that coverage (and possibly connectivity) gets severely degraded.

The energy and information management strategies we develop in Meerkats address the (frequently conflicting) requirements of the different system components (illustrated in Figure 1). For example, for the visual processing component, which aims at detecting and characterizing events within a node camera’s field of view, the more images acquired the more accurate its decision of whether an event worth reporting has occurred. However, the more images, the more power consumed. Another important trade-off addressed in Meerkats is whether it

is more efficient to perform on-board processing and thus save bandwidth by transmitting higher-level representation of the data (e.g., whether an event occurred or not), or simply send the full video stream. To make such decisions, information about the power budget of the nodes involved, the power consumed by the different operations involved (e.g., data acquisition, transmission, processing, etc.), as well as the application performance requirements (e.g., maximum acceptable mis-detection rate) is required.

Meerkats' main contributions include: (1) application-level visual sensor acquisition and processing techniques such as image acquisition policies (including cooperative, event-driven policies); visual analysis for event detection, parameter estimation, and hierarchical representation; (2) resource management strategies that dynamically assess the power versus application-level requirements to make decisions on the tasks to be performed by the system (e.g., what data representation level to use in transmitting data at a given point in time); and (3) network-level techniques for bandwidth- and power-adaptive routing as well as media scaling.

Recently, a few camera-based sensor networks have been developed. One notable example is the Panoptes project [10]. Even though Panoptes target similar sensor network applications, it has different goals than those of Meerkats. It addresses scarce and/or intermittent communication (e.g., when the wireless card is turned off to save power) by buffering data at nodes and/or selecting data to be discarded when buffers fill up. It does not consider data acquisition policies, visual processing techniques, or data representation level adaptation as a way to save power. Further, it does not employ network-level techniques for QoS or power awareness. Another example of a camera-based sensor network is reported in [36]. Their focus is on storing data (reducing network communication) to achieve energy efficiency, developing efficient querying techniques for data retrieval, and using a reliable transport protocol to reliably transfer data. PARC's video sensor network prototype [11] also targets surveillance applications. But, unlike Meerkats, it addresses mainly the issue related to collaborative sensor processing to reduce the amount of information collected/disseminated to support effective perimeter surveillance.

In this paper, we describe the Meerkats architecture and its components as illustrated in Figure 1. We also report results from the statistical models we developed for image acquisition scheduling which, given a probability of mis-detection, try to minimize power consumption by deciding when cameras should wake up. Additionally, we present results from our power consumption charac-

terization of the Meerkats sensor node which is based on the Crossbow Stargate [13]. Finally, we describe our visual processing approach based on motion analysis to perform event detection, motion orientation, ground plane positioning, and velocity determination.

II. SYSTEM ARCHITECTURE

In this section, we present an overview of Meerkats' architecture. We also describe our current testbed, including its power consumption characterization, strategies for radio and camera activation, as well as specific sensor placement issues.

The Meerkats system architecture is illustrated in Figure 1 which depicts the logical organization of Meerkats' components as well as their interaction. Note that every Meerkats node runs all of Meerkats' components.

The Visual Processing (VP) module, which is described in detail in Section IV, is responsible for detecting and characterizing events within a camera's field of view (FOV). It also performs collaborative sensor processing where information from multiple cameras is fused for more accurate event detection.

Using information from the VP module (which may contain the currently estimated location, direction and velocity of motion of a tracked target), the Image Acquisition Scheduling (IAS) component determines the miss rate as a function of the allocated number of snapshots, together with the optimal snapshot times. Of course, if cameras were always on, the probability of missing an event would be zero. IAS' main function is to schedule camera image acquisition to maximize the probability of event detection while being energy efficient to prolong system lifetime. The IAS component is described in detail in Section III.

In order to transmit images efficiently and timely from sensor nodes to information sinks, Meerkats includes a QoS Routing module (described in Section V), which performs discovery and maintenance of routes that are able to meet the quality-of-service (QoS) requirements of the application in a power-efficient way. To accomplish its task, QoS Routing needs information on the network's current conditions (e.g., bandwidth, delay), as well as remaining power in the nodes. This information is provided by the Network Probing module, which is presented in Section V-B.

Finally, the Resource Manager (RM) receives information from all modules and makes system-wide information management decisions. For instance, it decides when to activate a camera based on information about actual- versus required miss rate and available power. More specifically, in order to determine the optimal times at which to wake up the camera and take snapshots,

the RM considers input from nearby nodes. The RM integrates information from the IAS module (e.g., miss rate as a function of snapshots, optimal snapshot times, etc.) with any other available information to task camera nodes that are most likely to see the target next. The RM also accounts for power consumed by the various tasks involved (e.g., processing sensing, communication). This information is available from the power characterization benchmarking we conducted for the Stargates (see Section II-B). The RM also decides which data representation level should be employed for visual data transmission given the available bandwidth and power at a given node. Additionally, the RM advises QoS routing on which metric to use when selecting routes.

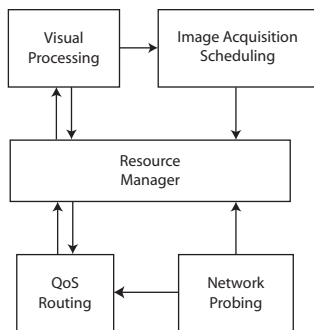


Fig. 1. Meerkats system architecture

A. Testbed

Currently, our testbed consists of eight visual sensor nodes and one gateway (also used as the information sink). Visual sensor nodes use Crossbow’s Stargates [13], while for the gateway we use a laptop.

The Crossbow Stargate is based on the XScale PXA255 CPU (400 MHz), has 32MB flash memory and 64MB SDRAM, and provides PCMCIA and Compact Flash connectors on the main board. It 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 through the USB. The QuickCam can capture video with resolution of up to 640x480 pixels.

The Stargate can be powered through a 5V DC adaptor or through a battery. Both the main- and daughter boards have battery input, but only the daughter board has a DC input. Since we are using the USB connector on the daughter board, we need to power the Stargate through the daughter board with 5V. To achieve this, we use a customized 7.4 Volt, 1000mAh, 2 cell Lithium-Ion (Li-Ion) battery, manufactured by Energy Sales, Inc. The operating system is Stargate version 7.3 which is an embedded Linux system (kernel 2.4.19).

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, namely *onewire* and *batmon*, provide access to the battery monitor chip and retrieve information on the battery’s current state. The original implementation of *batmon* only accesses voltage information; we extended it to also read instantaneous- and accumulated current. *batmon* maps the data available through the chip to `/dev/platx/batmon`. Then, to read battery information at a given time, one can do `'cat /dev/platx/batmon'`. Therefore, if we set the current accumulator (I_a) to the battery capacity (C) once a fully charged battery is connected, we can determine the battery lifetime by doing $L = (C - I_a)/C$. Note that the current flowing out of the battery will be decreased from the initial value of I_a .

Although Stargate’s battery monitoring capability is a nice feature, it is only available for a battery connected to the main board. Since we are using the daughter board to power the system, we need to modify the Stargate circuitry in order to monitor power consumption of the whole system. This is an item of future work.

The laptop we are using is a Dell Inspiron 4000 with PIII CPU, 512M memory, and 20G hard disk. It runs Linux (kernel 2.4.20) and uses an Orinoco Gold 802.11b wireless card for communication.

It is noteworthy that the Panoptes sensor platform [10] is based on the StrongARM processor, has 64MB of memory, a Logitech 3000 USB webcam, and a 802.11 wireless card. For Meerkats, we chose to use the Stargates, an off-the-shelf, low power platform. For example, while the power required by the Panoptes Sensor node for power intensive tasks is about 4.3W, the Stargate consumption for similar tasks is under 4W.

B. Power Consumption Characterization

To measure the power consumed by the Crossbow Stargate platform [13], we develop an energy consumption characterization benchmark, consisting of a set of basic operations that are representative of tasks performed by visual sensor nodes.

Our benchmark consists of five main task categories, namely *idle*, *processing intensive*, *storage intensive*, *communication intensive* and *visual sensing*.

a) *Idle*: The idle state or baseline benchmark captures the energy consumption behavior of the node when only basic operating system tasks are running. This benchmark characterizes energy consumption when the system is idle and also serves as a reference for all other tasks.

b) Processing-intensive: To characterize processing-intensive tasks, we use the FFT benchmark [1], which is part of SPEC’s CPU2000 [39], an industry-standardized CPU-intensive benchmark suite. FFT, short for fast Fourier transform, is an efficient algorithm to compute the discrete Fourier transform (DFT) and its inverse.

c) Storage-intensive: The storage media available on the Stargates is flash memory. In order to understand its energy consumption, we use a program that does file reads and writes. The program that writes a file uses random data to write a file with size provided as input parameter. The program that reads a file will take the file name as input parameter.

d) Communication-intensive: To characterize the energy consumed by communication-related tasks, we use a set of client/server programs. The client program transmits a certain amount of random bytes (provided as a argument) to the server. To obtain the energy cost of transmission, we run the client program on the Stargate being monitored. Then we monitor the Stargate running the sever program to obtain the energy cost of reception.

e) Visual sensing: To characterize power consumption due to the webcam, we use the *videotime* program available on the Stargate 7.3, to acquire a sequence of frames.

For our application, we consider that the Crossbow Stargate has three major components:

- **Processor core:** consists of the processor itself, memory (RAM and flash), and associated hardware;
- **Sensor core:** consists of the sensing devices, which in the Meerkats nodes is the webcam;
- **Communication core:** also called radio, consists of the wireless communication circuitry and antenna.

These different components can be in different states. For instance, the processor can be sleeping, idle, active (processing), writing data or reading data; the sensor can be sleeping, idle or active (acquiring image/video); while the radio can be sleeping, idle, receiving or transmitting. In order to characterize the different energy consumption levels, we need to cover all reasonable state combinations and possible tasks being executed. For example, in the Stargate, it does not make sense to have the processor sleeping and radio and/or sensors idle or active, since they need to be controlled by the processor.

In order to put the processor in sleep mode, one must execute the utility *sys_suspend*, giving as a parameter the time the processor should be sleeping. To put the wireless card in sleep mode, the utility to be used is *cardctl suspend*, while *cardctl resume* changes the wireless card from sleep to idle. The mechanism we use to put the webcam in sleep mode is to remove the

corresponding modules from the kernel (*rmmmod usb-ohci-sa1111*), while to change the webcam from sleep to idle, we insert the corresponding modules (*insmod usb-ohci-sa1111*).

Also, in the Stargate platform, when the timer for the *sys_suspend* command expires and the node ”wakes up”, the wireless card goes to idle no matter what its previous state was (set by the *cardctl suspend* or *cardctl eject* commands). This is not the case for the webcam.

To characterize the Stargate’s power requirements, we use the HP E3631A power supply to power the Stargate through its daughter board. The power supply is configured to provide 5V, but since we had a protection diode, the actual voltage provided to the Stargate is 4.2V. Through the power supply, we record the current being drawn. Table I shows all possible states for the Crossbow Stargate platform and their power requirements.

State	Processor	Sensor	Radio	Storage	Power(W)
Sleep	sleep	sleep	sleep	sleep	0.28
P-idle	idle	sleep	sleep	idle	0.56
P-active	active	sleep	sleep	idle	1.60
PR-idle	idle	sleep	idle	idle	1.27
PR-active	active	sleep	idle	idle	2.35
PR-rx	idle	sleep	active	idle	2.29
PR-tx	idle	sleep	active	idle	2.48
PRS-idle	idle	idle	idle	idle	2.00
PRS-active	active	idle	idle	idle	3.09
PRS-sens	idle	active	idle	idle	2.23
PRS-rx	idle	idle	rx	idle	2.94
PRS-tx	idle	idle	tx	idle	3.11
PS-idle	idle	idle	sleep	idle	1.28
PS-active	active	idle	sleep	idle	2.35
PS-sens	idle	active	sleep	idle	1.51
PT-read	idle	sleep	sleep	read	1.34
PT-write	idle	sleep	sleep	write	1.37
PRT-read	idle	sleep	idle	read	2.11
PRT-write	idle	sleep	idle	write	2.14
PST-read	idle	idle	sleep	read	2.10
PST-write	idle	idle	sleep	write	2.14
PRST-read	idle	idle	idle	read	2.81
PRST-write	idle	idle	idle	write	2.84

TABLE I

STARGATE ENERGY CONSUMPTION CHARACTERIZATION

Although these current readings are accurate, the HP E3631A power supply has a limitation on how frequent it samples the current being drawn. We use a serial cable to connect the HP E3631A power supply to a laptop and acquire a sample every second. This sampling rate does not provide enough granularity to understand power consumed by state transitions. As future work, we intend to use a sampling oscilloscope or a analog to digital converter.

Related Work: In Bhardwaj et al. [4], a simple theoretical power consumption model for sensor networks

is proposed. The model derives an upper bound for the network lifetime.

The Panoptes sensor hardware [10] is similar to the Stargate hardware, but has higher power requirements. When we compare the power requirements for the Stargate platform (presented in Table I) with the Panoptes hardware (presented in Table 5 in [10]), we observe that the Stargate provides energy savings of up to 25%.

Another widely used platform in sensor networks are the Berkeley Motes [44]. A detailed accounting of the energy consumed by the Motes is presented in [38].

C. Node Activation Issues

In Section II-B, we describe the different energy consumption states exhibited by the Stargates. It is clear from the power levels associated with each state that, in order to minimize energy consumption, the Stargate and its peripherals must be kept in sleep as much as possible.

Switching a node's radio to sleep is a well-known power conservation strategy that has been employed by several sensor network protocols (e.g., [46], [35], [43], [8], [45], etc.). In most cases, the assumption is that communication dominates power consumption when compared to the other sensor node's tasks (e.g., sensing and processing). One of Meerkats' distinguishing features is due to the fact that its main sensors are cameras, and thus, the energy consumed by sensing and processing is no longer negligible; in the contrary, it may dominate the overall power consumption in a Meerkats sensor node. This calls for novel power savings strategies (complementing existing ones) which try to minimize consumption due to video sensing and processing by turning the camera on only when necessary. The basic approach is to keep the camera off as long as the target scene remains unchanged. This is similar to keeping the radio off when a node has no messages to send to neighboring nodes nor does it expect to receive any messages from its neighbors,

The question then becomes when and how to wake up a camera such that the probability of mis-detection is minimized while still being energy efficient? There are different ways to approach the problem. One possibility is to use a different, cheaper (in terms of power consumption) sensor, e.g., motion sensors, as tripwire. In general, motion sensors consume very little energy; however, they raise a few issues such as relatively high number of false positives and significant delay (up to 1 second).

In the current version of our system, where cameras are the only sensors, we employ a duty cycle based approach which periodically turns a node's radio on for a specified period of time to listen for messages. If a node

has a message to send, it will send it during this interval. One such message can be a "camera wake-up", which is generated by a node that detects an event alerting its neighbors of activity in the region. Figure 2 illustrates this scenario. Upon receiving a "camera wake-up", a node wakes up its camera to start taking pictures.

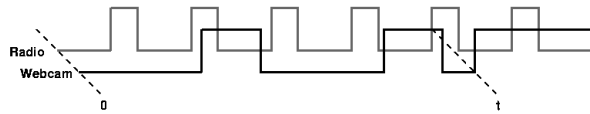


Fig. 2. At time t , radio receives a "camera wake-up" message from neighbor. Stargate turns on webcam and begins taking pictures.

Additionally, as described in Section III, nodes wake-up their cameras periodically to check whether there is an object of interest in its FOV. If so, it may start taking more pictures and/or send a "camera wake-up" message to neighboring nodes in the direction the object is traveling. This is illustrated in Figure 3.

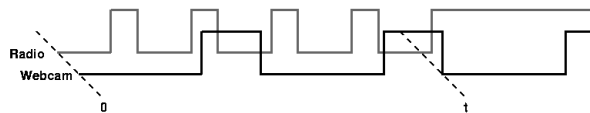


Fig. 3. At time t , camera sees object of interest. During next radio period, node sends 'wake-up' to neighbors and transmits image to sink.

As discussed in Section III, we anticipate that nodes in the periphery of the area under surveillance will act as primary event detectors, warning "internal" nodes which will take pictures "on-demand". However, by also having internal nodes wake up their cameras periodically, we decrease the system's overall mis-detection probability.

Sending a message only happens during the period the wireless card is active. In the worst case, if the wireless card is currently in sleep state, the node must wait until its next active cycle to send the message.

Every node maintains wake-up timers for each device (e.g., camera, wireless card). If these timers are large enough, the Stargate will begin a timed sleep for the entire node to minimize energy usage. At a given point in time, a node will load only the modules corresponding to the peripherals to be activated.

To achieve schedule synchronization among the nodes, we will employ a simple network-based clock synchronization mechanism. Basically, each Stargate's clock will be synchronized taking as reference a time server running on the gateway node. Time synchronization messages are exchanged under the radio duty cycle discussed above.

D. Camera Placement

The layout of the cameras in the environment is obviously very important for the performance of the overall system. A number of papers in the literature (e.g. [31], [16]) have studied the problem of visibility analysis and optimal placement. In this work we consider a sparse placement of cameras (as in [30], [25]) to maximize the covered area. This means that, in practice, the FOV of two cameras will seldom overlap.

An advantage of overlapping FOVs is that, when a surface patch is seen by both cameras, it is possible to compute the image correspondence and, if the cameras are geometrically calibrated, the 3-D position of the surface patch with respect to a reference frame attached to one of the cameras [28]. In some cases it is possible to compute the position of visible objects even from a single view. In particular, if the projection in the image plane of the point of contact between an object and the ground plane can be identified (see Sec. IV), the 2-D position of the object can be estimated, provided that the homography between the ground plane the camera's focal plane is known. This can be done by manual selection of a number of points of known geometry on the ground floor [18], or automatically by looking at trajectories of bodies moving of rectilinear motion at constant velocity [5].

For tracking applications and for efficient hand-off and camera activation, it is important that the cameras' relative localization and orientation is known, at least approximately. RF-based sensor localization as been often used for this purpose [6], [19], [32], [37], [14], [27]. An onboard compass, when available, may be used to determine the azimuth angle. Visual based approaches are also possible. For cameras with overlapping FOV, structured light [3], surveyed landmarks [12] and calibration widgets [2] have been proposed. Tracked moving objects can also be used for the calibration of two cameras with overlapping FOV [26], [40], [41]. If the cameras have non-overlapping FOV, then calibration is possible by tracking objects, normally under the assumption that the trajectory of the bodies between the two FOV is known or constrained (e.g. rectilinear) [17], [22], [34], [15].

We make a distinction between *end nodes* and *internal nodes* in the network. End nodes are placed at the edge of the network, or near points of high flux (e.g., near an entry door). These nodes are likely to be the first to see a body when it enters the monitored area. An internal node is normally in the neighborhood of one or more end nodes. As discussed in Sec. III, end and internal nodes require different activation policies.

III. IMAGE ACQUISITION SCHEDULING

The goal of the network is detect and track moving bodies within the covered area covered. Ideally, any time a body enters the FOV of a camera, the camera would take one or more snapshot of it. The visual data is used for event detection, data transmission in the chosen representation, and activation of nearby nodes which are likely to see the body next. This section is concerned with the tasking of the cameras in order to maximize a specific utility function while guaranteeing long network life.

A. Utility Function

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 body 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):

$$P(M_i^k | F_i^r, X^n) = P(M_i^k | F_i^r) \quad (1)$$

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 if 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 the 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" or MR_i):

$$MR_i = \frac{E[M_i]}{E[X]} \quad (2)$$

where the $E[\cdot]$ represents the expectation operator. Let $P_{M|F} = P(M_i^1 | F_i^1)$ and $P_{F|X} = P(F_i^1 | X^1)$. Using the total probability theorem, and remembering that the conditional distributions of interest are binomial, we can

write:

$$\begin{aligned}
\mathbb{E}[M_i] &= \sum_k k \mathbb{P}(M_i^k) \\
&= \sum_n \sum_r \sum_k \mathbb{P}(M_1^k | F_i^r) \mathbb{P}(F_i^r | X^n) \mathbb{P}(X^n) \\
&= \sum_n \sum_r \mathbb{E}[M | F_i^r] \mathbb{P}(F_i^r | X^n) \mathbb{P}(X^n) \\
&= \sum_n \sum_r r \mathbb{P}_{M|F} \mathbb{P}(F_i^r | X^n) \mathbb{P}(X^n) = \\
&= \mathbb{P}_{M|F} \sum_n \mathbb{E}[F | X^n] \mathbb{P}(X^n) = \mathbb{P}_{M|F} \mathbb{P}_{F|X} \mathbb{E}[X]
\end{aligned} \tag{3}$$

Hence, from (2), we maintain that:

$$\text{MR}_i = \mathbb{P}_{M|F} \mathbb{P}_{F|X} \tag{4}$$

In the next two subsections we will show how, in some practical situations, the two factors in the rhs of (4) can be estimated for end nodes and internal nodes, as defined in Sec. II-D.

B. End Nodes

Consider the case of Fig. III-B(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 triangle¹, 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.

Since this is an end node, it is likely to be the first node that can detect a person walking through that door. 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 be 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. III-B, 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:

$$\mathbb{P}_{F|X} = \int_{\phi \in \Phi} p_\phi(\phi) d\phi \tag{5}$$

The probability $\mathbb{P}_{M|F}$ of misdetection given that the person walks in the camera's FOV depends on the image acquisition policy of the camera. In practical cases, it is very convenient, in practice, to add a low-power sensor (such a passive infrared (PIR) motion sensor, which consumes fractions of mW) to trigger the more power-hungry camera node. However, in our analysis we assume that the camera takes snapshots at regular time intervals (with period T_i). This can be used as a worst-case scenario to benchmark the power consumption in the high-traffic case.

Under periodic image acquisition, a person walking through FOV_i is not detected if, for some m :

$$mT_i < t_{in} < t_{out} < (m+1)T_i \tag{6}$$

where $t_{in} = t_0 + t_1(\phi, v)$ is the time the person enters FOV_i , $t_{out} = t_0 + t_1(\phi, v) + t_2(\phi, v)$ is the time the person exits FOV_i , and t_0 is the time the person walks through the door. Since t_0 is, by hypothesis, an outcome of a Poisson process, then it is not difficult to see that the condition in (6) is verified with probability $1 - \min(t_2(\phi, v), T_i)/T_i$. Hence:

$$\mathbb{P}_{M|F} = \int_{\phi \in \Phi} \int_v p_\phi(\phi) p_v(v) \left(1 - \frac{\min(t_2(\phi, v), T_i)}{T_i} \right) dv d\phi \tag{7}$$

Fig. III-B(b) shows the relationship between the snapshot period T_i and the miss rate MR_i for the situation in Fig. III-B(a). This information (perhaps contained in a look-up table) can be used by the Resource Manager to decide a suitable snapshot 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 (see Sec. II-D), 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 suitably modifying (5) and (7). Likewise, uncertainty about the actual distributions of v and ϕ can be modeled by increasing the variance of the model distributions².

¹Note in passing that omniscams can be used [24], [42] resulting in anular traces. Omniscams have the advantage of larger FOV but reduced resolution. In addition they are more expensive.

²Note that $p_\phi(\phi)$ and $p_v(v)$ can, in principle, be learned by analyzing the data collected by the camera.

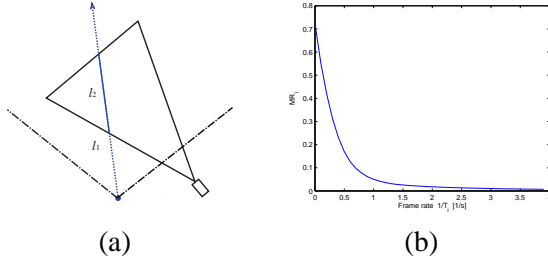


Fig. 4. (a): A possible layout of an end node. The triangular shape represents the node's FOV, while the angular sector represents the possible direction of motion. (b): The miss rate as a function of the snapshot rate ($1/T_i$).

C. 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, possibly together with other low-power sensors. Of course, in addition to this reactive behavior, an internal node may also follow a policy of regularly timed snapshots, to account for moving bodies that may have been missed by other nearby nodes/sensors.

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 localized within FOV_i at time t . Indeed, as discussed in Sec. IV, different levels of geometric information can be extracted, 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 is also known with a degree of uncertainty), the Resource Manager needs to decide: (1) which (if any) nearby cameras need to be alerted; (2) how many snapshots each of such cameras should take; (3) what are the optimal times for the snapshots. 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 snapshot should be taken at any time the body is within this field of view. Due to uncertainty in our knowledge of the precise camera and moving body geometry, this prediction will be only approximate, meaning that more than one cameras might have to be alerted, and more than one snapshot might 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 snapshot N_j it will take, and of the times $\mathbf{t}_j = \{t_{j,1}, \dots, t_{j,N_j}\}$ at which the snapshots are taken. For each value of N_j , the times \mathbf{t}_j that minimize the corresponding miss rate can be computed, resulting in the optimal (decreasing)

sequence of values $\text{MR}_j(N_j)$. Based on this knowledge, the Resource Manager can allocate the number of snapshot to be taken by each camera, balancing the need for a low miss rate with the available energy at each node.

We will give a brief example of how the sequence $\text{MR}_j(N_j)$ can be computed in the case of one nearby node detecting an event (Fig. III-C(a)). For simplicity, we will 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 velocity, $p_v(v)$ and of orientation, $p_\phi(\phi)$ of motion are modeled in the same way as in Sec. III-B. If no snapshot is taken by the j -th camera in response to the event detected by the i -th camera, then $\text{MR}_j(0) = P_{F|X}$, which is the probability that the body will cross FOV_j at some point, and can be computed as by (5). To compute $\text{MR}_j(1)$, we first need to express the miss rate as a function of the snapshot time $t_{j,1}$. This requires computing the probability that the times t_{in} and t_{out} at which the body enters and exits FOV_j are both before or both after $t_{j,1}$. In symbols:

$$t_{out} < t_{j,1} \text{ or } t_{in} > t_{j,1}$$

and therefore $P_{M|F} = P(t_{out} < t_{j,1}) + P(t_{in} > t_{j,1})$. Using the same symbols as in Sec. III-B, we observe that:

$$\begin{aligned} P(t_{out} < t_{j,1}) &= P(t_0 + t_1 + t_2 < t_{j,1}) \\ &= \int_{\phi \in \Phi} \int_v P(t_1(\phi, v) + t_2(\phi, v) < t_{j,1} - t_0) \cdot \\ &\quad \cdot p_v(v) p_\phi(\phi) dv d\phi \end{aligned} \quad (8)$$

Remembering that $t_1(\phi, v) = l_1(\phi)/v$ and $t_2(\phi, v) = l_2(\phi)/v$ we maintain that:

$$P(t_{out} < t_{j,1}) = \int_{\phi \in \Phi} \int_{\frac{l_1(\phi) + l_2(\phi)}{t_{j,1} - t_0}}^{\infty} p_v(v) p_\phi(\phi) dv d\phi \quad (9)$$

Likewise,

$$P(t_{in} > t_{j,1}) = \int_{\phi \in \Phi} \int_0^{\frac{l_1(\phi)}{t_{j,1} - t_0}} p_v(v) p_\phi(\phi) dv d\phi \quad (10)$$

Fig. III-C(b) shows the plot of $\text{MR}_j(1)$ as a function of $t_{j,1}$ for the case of Fig. III-C(a). In this case, $\text{MR}_j(1) = 0.63$.

The case of two snapshots ($t_{j,1} < t_{j,2}$) can be dealt with in a similar fashion. In this case, the j -th camera misses the moving body if any one of these disjoint events occurs:

$$t_{out} < t_{j,1} \text{ or } t_{j,1} < t_{in} < t_{out} < t_{j,2} \text{ or } t_{in} > t_{j,2}$$

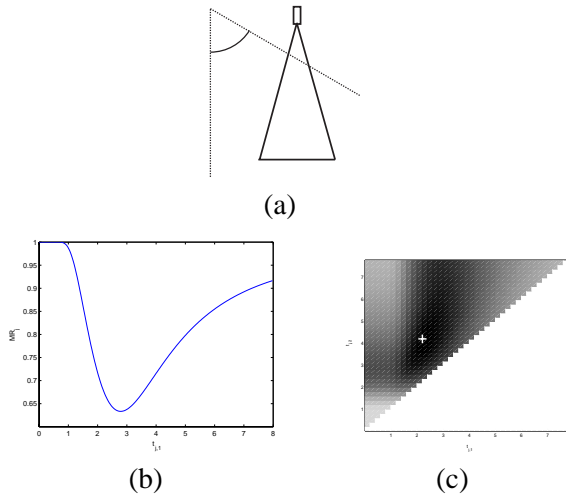


Fig. 5. (a): A possible layout of an internal node. An event has been detected at time t_0 by another sensor, which estimated that a body is moving within the specified angular sector. (b): The miss rate as a function of the time $t_{j,1}$ at which a single snapshot is taken by the camera. (c): The miss rate as a function of the times at which two snapshots $t_{j,1} < t_{j,2}$ are taken by the camera (the cross represent the pair of time instants that minimize the miss rate).

The probability of the first and of the third event above can be easily computed as in the single snapshot case. As for the second event, it is easy to see that:

$$\begin{aligned}
 & \mathbf{P}(t_{j,1} < t_{in} < t_{out} < t_{j,2}) \quad (11) \\
 &= \int_{\phi \in \Phi} \int_v \mathbf{P} \left(v < \frac{l_1(\phi)}{t_{j,1} - t_0} \text{ and } v > \frac{l_1(\phi) + l_2(\phi)}{t_{j,2} - t_0} \right) \\
 & \quad \cdot p_v(v) p_\phi(\phi) dv d\phi \\
 &= \int_{\phi \in \Phi} \int_{\frac{l_1(\phi)}{t_{j,1} - t_0}}^{\frac{l_1(\phi) + l_2(\phi)}{t_{j,2} - t_0}} p_v(v) p_\phi(\phi) dv d\phi
 \end{aligned}$$

with the understanding that the last integral is null whenever $\frac{l_1(\phi) + l_2(\phi)}{t_{j,2} - t_0} > \frac{l_1(\phi)}{t_{j,1} - t_0}$. Fig. III-C(c) shows the optimal $t_{j,1}, t_{j,2}$ for the case of Fig. III-C(a). The miss rate using two time instants ($\text{MR}_j(2)$) is equal to 0.43, which, as expected, is less than $\text{MR}_j(1)$.

IV. VISUAL PROCESSING

The goal of the Visual Processing module, which is implemented in each camera node, is to detect and characterize events within the camera’s field of view. There are two main strategies for visual event recognition. The first one is based on the detection of changes in appearance with respect to a “background” scene. This approach, sometime called *background subtraction*, models the appearance of the background when no events occur, and use statistical techniques to identify “foreground” areas where changes occurred. The main challenges in background subtraction techniques are in the representation of the background appearance, which

may change as a consequence of changes in illumination, slight camera motion, shadows, etc. Note that the background may include moving areas (corresponding, for example, to leaves rustling in the wind or water in a fountain). A training sequence is normally used to learn the parameters of the statistical model of the background.

The other strategy is to simply detect motion in the scene based on two (or more) snapshots taken at a short time lag ΔT from each other. The advantage of this approach is that it does not require a model for the change in appearance of the background, which is supposed to be constant in the brief time between the two snapshots. In addition, standard models for appearance changes during motion can be used, as discussed below. The disadvantage of motion analysis algorithms for event detection is that there is no motion in the two frames (e.g. because a person in the scene happened to stop just when the snapshots were taken), no event is detected.

In this paper we describe our approach to event detection based on motion analysis. Future work will look at combining these algorithms with background subtraction techniques for increased robustness. A visible surface patch moving in the scene determines (under some benign circumstances) the apparent motion u of points in the image, termed *optical flow*. For a given pixel, the variation ΔI of brightness due to motion is related to u and to the spatial image brightness gradient ∇I as by:

$$\nabla I \cdot u \approx \Delta I / \Delta T \quad (12)$$

where the notation “ \cdot ” here represent scalar product. Even though terms ∇I and ΔI are easily computed, relationship (12) cannot be used directly to compute u because it is under-constrained (one liner equation in the two components of u). This is the so-called *aperture effect*: only the component of the flow parallel to the image gradient can be computed based on local analysis. Several regularization methods have been proposed to overcome the aperture effect. One successful approach is based on the eigenvalue analysis of the *structure tensor* [20], which is computed starting from the brightness gradient values in a block centered around the pixel in exam. This process labels each image block with a “motion”, “no motion”, or “outlier” label (the latter indicating a block with change in appearance that is not well modeled by (12)).

Based on this motion analysis, an event can be detected when, for example, a certain number of image blocks are labeled as “moving” or “outlier”. Beside event detection, the following parameters can be estimated:

Orientation of motion. The average optical flow can provide a simple indicator of the orientation of the body

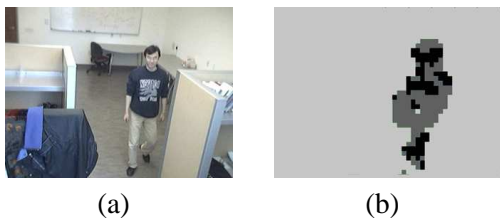


Fig. 6. (a): A frame of a sequence with a moving person. (b): Motion labeling using the optical flow algorithm of Sec. IV. White: no motion; Gray; motion; Black; outlier.

motion with respect to the optical axis. This information can be used to predict which camera will see the body next.

Position in ground plane. Suppose the homography mapping the ground plane to the image plane has been estimated with one of the methods mentioned in Sec. II-D. The position of the body on the ground plane can be computed if the projection of the body part that touches the ground (a person’s feet or a car’s wheels) can be identified. This is normally performed by detecting the lowest moving block in the image.

Velocity. To infer information about the moving body’s velocity, it is necessary to compute the body location at two (or more) different points in time. This requires that at least two pairs of snapshots are taken by the camera.

In the case of multiple bodies visible in the image, a motion clustering procedure should be implemented to identify all different bodies.

Hierarchical Representation

Visual motion detection and characterization is useful not only for tasking camera nodes, but also to enable a hierarchical representation of the image data. Depending on the currently available bandwidth and on the remaining power of the camera node (and on the relay nodes along the path to the sink), the resource manager decides the representation level at which to transmit image data. At the lowest level, all image data within a frame would be transmitted, compressed with a still image encoding standard such as JPEG. To reduce the amount of data to be transmitted, only the image blocks corresponding to moving areas could be encoded and sent. An intermediate strategy would be to assign fewer bits to the non-moving (background) areas using a Region-Of-Interest (ROI) technique. At a higher level, only the motion parameters in the image, or the body’s position and velocity, could be transmitted.

It is important to notice that, as the representation (abstraction) level increases, the data to be transmitted is reduced, but at the same time the risk of missing an important feature increases as well. For example, by only

transmitting data from a region of interest, we fully rely on the motion detection algorithm. Any error (a moving area that was not detected) is not recoverable, since the data is not being transmitted. It is important to factor the increased risk into the utility function that is used by the Resource Manager to decide at which representation level data should be transmitted.

V. QoS ROUTING

Because a large portion of the traffic flowing on the network will be (quasi) real-time (e.g., video), we need to have in place “better than best-effort” routing. Thus, when considering routes in Meerkats, metrics such as bandwidth and delay need to be taken into account. At the same time, energy is a critical limitation of the system and should also influence route selection.

In this section, we describe our approach to implement Meerkats’ QoS Routing component which includes (1) a variant of an ad-hoc routing protocol, and (2) network probing strategies to acquire current network state such as available bandwidth, delay, as well as residual battery power on nodes.

A. Routing

A number of ad hoc network unicast routing mechanisms have been proposed in recent years. In our initial tests, we employed the Ad-Hoc On-Demand Distance Vector (AODV) [33] protocol. We successfully ported AODV to the Stargate platform and were able to transmit video over a multi-hop Stargate network.

For our current implementation of Meerkats, we chose to use the Dynamic Source Routing (DSR) [23] protocol which, as a source routing mechanism, allows the complete path to be specified by the source. This means that, instead of having intermediate nodes make forwarding decisions at every hop, the source will select the complete path which will be based on: (1) information collected on the current state of the network and (2) input from the Resource Manager on what QoS metric(s) should take priority.

We are currently extending DSR’s reference implementation to discover multiple paths when they exist. The goal is to increase robustness to failures and spread the load associated with relaying packets as evenly as possible. Besides alleviating congestion, load balancing will also ensure that the energy consumed by forwarding data will be more evenly spread across nodes. We will investigate different strategies including alternate path- and multi-path routing. In the former, each of the multiple paths is used at a time, whereas in the latter, data is spread over the different paths.

Another benefit of maintaining multiple paths is that it will facilitate traffic differentiation through priority-based forwarding. In applications such as surveillance and tracking (especially when real-time response is required), certain traffic needs to be forwarded with high priority. For example, in the case the system believes an intruder has been detected, that information needs to be sent to the sink as soon as possible. When multiple paths exist, the one with the lowest delay could be used for urgent traffic.

Similarly to what we did for AODV in our initial testbed configuration, we are also modifying DSR's route cache mechanism to adjust itself to the type of traffic at hand. More specifically, we are using learning techniques to have the route cache timer automatically adjust its value based on the traffic dynamics. For example, if flows are long-lived, route cache entries could be kept for longer intervals.

B. Network Probing

Selecting routes that can deliver adequate quality of service to the application requires knowledge of the current state of the network through attributes such as available bandwidth and delay. Several efforts have addressed estimation of current network conditions in wired networks. As a result, a number of tools for assessing available bandwidth on the Internet have been proposed [21] [29]. One such tool uses a sequence of back-to-back packets sent at gradually increasing rate. Receive rate is measured and when it starts to be less than the sending rate, it is an indication that the available bandwidth was reached.

The problem with most of the (active) probing mechanisms designed for wired networks is that neither power nor bandwidth are considered scarce resources. More recently, bandwidth estimation techniques for multi-hop ad hoc networks (MANETs) attempt to estimate available bandwidth with minimal impact on power consumption. Typically, they employ passive probing which does generate additional traffic, and thus are less intrusive. For instance the *Listen* bandwidth estimation approach [9] listens in to the channel and determines how much available bandwidth there is during a given window of time. Essentially, available bandwidth is computed based on the portion of free channel time detected. Other MAC-layer passive probing mechanisms rely on measuring the average MAC queue size locally as well as in a node's neighborhood.

Note that we can also use an active probing strategy without really having to introduce additional traffic. The idea is to use real data to measure available bandwidth and delay. So, in the case that traffic is spread across

multiple paths, we can then measure their available bandwidth. However, if only one path is being used for a given interval of time, some form of statistical probing will need to be employed in order to measure the remaining path(s)'s conditions. For example, traffic can be sent on the alternate paths with some probability p . Alternatively, "real" active probing could be used on the alternate paths.

Bandwidth and delay estimates can be disseminated by piggy-backing them on MAC "Hello" messages or routing-layer path discovery signaling.

As previously discussed, routing must account for residual energy at nodes. Typically, given that energy is such a scarce resource, routes with higher minimum remaining battery will be preferred. Then, if more than one route can be selected, the one with the highest available bandwidth and/or shortest delay will be picked.

Similarly to how other attributes are disseminated, information on nodes' remaining battery can also be piggy-backed on MAC-layer "Hellos" and/or route discovery traffic. Section II-A describes how we will collect residual battery information in the Stargates.

VI. CONCLUSIONS

In this paper we described Meerkats, a wireless network of battery-operated camera nodes, that can be used for monitoring and surveillance of wide areas. We presented the Meerkats architecture, as well as results from (1) the statistical models we developed for image acquisition scheduling which, given a probability of mis-detection, try to minimize power consumption by deciding when cameras should wake up and acquire images; (2) our power consumption characterization of the Meerkats sensor node which is based on the Crossbow Stargate [13]; and (3) our visual processing algorithms that perform event detection, motion orientation, ground plane positioning, and velocity determination.

Besides the future work items mentioned throughout the paper, we intend to (1) develop an energy consumption model and associated prediction scheme, which will be used by the Meerkats Resource Manager; (2) implement and evaluate the QoS routing and network probing mechanisms we described; (3) develop a complete Meerkats prototype on the testbed we are building at UCSC.

REFERENCES

- [1] A. Aburto. FFT double precision benchmarks. <ftp://ftp.nosc.mil/pub/aburto/>, 2001.
- [2] P. Baker and Y. Aloimonos. Calibration of a multicamera network. In *IEEE Workshop on Omnidirectional Vision, Camera Networks and Non-Classical Cameras*, 2003.

- [3] J. Barreto and K. Daniilidis. Wide area multiple camera calibration and estimation of radial distortion. In *Proc. IEEE Workshop on Omnidirectional Vision, Camera Networks and Non-Classical Cameras*, 2004.
- [4] M. Bhardwaj and A. P. Chandrakasan. Bounding the lifetime of sensor networks via optimal role assignments. In *Proceedings of the Twenty First International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, New York, NY, USA, June 2002.
- [5] B. Bose and E. Grimson. Ground plane rectification by tracking moving objects. In *Proc. Joint IEEE Workshop on VS-PETS*, 2003.
- [6] N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, October 2000.
- [7] Engineering research center for collaborative adaptive sensing of the atmosphere (casa). <http://www.casa.umass.edu/>, 2004.
- [8] A. Cerpa and D. Estrin. Ascent: Adaptive self-configuring sensor network topologies. *SIGCOMM Comput. Commun. Rev.*, 32(1):62–62, 2002.
- [9] L. Chen and W. Heinzelman. Qos-aware routing based on bandwidth estimation for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 23(3):561–572, 2005.
- [10] W. chi Feng, B. Code, E. Kaiser, M. Shea, W. chang 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.
- [11] M. Chu, J. Reich, and F. Zhao. Distributed attention for large video sensor networks. In *Intelligent Distributed Surveillance Systems 2004 Seminar*, London, UK, February 2004.
- [12] R. T. Collins and Y. Tsin. Calibration of an outdoor active camera system. In *Proc. IEEE CVPR*, 1999.
- [13] Crossbow. Stargate. <http://www.xbow.com/>, 2004.
- [14] L. Doherty, K. Pister, and L. Ghaoui. Convex position estimation in wireless sensor networks. In *Proc. IEEE INFOCOM*, 2001.
- [15] T. Ellis, D. Makris, and J. Black. Learning a multi-camera topology. In *Proc. Joint IEEE Workshop on VS-PETS*, 2003.
- [16] M. Erdem and S. Sclaroff. Optimal placement of cameras in floorplans to satisfy task requirements. In *Proc. OMNIVIS*, 2004.
- [17] R. B. Fisher. Self-organization of a field of randomly placed ubiquitous sensors. In *ECCV 02*, 2002.
- [18] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, 2000.
- [19] J. Hightower, G. Borriello, and R. Want. SpotON: An indoor 3D location sensing technology based on RF signal strength. Technical Report 2000-02-02, University of Washington CSE, 2000.
- [20] H. Haussecker and B. Jaehne. A tensor approach for precise computation of dense displacement vector fields. In *Proc. Mustererkennung, Informatik Aktuell*, 1997.
- [21] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput. *IEEE/ACM Trans. Netw.*, 11(4):537–549, 2003.
- [22] O. Javed, Z. Rasheed, O. Alatas, and M. Shah. KNIGHT: A real time surveillance system for multiple overlapping and non-overlapping cameras. In *Proc. Int. Cong. Multimedia and Expo*, 2003.
- [23] 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.
- [24] M. T. K.C. Ng, H. Ishiguro. Monitoring dynamically changing environments by ubiquitous vision system. In *IEEE Workshop on Visual Surveillance*, June 1999.
- [25] S. Khan, O. Javed, Z. Rasheed, and M. Shah. Human tracking in multiple cameras. In *Proc. ICCV*, 2001.
- [26] S. Khan, O. Javed, and M. Shah. Tracking in uncalibrated cameras with overlapping field of view. In *Proc. IEEE Workshop on Performance Evaluation of Tracking and Surveillance*, 2001.
- [27] A. Ladd, K. Bekris, A. Rudys, Wallach, D.S., and L. Kavraki. On the feasibility of using wireless ethernet for indoor localization. *IEEE Trans. On Robotics and Automation*, 20(3):555–559, June 2004.
- [28] T. Matsuyama and N. Ukita. Realtime multitarget tracking by a cooperativedistributed visual system. *Proceedings of the IEEE*, 90(7):1136–1150, July 2002.
- [29] B. Melander, M. Bjorkman, and P. Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In *IEEE Globecom*, pages 415–420, 2000.
- [30] A. Mittal and L. Davis. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *International Journal on Computer Vision*, 51(3):189–203, Feb. 2003.
- [31] A. Mittal and L. S. Davis. Visibility analysis and sensor planning in dynamic environments. In *Proc. ECCV*, 2004.
- [32] N. Patwari, I. Alfred O. Hero, M. Perkins, N. S. Correal, and R. J. O’Dea. Relative location estimation in wireless sensor networks. *IEEE Transactions on Signal Processing, Special Issue on Signal Processing in Networks*, Nov. 2002.
- [33] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *WMCSA '99: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, page 90, Washington, DC, USA, 1999. IEEE Computer Society.
- [34] A. Rahimi, B. Dunagan, and T. Darrell. Simultaneous calibration and tracking with a network of non-overlapping sensors. In *Proc. IEEE CVPR*, 2004.
- [35] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 181–192, New York, NY, USA, 2003. ACM Press.
- [36] A. Rajgarhia, F. Stann, and J. Heidemann. Privacy-sensitive monitoring with a mix of IR sensors and cameras (demo). Technical Report ISI-TR-582, USC/Information Sciences Institute, November 2003.
- [37] C. Savarese, J. Rabay, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks USENIX technical annual conference. In *Proc. USENIX Technical Annual Conference*, 2002.
- [38] V. Shnayder, M. Hempstead, B. rong Chen, G. Werner-Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *ACM SenSys 04*, Baltimore, MA, November 2004.
- [39] Standard performance evaluation corporation. <http://www.spec.org>, 2004.
- [40] C. Stauffer and K. Tieu. Automated multicamera planar tracking correspondence modeling. In *Proc. IEEE CVPR*, pages 259–266, 2003.
- [41] G. P. Stein. Tracking from multiple view points: Self-calibration of space and time. In *Proc. DARPA Image Understanding Workshop*, 1998.
- [42] M. Trivedi, A. Prati, and G. Kogut. Distributed interactive video arrays for event based analysis of incidents. In *Proc. IEEE International Conference on Intelligent Transportation Systems*, pages 950–956, 2002.
- [43] T. van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *SenSys '03*:

Proceedings of the 1st international conference on Embedded networked sensor systems, pages 171–180, New York, NY, USA, 2003. ACM Press.

- [44] A. Woo. Mote documentation and development information. <http://www.eecs.berkeley.edu/~awoo/smartdust/>, 2000.
- [45] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. Peas: A robust energy conserving protocol for long-lived sensor networks. In *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 28, Washington, DC, USA, 2003. IEEE Computer Society.
- [46] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*, pages 1567–1576, New York, NY, USA, June 2002. USC/Information Sciences Institute, IEEE.