SensEye: A Multi-tier Camera Sensor Network

Purushottam Kulkarni, Deepak Ganesan, Prashant Shenoy and Qifeng Lu
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
{purukulk, dganesan, shenoy, qlu}@cs.umass.edu

ABSTRACT

This paper argues that a camera sensor network containing heterogeneous elements provides numerous benefits over traditional homogeneous sensor networks. We present the design and implementation of *SensEye*–a multi-tier network of heterogeneous wireless nodes and cameras. To demonstrate its benefits, we implement a surveillance application using *SensEye* comprising three tasks: object detection, recognition and tracking. We propose novel mechanisms for low-power low-latency detection, low-latency wakeups, efficient recognition and tracking. Our techniques show that a multi-tier sensor network can reconcile the traditionally conflicting systems goals of latency and energy-efficiency. An experimental evaluation of our prototype shows that, when compared to a single-tier prototype, our multi-tier *SensEye* can achieve an order of magnitude reduction in energy usage while providing comparable surveillance accuracy.

1. INTRODUCTION

1.1 Motivation

The relentless pace of technological growth has led to the emergence of a variety of sensors and networked sensor platforms that span the spectrum of cost, form-factor, resolution, and functionality. As an example, consider camera sensors, where available products range from expensive pan-tilt-zoom cameras to high-resolution digital cameras, and from inexpensive web-cams and "cell-phone-class" cameras to even cheaper, tiny cameras such as Cyclops [6] (see Table 1). A similar set of options are becoming available for sensor platforms, with choices ranging from embedded PCs to PDA-class Stargates [19], and from low-power Motes [10, 14] to even lower power systems-on-a-chip [1] (see Table 2). Due to these advances, the design and deployment of camera sensor networks—wireless networks of sensor nodes equipped with cameras—is now feasible and useful in a variety of application scenarios.

Consider the following applications of camera sensor networks: (i) *environmental monitoring*, where a network of wireless camera sensors is used to monitor wild-life habitats, rare species in remote locations and phenology (study of periodic biological phenomena) without being disturbed by humans; (ii) *surveillance*, where camera sensors are used to detect, recognize, and track people and vehicles in high-security areas such as airports; (iii) *live virtual tours*, where remote users navigate through an environment such as a museum and receive live, real-time video from camera sensors; and (iv) *live distance education*, where remote students use camera sensors deployed in a classroom to get a rich, live tele-immersive experience of an ongoing lecture.

Camera	*Power*	*Cost*	*Features*
Cyclops	33mW	Unpriced	128x128, fixed-angle, 10fps
Web-Cam	600mW	$75	640x480, auto-focus, 30 fps
PTZ Camera	1W	$1000	1024x768, retargetable pan-tilt-zoom, 30 fps

Table 1: Different camera sensors and their characteristics.

Platform	Type	Resources
Mica Mote	Atmega128 (6MHz)	84mW, 4KB RAM, 512KB Flash
Yale XYZ	OKI ArmThumb (2-57 MHz)	7-160mW, 32K RAM, 2MB external
Stargate	XScale PXA255 (100MHz–400MHz)	170-400 mW, 32MB RAM, Flash and CF card slots

Table 2: Different sensor platforms and their characteristics.

One possible approach for designing a camera sensor application is to choose a particular camera sensor and a suitable sensor platform (see Tables 1 and 2) and program each node to perform all application tasks. Such an approach yields a flat, single-tier network of homogeneous sensor nodes. However, given the availability of sensors and nodes with different capabilities and power requirements, it is also feasible to design the same application by employing heterogeneous elements. In this approach, resource-constrained, low-power elements are employed to perform simpler application tasks, while more capable, high-power elements take on more complex tasks. Doing so results in more judicious use of precious energy resources. To illustrate, a surveillance application can employ low-fidelity cameras to perform the simpler task of motion detection, while high-fidelity cameras can be woken up on-demand for object recognition and tracking. In contrast, in the single-tier approach, the choice of the camera sensor and the node is dictated by the most-demanding application task, causing simpler tasks to consume more resources than are necessary when executing on these more capable elements.

Since power consumption is a critical design issue in sensor networks, a heterogeneous approach can optimize power consumption and maximize network lifetime when compared to the single-tier approach. Consequently, in this paper, we study techniques for designing *multi-tier camera sensor networks*. By a multi-tier net-

work, we mean that the sensors are organized hierarchically into multiple tiers. For instance, a two-tier surveillance application may consist of low power cameras at the bottom tier that trigger higher resolution cameras at the upper tier in an on-demand fashion. The advantages of a multi-tier sensor network over a single-tier network are many: low cost, high coverage, high functionality, and high reliability. Depending on how they are designed, single tier systems often meet only a subset of these requirements. For instance, low cost can be achieved by using a single tier of inexpensive sensors but at the expense of functionality. High coverage can be achieved using a dense deployment of untethered sensors that can be placed anywhere but power considerations can sacrifice reliability. High functionality can be achieved by employing high fidelity sensors but at the expense of sacrificing coverage due to the high cost. Thus, a single choice along the axes of power, cost, or reliability will result in a sensor network that sacrifices one or more of the key requirements. In contrast, by employing different elements to perform tasks with different requirements, multi-tier networks provide a better balance of cost, coverage, functionality, and reliability.

1.2 Research Contributions

This paper presents design and implementation of *SensEye*, a multi–tier network of wireless sensor nodes and camera sensors that have different capabilities across tiers. To the best of our knowledge, this is the first work to demonstrate the benefits of employing a multi-tier camera sensor network over traditional single-tier sensor networks. The design and implementation of *SensEye* has resulted in several contributions.

Whereas latency (performance) and energy-efficiency are conflicting goals in a battery-powered single-tier network, we show that a multi-tier network can *achieve low latencies without sacrificing energy-efficiency*—something that is infeasible in traditional single-tier networks.

To demonstrate that these conflicting goals can be reconciled in a multi-tier network, we implement a simple surveillance application using *SensEye*. Our goal is not to build a better surveillance application than that in the literature [11], rather it is to demonstrate the benefits of multi-tier networks. *SensEye* surveillance comprises three tasks: object detection, recognition and tracking. We propose numerous mechanisms and optimizations to achieve low-latency, low-power object detection, accurate object localization, low-latency inter-tier wakeup, low-power object recognition and tracking. Overall, our design process illustrates how various sensing and processing tasks should be mapped to different tiers of a multi-tier network. Our mechanisms are designed to exploit redundancies in camera coverage resulting from a dense deployment of nodes. For instance, we demonstrate how multiple overlapping cameras can collaborate to localize an object and how localization can be exploited for energy-efficient wakeups.

We implement *SensEye* in a three-tier network comprising four types of camera sensors on Motes, Stargates and embedded PCs. An experimental evaluation of our prototype shows that in terms of energy usage, *SensEye* is better than a single-tier system by factors of 9.75 and 6.3, when using Cyclops and CMUcam cameras respectively. Despite this significant energy reduction, *SensEye* provides similar detection performance, with only 6% more missed detections when compared to a single-tier system. Our component-level benchmarks indicate that the detection latency and energy usage at Tier 1 is an order of magnitude less than that at Tier 2. Our experiments also reveal that the mean localization errors of a CMUcam and a webcam are 20-35%and 4.8%, respectively, indicating that while detection can be performed using lower-fidelity CMUcams, tracking is best done using higher-fidelity web-cams.

The remainder of the paper is structured as follows: Section 2 presents background and our system model. Section 3 present the design of *SensEye* while Section 4 presents implementation details. We present an experimental evaluation of *SensEye* in Section 5 and related work in Section 6. Finally, Section 7 presents our conclusions.

2. BACKGROUND AND SYSTEM MODEL

In this section, we discuss common processing tasks in a camera sensor network, followed by the system model and the key design principles that govern our work.

2.1 Camera Sensor Network Tasks

A camera sensor network will need to perform several processing tasks in order to obtain useful information from the video and images acquired by various camera sensors. Our work is motivated by two applications, namely monitoring of rare species in remote forests and surveillance in high-security environments. Both applications have numerous characteristics in common and involve three key tasks.

Object detection: First, the application needs to detect the presence of a new object whenever it enters the monitored environment. To illustrate, the rare species monitoring application needs to detect the presence of each animal that enters the monitored environment, while the surveillance application needs to detect vehicles or people that enter the high-security area. A good detection algorithm will minimize the latency to detect each new object that enters the monitored area.

Object recognition: Once a new object is detected, it needs to be classified to determine its type (e.g., a car versus a truck, a tiger versus a deer). This process, referred to as object recognition, enables the application to determine if the object is of interest and whether further processing is warranted. For instance, a surveillance system may be interested in counting the number of trucks on a highway but not cars. In our work, we assume that an image database of all interesting objects is available a priori, and the recognition step involves determining if the newly detected object matches one of the objects in this database.

Object tracking: Assuming the new object is of interest to the application, it can be tracked as it moves through the environment. Tracking involves multiple tasks: (i) computing the current location of the object and its trajectory, (ii) handoff of tracking responsibility as an object moves out of visual range of one camera sensor and into the range of another, and (iii) streaming video or a sequence of still images of the object to a logging store or a monitoring station. The goal of our work is to devise a hardware and software architecture to perform these tasks so as to optimize power consumption, without sacrificing performance metrics such as latency and reliability. As explained earlier, rather than choosing a single platform and a single type of camera sensor, our work focuses on multi-tier networks where the detection, recognition and tracking may be performed on different nodes and cameras to achieve the above goal.

2.2 System Model

Our work assumes a camera sensor network comprising multiple tiers (see Figure 1). A canonical sensor node within each tier is assumed to be equipped with a camera sensor, a micro-controller, and a radio as well as on-board RAM and flash memory. Nodes are assumed to be tetherless and battery-powered, and consequently, the overall constraint for each tier is energy. Within each tier, nodes are assumed to be homogeneous, while different tiers are assumed to be heterogeneous with respect to their capabilities. In general, we assume that the processing, networking, and imaging capabil-

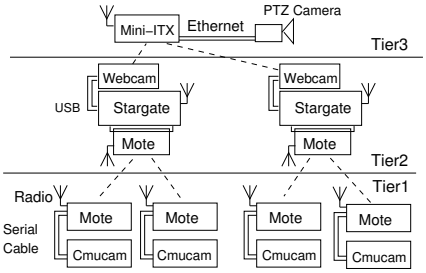


Figure 1: The multi-tier *SensEye* hardware architecture.

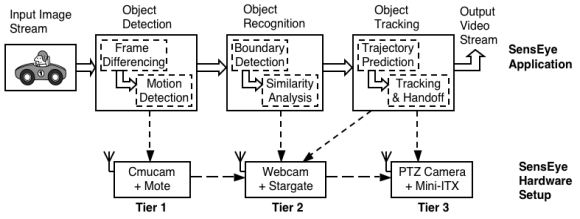


Figure 2: Software architecture of *SensEye*.

ities improve as we proceed from a lower tier to a higher tier, at the expense of increased power consumption. Consequently, to maximize application lifetime, the overall application should use tier-specific resources judiciously and should execute its tasks on the most energy-efficient tier that has sufficient resource to meet the needs of that task. Thus, different tasks will execute on different tiers and various tiers of camera sensor network will need to interact and coordinate to achieve application goals. Given these intra- and inter-tier interactions, application design becomes more complex—the application designer needs to carefully map various tasks to different tiers and carefully design the various interactions between tasks.

One of the goals of *SensEye* is to illustrate these tradeoffs while demonstrating the overall benefits of the multi-tier approach. To do so, *SensEye* assumes a three-tier architecture (see Figure 1). The lowest tier in *SensEye* comprises Mote nodes [14] equipped with 900MHz radios and low-fidelity Cyclops or CMUcam camera sensors. The second *SensEye* tier comprises Stargate [19] nodes equipped with web-cams. Each Stargate is equipped with an embedded 400MHz XScale processor that runs Linux and a web-cam that can capture higher fidelity images than Tier 1 cameras. Each Tier 2 node also consists of two radios—a 802.11 radio that is used by Stargate nodes to communicate with each other, and a 900MHz radio that is used to communicate with Motes in Tier 1. The third tier of *SensEye* contains a sparse deployment of high-resolution pan-tilt-zoom cameras connected to embedded PCs. The camera sensors at this tier are retargetable and can be utilized to fill small gaps in coverage provided by Tier 2 and to provide additional redundancy for tasks such as localization.

Nodes in each tier and across tiers are assumed to communicate using their wireless radios in ad-hoc mode; no base-stations are assumed in this environment. The radio interface at each tier is assumed to be individually duty-cycled to meet application requirements of latency and lifetime constraint on each node. Consequently, the application tasks need to be designed carefully since the radios on the nodes (and the nodes themselves) are not "always-on".

Given the above system model, we present key design principles, followed by the design and implementation of *SensEye*.

2.3 Design Principles

Our design of the *SensEye* multi-tier camera sensor network is based on the following principles.

- **Principle 1:** *Map each task to the least powerful tier with sufficient resources:* In order to judiciously use energy resources, each sensing and processing task should be mapped to the least powerful tier that is still capable of executing it reliably within the latency requirements of the application—running the task on a more capable tier will only consume more energy than is necessary.
- **Principle 2:** *Exploit wakeup-on-demand:* To conserve energy, the processor, radio and the sensor on each node are duty-cycled. Our system employs triggers to wake up a node in an on-demand fashion and only when necessary. For example, a higher-fidelity camera can be woken up to acquire a high-resolution image only after a new object is detected by a lower tier. By putting more energy-constrained higher-tier nodes in sleep mode and using triggers to wake them up on-demand, our system can maximize network lifetime.
- **Principle 3:** *Exploit redundancy in coverage:* The system should exploit overlaps in the coverage of cameras whenever possible. For example, two cameras with overlapping coverage can be used to localize an object and compute its (x, y, z) coordinates in the environment; this information can then be used to intelligently wakeup other nodes or to determine the trajectory of the object. Thus, redundancy in sensor coverage should be exploited to improve energy-efficiency or performance.

3. SENSEYE DESIGN

SensEye seeks to provide a *low-latency yet energy-efficient* camera sensing solution. Latency and energy-efficiency are conflicting system goals. To achieve low-latency sensing, sensors need to detect, recognize and track new objects as they enter and move across the field of view of the camera network and minimize missed objects. In contrast, energy-efficient sensing requires that sensors and nodes are switched off as much as possible (duty-cycled), which adversely impacts the latency of sensing and hence the reliability. Duty-cycling a distributed camera network incurs other sources of latency since wakeup triggers need to be propagated across distributed sensor nodes, and operating system latency is incurred for switching from sleep state to active state.

The primary insight in *SensEye* is that *careful task allocation across tiers enables the system to achieve low energy usage while providing latencies that are close to an always-on single-tier system*. In this section, we present different components of the *SensEye* architecture (see Figure 2) and details of how these components are handled in our multi-tier system.

3.1 Object Detection

The first task of a camera sensor network is to ensure that an object is detected as soon as it enters, and before it leaves the field of view of the network of cameras. While keeping the cameras always on achieves low-latency detection, it is very energy-inefficient, since cameras and nodes will continuously consume energy. On the other hand, the camera and the sensor node can be duty-cycled and woken up periodically to acquire an image of the environment; the image is then processed to detect the presence of new objects. The longer the period between successive images, the higher the latency

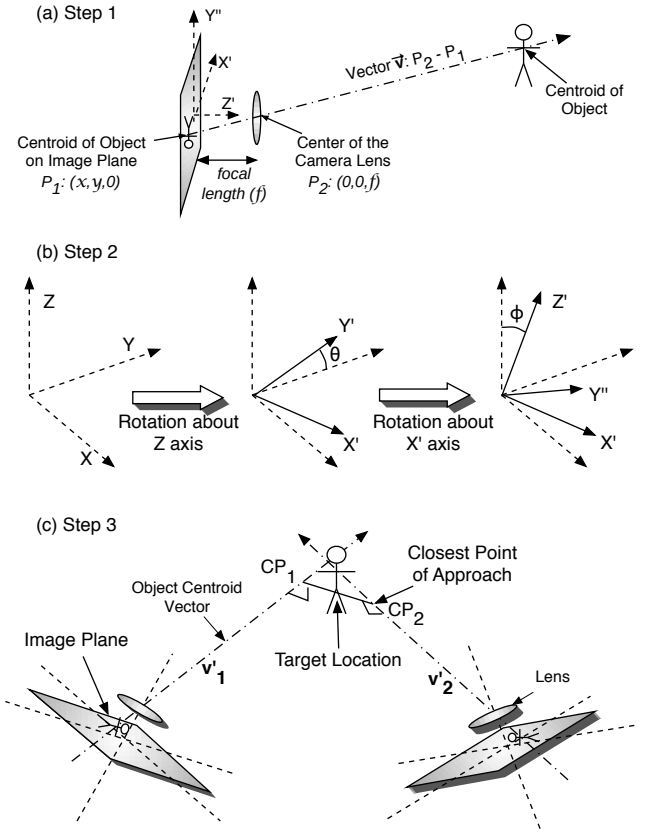


Figure 3: Localization in three dimensions using two cameras.

of detection, and the lower the reliability. The advantage, though, is the efficient use of energy resources.

In general, object detection requires little resources, and hence, it can be performed at the most energy-efficient tier. At this tier, since each wakeup consumes limited energy, the sleep period between *successive wakeups can be made small*, thereby enabling low-latency detection. In addition, *SensEye* employs a randomized duty-cycling algorithm where different cameras are woken up at different times to further reduce the latency to detect an object.

Each camera and its node performs object detection via simple frame differencing. Each camera acquires an background image of the environment at system calibration time. The pixel difference between each newly acquired image and the background image is computed and this difference is used to flag the presence of a new object when it enters the visual range of the camera. Observe that frame differencing is a relatively simple operation, and hence, *SensEye* employs low-fidelity energy-efficient sensors at Tier 1 for the task of object detection.

3.2 Object Localization

Once an object has been detected at Tier 1, nodes in Tier 2 need to be woken up for further processing. To intelligently wakeup the "correct" nodes in Tier 2, the Tier 1 nodes need to compute the 3D coordinates of the object and then determine which Tier 2 nodes have cameras pointing at this location. The location of the object can also be used by the retargetable Tier 3 cameras to get corresponding angles of pan and tilt to view the object. The process of computing the coordinates of an object is referred to as localization. *SensEye* uses triangulation techniques for localization—if the object is simultaneously viewed from two cameras, and if the location and orientation of the two cameras is known, then the location of the object can be calculated. The key insight is that even though the Tier 1 cameras are low-power and have coarse resolution, they can provide *sufficiently accurate localization* for making decisions on where the target is, and which Tier 2 nodes to wakeup.

Accurate localization requires three elements: (a) camera calibration to find the relative locations and orientations of cameras at different tiers, (b) synchronized readings at multiple cameras to limit localization error in the case of moving objects, and (c) location estimation based on optics and geometry.

Our localization scheme works for a 3D setting and assumes that cameras are calibrated at system setup time and their orientations are known relative to a global reference frame[1]. To enable synchronized sampling, different Tier 1 devices are assumed to be synchronized using a single reference beacon from a Tier 2 node (using a time-synchronization protocol such as RBS [9]). These beacons enable cameras to synchronize their pictures for purposes of localization.

Camera localization in 3D consists of three steps as shown in Figure 3. First, each camera calculates the vector along which the object's centroid is located with respect to its own frame of reference. Second, these vectors are rotated and translated to the global frame of reference using information about each camera's location and orientation. Finally, the location of the object is computed from the closest point of approach between the two vectors. We describe these steps in more detail below.

Step 1: Calculation of vector along direction of object location

As shown in Figure 3(a), the camera coordinate space is assumed to be the following: the image plane is the X-Y plane and the central axis perpendicular to the image plane is the Z axis. The center of the camera lens is at point $P_2 : (0, 0, f)$, where f is the focal length of the lens, and the centroid of the image of the object on the image plane is $P_1 : (x, y, 0)$. The vector, v, along which the object's centroid lies is, therefore, computed as $\mathbf{v = P_2 - P_1 = \{x, y, f\}}$.

Step 2: Transforming vector to global reference frame

To translate the object's vector, v, from the camera's reference frame to the global reference frame, we use the rotation and translation matrices obtained during calculation of the camera orientations. Each camera's orientation consists of a translation and two rotations. The translation from the global reference origin to the camera location is denoted by a translation matrix $\mathbf{T}$. Figure 3(b) shows the orientation of a camera as a composite of two rotations Initially, the camera is assumed to positioned with its central axis along the Z axis and its image plane parallel to the global X-Y plane. First, the camera is rotated by an angle of θ in the counter clockwise direction about the Z axis, resulting in X' and Y' as the new X and Y axes. Next, the camera is rotation by an angle ϕ in the clockwise direction about the X' axis, resulting in Y" and Z' as the new Y and Z axes. The two rotations are represented by a rotation matrix $\mathbf{R}$ and can be used to reverse transform the vector calculated in Step 1 to the global reference frame. If v_1 and v_2 are the two vectors along the direction of object location from cameras 1 and 2 respectively, the two corresponding vectors in global reference frame are:

$$v_1' = R_1.v_1 + T_1 \quad (1)$$

$$v_2' = R_2.v_2 + T_2 \quad (2)$$

where, R_1 and R_2 are the rotation matrices and T_1 and T_2 are the

[1]The calibration process can be automated using a combination of tilt sensors and positioning systems such as GPS and Cricket [15].

translation matrices for the two cameras respectively. The rotation matrix $\mathbf{R}$ takes the following form:

$$R = \begin{bmatrix} Cos\theta & -Sin\theta Cos\phi & -Sin\theta Sin\phi & 0 \\ Sin\theta & Cos\theta Cos\phi & Cos\theta Sin\phi & 0 \\ 0 & -Sin\phi & Cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

where, θ and ϕ are rotation angles as described in Step 2.
The translation matrix $\mathbf{T}$ takes the form:

$$T = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

where a,b and c are the translation magnitudes along the X,Y and Z global reference axes respectively.

Step 3: Object Location using Closest Point of Approach

Given the two vectors, v_1' and v_2', their intersection is the location of the object as shown in Figure 3(c). Since the lines are in three dimensions they are not guaranteed to intersect especially due to error in centroid computation and camera calibration. A standard technique used for approximating the intersection is using the Closest Point of Approach [7]. The closest point of approach gives the shortest distance between the two lines in three dimensions. We use this method to get points CP_1 and CP_2, the closest points between vectors v_1' and v_2' respectively. The location of the object is given by the mid-point of CP_1 and CP_2.

Note that camera calibration and localization in 2D are simpler cases of the more general 3D technique presented above.

3.3 Inter-tier Wakeup

Due to the higher power requirements of Tier 2 nodes and cameras (e.g.: Stargate and the web-cams in Table 1 and 2), each Tier 2 node is normally in sleep (or suspend) mode to conserve energy. Once an object is detected at Tier 1, one or more Tier 2 nodes need to be woken up for further processing. Inter-tier wakeup is a challenging problem both from an energy and latency perspective.

From an energy perspective, inter-tier wakeup should ensure that there are no wasteful wakeups of Tier 2 nodes. Localization at Tier 1 is a key component of our energy-efficient wakeup algorithm since it can be used to make intelligent decisions on precisely which node to wakeup. We assume that Tier 1 nodes know the visual range of each Tier 2 camera in their vicinity (from system calibration), and hence, can use the localized coordinates of the object to determine the most appropriate Tier 2 node with a camera pointing at this location. If no appropriate Tier 2 node can be identified, *SensEye* wakes up a Tier 3 retargetable camera, and uses its pan and tilt capability to point it to the location where the target was localized. Localization is feasible only when at least two nodes can view the object—if only a single Tier 1 sensor detects an object, then localization can not be performed and the node must wake up *all* Tier 2 nodes that have overlapping coverage with itself (this list depends on the initial placement).

From a latency perspective, the separation of detection and recognition tasks across two different tiers introduces latency between the execution of these two tasks. The latency includes the delay in receiving and processing the wakeup packet as well as the delay in waking up the Tier 2 node. To ensure that recognition is performed before a moving object leaves the visual range of the Tier 2 camera, this latency should be as small as possible. *SensEye* uses several optimizations to reduce the total latency of detection and recognition. The wakeup process begins by the transmission of a wakeup packet to a Tier 2 node (similar to "wakeup-on-wireless" [8]). Upon receiving this wakeup message, the Tier 2 node needs to transition from the suspend state to awake state. This transition duration is kept small by ensuring that only the bare minimum of device drives are running—thereby keeping the driver load times small during wakeup. Several additional suspend-to-active switching optimizations are also performed as discussed in [18].

3.4 Object Recognition

Once a new object is detected at Tier 1, it needs to be classified using a recognition algorithm to determine if it is of interest to the application. The recognition step eliminates uninteresting objects and helps focus application resources on objects that merit further attention. In *SensEye* recognition involves obtaining an image of the object, identifying object features and searching the image database for a match. Clearly, accurate recognition requires a high-fidelity image of the object and significantly greater processing and memory resources than available on a Tier 1 node such as a Mote (6MHz processor and 4KB RAM). Consequently, *SensEye* executes the recognition algorithm at Tier 2 using higher-fidelity web-cams and the more-capable 400MHz XScale processors on the Stargates.

Object recognition is well studied in the vision community, and a slew of techniques have been designed [7]. Since the focus of our work is on the design of a multi-tier camera sensor network, as opposed to computer vision, we assume that any of these algorithms can be employed in *SensEye*. As proof of concept, we implement two recognition algorithms from the literature: a simple face recognition algorithm [3, 13] and a second algorithm that isolates the object using connected components [7, 17], and uses a simple color-based heuristic to match the object to the image database. While these algorithms are adequate for our purpose, more sophisticated recognition algorithms can be employed in real-world settings [11]. *SensEye* uses software optimizations to reduce the latency of object recognition and hence the energy. Latency is incurred since the object recognition program has to initialize the camera and wait for the camera frame capture to stabilize before performing the object recognition task. *SensEye* uses accurate calibration of the camera on the Tier 2 node to determine the minimum time required for the image stabilization upon wakeup, so that recognition can be performed with low latency.

3.5 Object Tracking

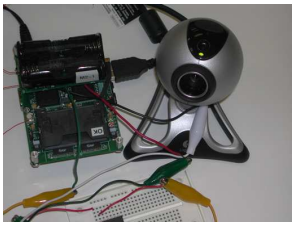
Tracking of moving objects involves multiple sensing and processing tasks—continuous object detection as it moves through the field of view of cameras, object recognition to ensure that the object of interest to the application is tracked across cameras, and finally trajectory prediction to estimate the movement pattern of the object.

Object tracking in *SensEye* involves a combination of detection, localization, inter-tier wakeup as well as recognition. As the object moves through the covered region, different Tier 1 nodes detect the target. If multiple nodes detect the target, localization can be used to accurately pinpoint the location of the target. Continuous localization can used to track the path of the moving object. Our current prototype can handle slow moving objects, and trajectory prediction schemes for fast moving objects (using techniques such as [23]) is the subject of ongoing research. Future *SensEye* mechanisms will also enable images or video acquired from the object to be displayed at a monitoring station or logged to a persistent store.

4. SENSEYE IMPLEMENTATION

This section describes the implementation of *SensEye* based on the design discussed in the previous section.

(a) Tier 1



(b) Tier 2

Figure 4: Prototype of a Tier 1 Mote and CMUcam and a Tier 2 Stargate, web-cam and a Mote.

4.1 Hardware Architecture

Our *SensEye* implementation uses four types of cameras—the Agilent Cyclops [6], the CMUcam Vision sensor [5], a Logitech Quickcam Pro Webcam and a Sony PTZ camera—and three platforms—Crossbow Motes [14], Intel Stargates [19] and a mini-ITX embedded PC. *SensEye* is a three-tier network, with the first two tiers shown in Figure 4.

Tier 1: Tier 1 of *SensEye* comprises a low-power camera sensor such as Cyclops [6] connected to a low-power Mote [14] sensor platform. The Cyclops camera is currently available only as a prototype. Therefore, we use the Cyclops platform for our individual component benchmarks and substitute it with a similarly constrained but higher power CMUcam for our multi-tier experiments. The Cyclops platform comprises an Agilent ADCM–1700 CMOS camera module, an ATMega128 micro-controller and a Xilinx FPGA. The board attaches using a standard 32-pin connector to a Mote, and communicates to it using UART. The software distribution for Cyclops [6] provides support for frame capture, frame differencing and object detection.

The CMUcam is a less power-optimized camera that comprises an OV7620 Omnivision CMOS camera and a SX52 micro-controller. The CMUcam connects to a Mote using a serial interface, as shown in Figure 4(a). The CMUcam has a command set for its micro-controller, that can be used to wakeup the CMUcam, set camera parameters, capture images, perform frame differencing and tracking.

Tier 2: A typical Tier 2 sensor comprises of a more-capable platform and camera and a wakeup circuit to wakeup the node from the sleep or suspend state upon receiving a trigger from a Tier 1 node. In our implementation, as shown in Figure 4(b), we use a Intel Stargate sensor platform with an attached Mote that acts as the wakeup trigger. Since the Stargate does not have hardware support for being woken up by the Mote, we used a relay circuit described in Turducken [18] for this purpose. The Logitech Webcam connects to the Stargate through the USB port.

Tier 3: A Tier 3 node comprises a Sony SNC-RZ30N PTZ camera connected to an embedded PC running Linux.

4.2 Software Architecture

The software framework of *SensEye* is shown in Figure 5. The description of our software framework assumes that Tier 1 comprises Motes connected to CMUcam cameras. Substituting a CMUcam with a Cyclops involves minimal change in the architecture. The first two tiers of *SensEye* comprise four software components: (i) CMUcam Frame Differentiator, (ii) Mote–level Detector, (iii) Wakeup Mote, and (iv) Object Recognition at the Stargate. Following is the description of each component's functionality.

CMUcam Frame Differentiator: The CMUcam receives periodic instructions from the Mote to capture an image for differencing. On each such instruction, the CMUcam captures the image in view, quantizes it into a smaller resolution frame, performs frame differencing with the reference background frame and sends back the result to the Mote. Frame differencing results in image areas where objects are present to be highlighted (by non–zero difference values). The CMUcam has two modes of frame differencing, (i) a low resolution mode, where it converts the current image (of 88×143 or 176×255) to a 8×8 grid for differencing, or (ii) high resolution mode, where a 16×16 grid is used for differencing. The frame differencing is at very coarse level and hence has relatively high error to estimate location of the object or its bounding box.

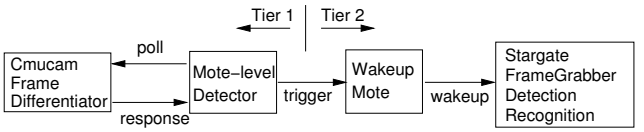


Figure 5: *SensEye* Software Architecture.

Mote–Level Detector: The function of the Tier 1 Mote is to control the CMUcam and send object detection triggers to the higher level nodes. On startup, the Mote sends initialization commands to the CMUcam, to set its background and frame differencing parameters. Periodically, based in its sampling rate, the Mote sends commands to the CMUcam to capture an image and perform frame differencing. The CMUcam responds with the frame difference result. The Mote uses a user–specified threshold and the returned frame difference result to decide whether an event (object appearance or object motion) has occurred. If an event is detected, the Mote broadcasts a trigger for the higher tier. On no event detection, the Mote sleeps till the next sampling time. Additionally, the Mote duty-cycles the CMUcam by putting it to sleep between two sampling instances.

Wakeup Mote: The Mote connected to the Stargate receives triggers from the lower tier Motes and is the interface between the two tiers. On receiving a trigger, the Mote can decide whether to wakeup the Stargate for further processing. Typically, the localized coordinates are used for this purpose. Rather than actually computing the object coordinates at a Tier 1 Mote, which requires significant coordination between the Tier 1 nodes, our implementation relies on a Tier 2 Mote to compute these coordinates—the Tier 1 nodes simply piggyback parameters such as θ, ϕ and the centroid of the image of the object with their wakeup packets. The Tier 2 Mote then uses techniques described in Section 3.2 to derive the coordinates. The Stargate is then woken up if the object location is within its field of view, otherwise the trigger is ignored.

High Resolution Object Detection and Recognition: Once the Stargate is woken up, it captures the current image in view of the webcam. Frame differencing and connected component labeling [17] of the captured image along with the reference background image is performed. This yields the pixels and boundaries where the potential objects appear in the image. Smoothing techniques based on color threshold filtering and averaging of neighboring region are used to remove noise pixels. Each potential object then has to be recognized. In our current implementation, we use an averaging scheme based on the pixel colors on the object. The scheme produces an average value of the red, green and blue components of the object. The values can be matched against a library of objects and

Mode	Latency (ms)	Average Current (mA)	Power Consumption (mW)	Energy Usage (mJ)
Mote Processing	136	19.7	98.5	13.4
CMUcam Object Detection	132	194.25	1165.5	153.8

Table 3: *SensEye* Tier 1 (with CMUcam) latency breakup and energy usage. Total latency is 136 ms and total energy usage is 167.24 mJ.

the closest match is declared as the object's classification. *SensEye* can be extended by adding sophisticated classification techniques, face recognition and other vision algorithms. We evaluate a face recognition system in the Experimental section to get an idea of its latency and power requirements.

PTZ Controller: The Tier 3 retargetable cameras are used to fill gaps in coverage and to provide additional coverage redundancy. The pan and tilt values for the PTZ cameras are based on localization techniques as described before. The cameras export a HTTP API for program–controlled camera movement. We use one such HTTP–based camera driver [4] to retarget the Tier 3 PTZ cameras.

5. EXPERIMENTAL EVALUATION

In this section we present detailed experimental evaluation of *SensEye*. Specifically, we evaluate several power consumption, latency and camera benchmarks to characterize individual components and compare single–tier and multi–tier *SensEye* systems.

5.1 Component Benchmarks

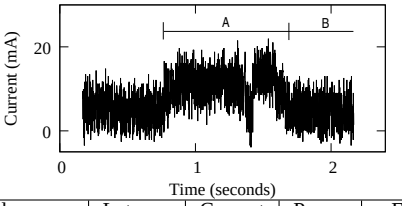
In this section we measure benchmarks of individual components that collectively form the *SensEye* system. The benchmarks reported are latency and energy usage, localization accuracy and object recognition performance.

5.1.1 Latency and Energy Consumption

Since minimizing energy usage is an important goal of *SensEye*, we systematically breakdown the power consumption and latency of each hardware and software component in its different modes of operation. Tables 3 and 4 report latency, average power consumption and the energy usage for object detection at Tier 1 and Table 5 provides a similar breakdown for object recognition at Tier 2.

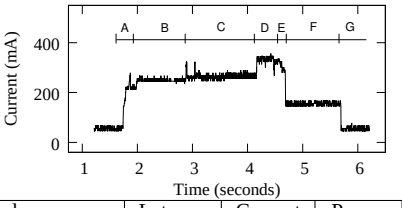
Tier 1: As seen from Table 3, 97%of the total latency of object detection at Tier 1, i.e., 132 ms out of 136 ms, is due to CMUcam processing (frame capture and frame differencing). Also, due its higher power requirements, CMUcam uses 92% of the energy, i.e., 153.8 mJ out of 167.2 mJ. In contrast, the Cyclops (refer Table 4) is much more energy efficient as compared to the CMUcam and consumes 33 mW for 892 ms, which is better than the CMUcam by a factor of 5.67 in terms of energy usage. However, the latency of detection at the Cyclops is around 900 ms, which is more than 6 times as much as the CMUcam. This latency number is an artifact of the current Cyclops hardware and can be reduced to around 200ms with optimizations expected in future revisions of the node. A breakup of the energy consumption of the Cyclops camera for detection is given in Table 4.

Tier 2: The processing tasks at Tier 2 of *SensEye* can be divided as: wakeup from suspend of the Stargate, stabilization after wakeup for program to start executing, camera initialization, frame grabber, vision algorithm for detection and recognition and finally the shutdown procedure for suspend, as shown in Table 5. The total latency



Mode	Latency (ms)	Current (mA)	Power (mW)	Energy Usage(mJ)
A: Object Detection	892	11	33	29.5
B: Idle	–	0.34	1	–

Table 4: *SensEye* Tier 1 (with Cyclops) latency breakup and energy usage.



Mode	Latency (ms)	Current (mA)	Power (mW)	Energy Usage(mJ)
A: Wakeup	366	201.6	1008	368.9
B: Wakeup Stabilization	924	251.2	1256.5	1161
C: Camera Initialization	1280	269.6	1348	1725.4
D: Frame Grabber	325	330.6	1653	537.2
E: Object Recognition	105	274.7	1373.5	144.2
F: Shutdown	1000	153.7	768.5	768.5
G: Suspend	–	3	15†	–

Table 5: *SensEye* Tier 2 Latency and Energy usage breakup. The total latency is 4 seconds and total energy usage is 4.71 J.

† This is measured on an optimized Stargate node with no peripherals attached.

at Tier 2 to complete all operations is 4 seconds. The largest delays are during camera initialization (1.28 s) and shutdown for suspend (1 s), with corresponding energy usages of 1725.4 mJ and 768.5 mJ. The least latency task is the algorithm used for object detection and recognition, which has a latency of 105 ms and the least energy usage of 144.2 mJ.

The comparison of energy consumption and latency reveals some of the benefits of using a two-tier rather than a single-tier camera sensor network. Every wakeup to shutdown cycle at Tier 2 consumes around 28 times as much energy as similar task at Tier 1 comprising of CMUcams. When the Tier 1 comprises of Cyclops cameras instead of CMUcams the ratio of energy usage is 142. There are two reasons for this large difference in energy consumption between tiers. First, the latency associated with Linux operating system wakeup from suspend state is significantly greater than the wakeup latency on a highly limited Mote platform that runs TinyOS. Second, the Stargate platform consumes significantly greater power than a Mote during the wakeup period. The net effect of greater latency and greater power consumption results in significantly greater total energy consumption for Tier 2.

5.1.2 Localization

As described in 3.2, localizing a detected object has several benefits. Localization at Tier 1 can be used to wakeup appropriate (e.g., nearest) Tier 2 nodes for further processing as well as compute approximate trajectory information for tracking and handoff purposes.

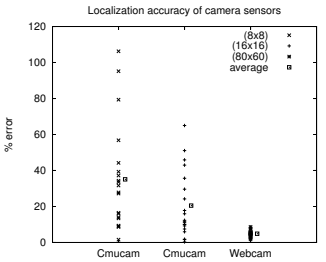


Figure 6: Localization accuracy of different camera sensors.

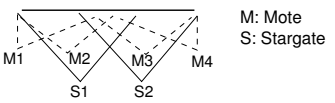


Figure 7: Placement of Tier 1 Motes and Tier 2 Stargates in *SensEye*.

Figure 6 is a scatter plot of 2D localization accuracy for objects using the CMUcam and the Webcam. The CMUcam uses 8×8 and 16×16 matrix representations of the captured image (converted from 88×143 and 176×255 pixels respectively) for frame differencing. This is representative of a typical centroid computation that we would expect on Cyclops nodes since these devices are resource-constrained both in memory and computation capability. The webcam uses a 80×60 representation calculated from a 320×240 pixels image. As seen from the figure, the webcam has the least localization error and the CMUcam using a 8×8 representation the largest error. The average error for each configuration is 35%, 20.5% and 4.85% respectively. The trends depicted in the figure indicate that if coarse location information is desired or suffices to wakeup higher tier nodes, Tier 1 based localization is sufficient. If accurate location information is required localization should be performed at the second tier of *SensEye*.

5.1.3 Object Recognition

To get an idea of the latency and power consumption of a recognition algorithm, we used a neural network based face recognition system [13]. The system is very constrained and uses face images of 960 pixels and a 960x40x1 neural network for learning. The system when executed on a Stargate to recognize faces had the following measurements: average latency 228 ms, average current draw 244.8 mA, average power consumption 1.23 W and average energy usage of 280.44 mJ. These measurements do not exactly reflect the increments of energy usage of *SensEye* as the face recognizer is not integrated into it. The measurements represent a crude estimate of the additions to latency and energy usage. We intend to replace the existing pixel averaging–based recognizer in *SensEye* with other sophisticated recognition algorithms in future work.

5.2 Comparison of *SensEye* with a Single-Tier Network

In this section we present an evaluation of the full *SensEye* system and compare it to a single-tier implementation of our algorithms. The comparison is along two axes—energy consumption and sensing reliability. Sensing reliability is defined as the fraction of objects that are accurately detected and recognized.

In our experiment, circular objects were projected onto a wall with an area of $3m \times 1.65m$. Objects appeared at random locations sequentially and stayed for a specified duration. Only one object was present in the viewable area at any time. Object appearances were interspersed with periods of no object being present in the viewable area. A set of four Motes, each connected to a CMUcam, constituted Tier 1 and two Stargates, each connected to a webcam, constituted Tier 2 of *SensEye*. Tier 1 Motes used a sampling period of 5 seconds and their start times were randomized. The object appearance time was set to 7 seconds and the interval between appearances was set to 30 seconds. The single–tier system consisted of the two Stargate nodes which were woken up every 5 seconds for object detection. This differs from *SensEye* where a Stargate is woken up only on a trigger from Tier 1. The nodes at both the tiers were placed in such a manner that each tier covered the entire viewable region as shown in Figure 7. The experiment used 50 object appearances for measuring the energy and reliability metrics.

Component	Total Wakeups	On Wakeup		Energy Usage (Joules)
		Object Found	No Object Found	
Stargate 1	311	32	279	1464.8
Stargate 2	310	42	268	1460.1

Table 6: Number of wakeups and energy usage of a Single–tier system. Total energy usage of both Stargates when awake is 2924.9 J. Total missed detections are 5.

Component	Total Wakeups	On Wakeup		Energy Usage (Joules)	Cyclops Expected Energy(J)
		Object Found	No Object Found		
Mote 1	304	15	289	50.7	8.96
Mote 2	304	23	281	50.7	8.96
Mote 3	304	27	277	50.7	8.96
Mote 4	304	10	294	50.7	8.96
Stargate 1	27	23	4	127.17	127.17
Stargate 2	29	25	4	136.59	136.59

Table 7: Number of wakeups and energy usage of each *SensEye* component. Total energy usage when components are awake with CMUcam is 466.8 J and with Cyclops is 299.6 J. Total missed detections are 8.

5.2.1 Energy Usage

We use two metrics to compare the energy usage between *SensEye* and the single–tier system, energy usage when awake and energy usage in suspend mode.

Tables 6 and 7 report the number of wakeups and details of detection at each component of the single–tier system and *SensEye* respectively. As can be seen from the tables, the Stargates of the single–tier system wakeup more often than the Stargates at Tier 2 of *SensEye*. A total of 621 wakeups occur in the single–tier system, whereas 58 wakeups occur at Tier 2 of *SensEye*. The higher number of wakeups with the single–tier are due the periodic sampling of the region to detect objects. Of out the total 621 wakeups, an object is detected only 74 times in the single-tier system whereas in *SensEye* Tier 1 performs initial detection and the Tier 2 Stargates are woken up fewer times— resulting in lower energy usage. The Tier 1 sensor nodes are cumulatively woken up 1216 times. The energy usage of *SensEye* during the experiment is 466.8 J, as compared to 2924.9 J by the single–tier node, a factor of 6.26 reduction. If the CMUcams in *SensEye* were replaced by Cyclops cameras, a factor of 9.75 reduction in energy usage is obtained.

As reported in [6], the Cyclops with Mote consumes 1 mW in its sleep state whereas an optimized Stargate consumes 15 mW in suspend mode. The CMUcam has a power consumption of 464 mW in

sleep mode and is highly unoptimized. Thus, in the suspend state, the Tier 2 node consumes more than an order of magnitude more power than the Tier 1 nodes with Cyclops cameras. For our experimental setting of 30 seconds of idle time between objects, this corresponds to an energy reduction by a factor of 33 for *SensEye*.

5.2.2 Sensing Reliability

Next we compare the reliability of detection and recognition of the two systems in the above described experimental setup. The single–tier system detected 45 out of the 50 object appearances and *SensEye* detected 42—a 6% decrease in sensing reliability. The result shows the efficacy of using *SensEye* instead of a single-tier network, as *SensEye* provides similar detection performance (6% more missed detections) at an order of magnitude less energy requirements.

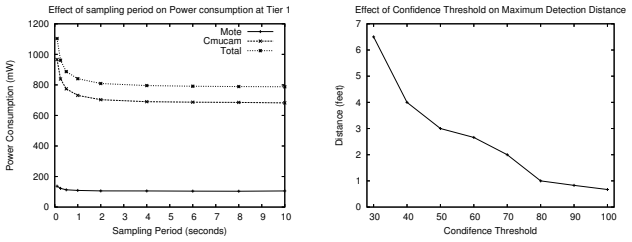
The sensing reliability of *SensEye* is dependent on the time for which an object is in view, the sampling period at Tier 1 and speed of the object if it is moving. Since increasing sampling period is same as increasing time for which object in view, we study the effect of different times for which object is view on sensing reliability. Figure 8(a) plots the fraction of undetected objects with object in–view timings of 5,7 and 9 seconds. As seen from the figure, when an object is in view for 5 seconds, 52% objects are not detected. With a time of 9 seconds for each object to be in view, the percentage drops to zero. A timing of 7 seconds yields an intermediate value of 16% undetected objects.

To study the effect of speed of moving objects on sensing reliability, we conducted an experiment where objects moved across the viewable area. The object started from a random point on one side of the rectangular area and exited from another random point on the other side. The sampling period used at the Tier 1 nodes was 5 seconds. Figure 8(b) plots the percentage of undetected objects at different speeds of the moving object. As can be seen, at the slowest considered speed of 0.2 m/s, a sampling rate of 5 seconds is able to detect all objects atleast once. A speed of 0.6 m/s results in 62% undetected objects. The trend shown is intuitive, given a sampling rate, higher speeds lead to higher undetected objects. Based on the desired probability of detection, the plots can be used to choose sampling rates for different object movement speeds.

5.2.3 Coverage

A key benefit of *SensEye* is that more sensing elements can be placed at lower energy cost than a single-tier architecture. This gives *SensEye* greater spatial redundancy between nodes and benefits both the latency of object detection as well as the accuracy of localization. We now look at the overlapping coverage provided by the Tier 1 nodes and the Tier 2 nodes (also the nodes of the single–tier network). Figure 8(c) plots, for each component, the cases when only a single node covered and detected a object. As can be seen, the coverage of Mote2 and Mote 3, which were centrally placed, shared a lot of area with the other Motes. Hence these nodes are woken up most and also have the most redundant wakeups as compared to Mote1 and Mote4, which were placed at the corners. The Tier 2 Stargates also have a small overlapping region and are woken up a small fraction of times redundantly. Based on the coverage and overlapping nodes woken up for detection, 54% objects can be localized in *SensEye* whereas 36% can be localized in a single-tier network comprising only the Stargate nodes. This metric of coverage can be used to guide further node placements to reduce or increase redundancy, in order to minimize energy usage or increase localization opportunities respectively.

5.2.4 Coverage with Tier 3 Retargetable Cameras



(a) Effect of sampling period (b) Effect of confidence threshold

Figure 9: Sensitivity to *SensEye*system parameters.

To test the coverage and retargatable feature of the Tier 3 PTZ cameras, we measure the number of times a Tier 3 node successfully views an object after its pan and tilt movements. The experimental setup had 40% overlapping coverage among Tier 1 nodes and the PTZ camera could view at most a quarter of the total coverage area at any time. When an object was detected by more than one Tier 1 node, previously described 3D localization techniques were used to calculate the pan and tilt values and retarget the Tier 3 camera. Out of the 50 object appearances, the PTZ camera could view 46—a 92% success rate. The experiment verifies that 3D localization techniques along with retargetable cameras have a high success rate and are useful to improve coverage.

5.2.5 Sensitivity to System Parameters

SensEye has several tunable parameters which effect energy usage and sensing reliability. In this section, we explore the sensitivity to two important system parameters, sampling rate and camera detection threshold.

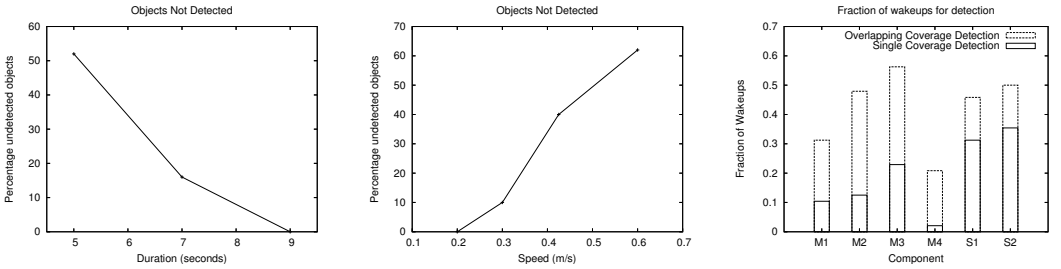
The power consumption at Tier 1 is a function of the sampling period used to probe the CMUcam and check for object detections. Figure 9(a) plots the power consumption at a Mote with increasing values of sampling period. The sampling period is varied from 100 ms to 10 seconds and the power consumption at these two ends is 137 mW and 105.7 mW respectively. While the power consumption reduces with increasing sampling period as expected, it quickly plateaus since the large sleep power consumption of the CMUcam dominates at lower sampling periods.

From a sensing reliability perspective, each Mote uses a confidence threshold value to compare with the confidence with which a CMUcam reports a detection. The threshold determines when triggers are sent to Tier 2. A higher threshold means closer objects will be detected more easily than farther objects and a lower threshold can more easily detect objects at larger distances. The trend is verified by the plot shown in Figure 9(b). We varied the confidence threshold from 30 to 100 and measured to maximum distance at which objects are flagged as detected and its trigger sent to Tier 2. As can be seen in the figure, a threshold of 30 can detect objects till a distance of 6.5 feet and with thresholds greater than 80 the maximum distance drops to less than 1 feet. Choosing a good threshold is important since it controls the false positives and false negatives, and hence the energy consumption and reliability of the system.

6. RELATED WORK

SensEye draws upon numerous efforts in camera sensors, power management, sensor placement and surveillance, which we review here.

Multimedia Sensor Networks: Several studies have focused on single-tier camera sensor networks. Panoptes [22] is an example of a video sensor node built using a Intel StrongARM PDA platform



(a) Undetected objects. (b) Undetected moving objects. (c) Coverage Redundancy.

Figure 8: *SensEye* sensing reliability and coverage.

with a Logitech Webcam as the vision sensor. The node uses the 802.11 wireless interface and can be used to setup a video–based monitoring network. Panoptes is an instance of a single-tier sensor network and is not a multi-tier network like *SensEye*. A Tier 2 node of *SensEye* is similar to Panoptes, with additional support for network wakeups and optimized wakeup-from-suspend energy saving capability. Panoptes also incorporates compression, filtering and buffering and adaptation mechanisms for the video stream and can be used by Tier 2 nodes of *SensEye*. Other types of multimedia sensors, like audio sensors [21], have also been used for calibration and localization applications.

Video Surveillance: A distributed video surveillance sensor network is described in [12]. The video sensor network is used to solve the problem of attention to events in presence of limited computation, bandwidth and several event occurrences. The system implements processing at cameras to filter out uninteresting and redundant events and tracks abnormal movements. The CVSN Project [2] focuses on developing distributed vision processing techniques for counting the number of people in an area.

Another example of a single-tier video surveillance and monitoring system is VASM [16]. The main objective of the system is to use multiple, cooperative video sensors for continuous tracking and coverage. The system develops sophisticated techniques for target detection, classification and tracking and also a central control unit to arbitrate sensors to tracking tasks. A framework for single-tier multi-camera surveillance is presented in [11]. The emphasis of the study is efficient tracking using multi-source spatio-temporal data fusion, hierarchical description and representation of events and learning-based classification. The system uses a hierarchical master-slave configuration, where each slave camera station tracks local movements and relays information to the master for fusion and global representation. While our general aim is to build similar systems, we focus on systems, networking and performance issues in a multi-tier network using video surveillance as an application. The vision algorithms and cooperation techniques of the above systems can extend capabilities of *SensEye*.

Sensor Placement: An important criteria of sensor networks is placement and coverage. Single tier placement of cameras is studied in [20]. The paper solves the problem of efficient placement of cameras given an area to be covered to meet task–specific constraints. This method provides solutions for the single–tier placement problem and is useful to place each tier of *SensEye* independently. Some of these techniques apply to placement of nodes in *SensEye* but need to be extended for multi–tier settings.

Power management: Power management schemes, like wake–on-wireless [8] and Turducken [18], are techniques to efficiently use the limited battery power and thus extend lifetime of sensor platforms. The wake–on–wireless solution uses a incoming call to wakeup the PDA and reduces power consumption by shutting down the PDA when not in use. Turducken uses a combination of devices, a laptop, a Stargate and a mote, and uses lower subsystems to reduce power consumption and wakes up the more power hungry devices only when required. The *SensEye* Tier 2 node is optimized using both the above solutions.

7. CONCLUSIONS

In this paper, we argued about the benefits of using a multi-tier camera sensor network over single tier networks and presented the design and implementation of *SensEye*, a multi-tier camera sensor network. Using a implementation of a surveillance application on *SensEye* and extensive experiments, we demonstrated that a multi-tier network can achieve an order of magnitude reduction in energy usage when compared to a single-tier network, without sacrificing reliability.

As part of future work, we plan to study placement and self-calibration techniques for camera sensor networks.

8. REFERENCES

[1] J. M. Kahn and R. H. Katz and K. S. J. Pister. Mobile Networking for "Smart Dust". In *Fifth International Conference on Mobile Computing and Networking*, 1999.

[2] A. Gamal and L. J. Guibas. Collaborative visual sensor networks. http://mediax.stanford.edu/projects/cvsn.html.

[3] Anonymous. SensEye: A Multi-tier Camera Sensor Network. Technical report, Anonymous Institution, 2005.

[4] Sony SNC-RZ30N Camera Driver. http://cvs.nesl.ucla.edu/cvs/viewcvs.cgi/CoordinatedActuation/Actuate/cameradriver/.

[5] The CMUcam2. http://www-2.cs.cmu.edu/ cmucam/cmucam2/index.html.

[6] Cyclops. http://www.cens.ucla.edu/˜mhr/cyclops/cyclops.pdf.

[7] D A. Forsyth and J Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[8] E. Shih and P. Bahl and M. Sinclair. Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices. In *Eighth ACM International Conference on Mobile Computing and Networking*, 2002.

[9] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, Boston, MA, 2002.

[10] J. Hill and R. Szewczyk and A. Woo and S. Hollar and D. Culler and Kr. Pister. System architecture directions for network sensors. In *ASPLOS*, 2000.

[11] L.Jiao and Y. Wu and G. Wu and E. Y. Chang and Y. F. Wang. The Anatomy of a Multi-camera Security Surveillance System. *ACM Multimedia System Journal Special Issue*, October 2004.

[12] M. Chu and J. E. Reich and F. Zhao. Distributed Attention for Large Video Sensor Networks. In *Intelligent Distributed Surveillance Systems*, 2004.

[13] T. Mitchell. *Machine Learning*. Mc–Graw Hill, 1997.

[14] Crossbow wireless sensor platform. http://www.xbow.com/Products/Wireless_Sensor_Networks.htm.

[15] N B. Priyantha and A. Chakraborty and H. Balakrishnan. The Cricket Location-Support System. In *Sixth International Conference on Mobile Computing and Networking*, 2000.

[16] R. Collins and A Lipton and T. Kanade. A System for Video Surveillance and Monitoring. In *American Nuclear Society (ANS) Eighth International Topical*

Meeting on Robotics and Remote Systems, 1999.

[17] A. Rosenfeld and J. L. Pfaltz. Sequential Operations in Digital Picture Processing. *J. ACM*, 13(4):471–494, 1966.

[18] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins. Turducken: Hierarchical Power Management for Mobile Devices. In *The Third International Conference on Mobile Systems, Applications, and Services*, 2005.

[19] Stargate platform. http://www.xbow.com/Products/XScale.htm.

[20] U.M. Erdem and S. Sclaroff. Optimal Placement of Cameras in Floorplans to Satisfy Task Requirements and Cost Constraints. In *OMNIVIS Workshop*, 2004.

[21] V.C. Raykar, I. Kozintsev and R. Lienhart. Position Calibration of Audio Sensors and Actuators in a Distributed Computing Platform. In *ACM Multimedia*, 2003.

[22] W. Feng and B. Code and E. Kaiser and M. Shea and W. Feng and L. Bavoil. Panoptes: A Scalable Architecture for Video Sensor Networking Applications. In *ACM Multimedia*, 2003.

[23] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich. Collaborative signal and information processing: An information directed approach. *Proceedings of the IEEE*, 91(8):1199–1209, 2003.