SCOPES: Smart Cameras Object Position Estimation System

Ankur Kamthe, Lun Jiang, Matthew Dudys, and Alberto Cerpa

Electrical Engineering and Computer Science
University of California, Merced CA 95343, USA
{akamthe,ljiang2,mdudys,acerpa}@ucmerced.edu

Abstract. Wireless camera sensor networks have to balance the conflicting challenges imposed by the detection performance, latency and lifetime requirements in surveillance applications. While previous studies for camera sensor networks have addressed these issues separately, they have not quantified the trade-offs between these requirements. In this paper, we discuss the design and implementation of SCOPES, a distributed Smart Camera Object Position Estimation System that balances the trade-offs associated with camera sensor networks. The main contribution of the paper is the extensive evaluation of parameters affecting the performance of the system through analysis, simulation and experimentation in real-life conditions. Our results demonstrates the effectiveness of SCOPES, which achieves detection probabilities ranging from 84% to 98% and detection latencies from 10 seconds to 18 seconds. Moreover, by using coordination schemes, the detection performance of SCOPES was improved with increased system lifetime. SCOPES highlights that intelligent system design can compensate for resource-constrained hardware and computationally simple data processing algorithms.

1 Introduction

In the past decade, the day-to-day life of every human has been invaded by a plethora of sensors. Networks comprised of large numbers of these sensors have been deployed to gather data from the surrounding environment. From simple inexpensive photo sensors to complex camera sensors, the acquisition of data has grown easier but has imposed a greater challenge on the data-processing side. In general, the data processing software is comprised of image processing algorithms for feature extraction and interpretation. Techniques like Viola-Jones [1] build a classifier from a large set of potential features to detect faces with a very high detection rate and provide results in real-time on platforms with high processing power (384x288 pixel images at 15fps on a 700MHz CPU). However, an implementation of the Viola-Jones algorithm provided with the CMUcam3 platform (60MHz with no floating point capability) takes about 5-6 seconds to detect faces [2]. Due to high computational complexity, these techniques do not find widespread adoption in resource-limited wireless sensor networks.

Wireless sensor network-based detection and tracking systems use either computationally lightweight algorithms when performing in-node processing of data

U. Roedig and C.J. Sreenan (Eds.): EWSN 2009, LNCS 5432, pp. 279–295, 2009.

or data aggregation techniques when transmitting raw data for centralized processing. Aslam et al. [3] describe a centralized framework for tracking a moving object by using a particle-filter for a binary sensor network. VigilNet [4] is a detection and classification system for tracking metallic objects using information from multiple magnetometer and acoustic sensors. In such approaches, the use of simple sensors for tracking entails dense deployment to localize an object or to infer its direction information.

In contrast, using camera sensor networks, we can infer position and direction information from a single sensor by identifying an object and tracking its position in successive images, respectively. This reduces the density of nodes required to gather data while improving the data fidelity manyfold. However, camera sensors are not without their caveats; namely, network lifetime, latency and detection performance. The power consumption of wireless radios transmitting high bandwidth image/video data to a central base station would result in rapid decrease in the lifetime of the sensor mote, in addition to causing network congestion. In-node computations process the raw data into summarized meta-data, which reduces the amount of data to be transmitted. However, this introduces latency because of limitations in the computational capabilities of sensor nodes. This necessitates, to reduce latency, the usage of lightweight processing algorithms and, to avoid missed detections, the redistribution of sensing over multiple sensors. At the same time, the low complexity of the data processing techniques should not compromise the detection requirements of the application.

Our goal is to improve the performance of wireless sensor network-based surveillance systems with synergy between the underlying hardware and software infrastructure. The paper aims to show that SCOPES can achieve comparable, if not better, detection performance by intelligent utilization of resources and computationally lightweight algorithms in a resource-constrained environment, while ensuring long lifetimes. Previous studies [5,6,7] have addressed algorithmic issues for camera sensor networks but provide limited performance evaluation (see Section 5). In this paper, we do not concentrate on the development of new image processing approaches for camera sensor networks. Instead, we provide a much more complete and extensive performance evaluation in a real-world environment and design and implement solutions to the network lifetime, latency and detection quality issues affecting the performance of any camera sensor network. The design of SCOPES incorporates simple but fast image processing algorithms for background update and object recognition, thereby reducing the amount of data transmitted and the processing latency. The redundancy in sensor deployment is harnessed to implement a distributed coordination algorithm to handle node failures. SCOPES was evaluated in real world conditions, with results being compiled from data of a real-life deployment of a camera sensor network system for counting people in different sections of a building. We present results quantifying the trade-offs between data processing latency, memory usage, power consumption and detection performance for our system. To complete the discussion, the paper provides a comparison of SCOPES with closely related works in wireless camera sensor networks for surveillance purposes (see Section 5).

2 System Description

2.1 Hardware and Software Infrastructure

Our SCOPES implementation comprises of an Cyclops camera [8] interfaced with a Moteiv Tmote Sky module via an intermediate adapter board. The Cyclops consists of an ADCM-1700 imager, 512KB of external SRAM and a 4MHz ATmega128L micro-controller (MCU). As the MCU can address a maximum of 64KB of memory, the external SRAM is divided into eight, 64KB memory banks ($nBanks$, from Table 3). The Cyclops captures 10 64x64 pixel grayscale images per bank (i.e., 80 total). It performs local detection and processing of the iamges and sends summarized data (see Section 2.2: Object Recognition) to the Tmote module which routes it to the base station using multihop communication. The Cyclops and the Tmote run the TinyOS operating system (see Figure 1).

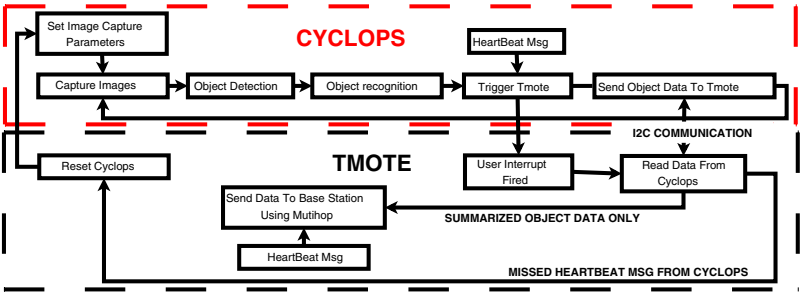


Fig. 1. Software Block Diagram of SCOPES: the arrows indicate the logical sequence of operations and interactions between the two devices

2.2 Algorithms

Object Detection (OD): The goal of object detection is to determine the presence of an object in the image foreground, if any, and to update the background. In order to achieve low processing latency, a modified background subtraction algorithm is implemented for object detection. After background subtraction, we use two preset thresholds (OBJECT-THRES = 40 and SHADOW-THRES = 15) to assign a label (OBJECT, SHADOW or BG) to each pixel (see Figure 2). Depending on the label assignment, the value of the corresponding background pixel is updated using an Exponentially Weighted Moving Average (EWMA). In the EWMA, we give less weight to the value of the current pixel if it is classified as an object or shadow, as these changes to the background are less likely to be of a long term nature. The weights are preset based on the image capture speed of the camera (see Table 1), the area covered by the camera 2.4m x 2.4m) and the speed of objects (approx. 1.2 m/s). In the current implementation, an object would need to be immobile for atleast 5s before being classified as a background pixel. This way, temporal background changes will be assimilated into the background over time. When the number of pixels labelled as OBJECT

```
count = 0
for (each pixel i in current image) do
    delta = | img(i) - bg(i) |
    if (delta ≥ OBJECT-THRESH)
        pixel(i) = OBJECT
        bg(i) = 0.99 * bg(i) + 0.01 * img(i)
        count++
    else if (delta ≥ SHADOW-THRESH)
        pixel(i) = SHADOW
        bg(i) = 0.95 * bg(i) + 0.05 * img(i)
    else
        pixel(i) = BG
        bg(i) = 0.85 * bg(i) + 0.15 * img(i)
```

Fig. 2. Pseudo-code for background subtraction and update

a b c

d x

Fig. 3. Grouping Pixels

Table 1. Time (in second) for executing various image processing operations $nFrames$ images at a time on the Cyclops (Note: averaged over 100 cycles)

Cyclops Action	nFrames 1	5	10
Image Capture (IC)	0.68	1.43	2.51
IC w/ OD	0.7	1.54	2.74
IC w/ (OD + OR)	1.3	4.71	9.50

Table 2. Basic algorithmic rules for group assignments for pixel x. (Note:− indicates group id is not assigned.)

Pixel b	Pixel d	Action
-	-	new group id
G_1	-	group G_1
-	G_1	group G_1
G_1	G_2	group G_1 (merge G_2 in G_1)

exceed a preset threshold, an object detection is signalled. In the case of an object detection, pixels labelled SHADOW are relabelled as OBJECT if they have at least one neighbor that is an OBJECT pixel.

Object Recognition (OR): In this step, we try to group all pixels labelled as OBJECT by raster scanning the image starting from the top left corner. A pixel is considered to be part of an object if its left, top, top-left and top-right neighbors are labelled OBJECT. For example, in Figure 3, let x be the pixel in consideration and a, b, c and d are its top-left, top, top-right and left neighbors, respectively. If x, a, b, c and d are all labelled OBJECT, then x is considered to be part of an object. Assigning group id's is done using the rules explained in Table 2. Small groups in close proximity of each other are merged to account for fragmentation of a big object. For each object, information regarding the centroid (x and y coordinates), the number of pixels and the number of consecutive frames in which the object is detected, is maintained.

Direction Inference: In SCOPES, the nodes were deployed in hallways to detect the transitions of people between different sections of a building floorplan (see Section 3.3). This entailed determining movement of people in only two directions. In order to infer direction, objects in successive frames are matched according to their size (in pixels). Any object in the current frame that cannot be matched to another object in the previous frame is stored as a new object. Information for objects from previous frames that disappear or cannot be matched to objects in current frame are saved into a data structure.

Table 3. Notations used in the paper with the associated meanings

Term	Explanation
$nFrames$	number of images captured consecutively
$nBanks$	number of memory banks
$T_S^{nFrames}$	cyclops IC time for $nFrames$ images
T_S	total cyclops IC time (camera ON), depends upon $nBanks$
T_{OD}	(avg.)object detection time for $nBanks \times nFrames$ images
T_{OR}	(avg.) object recognition time per image
P	power consumption per node
DP	detection probability per node
DFP	detection failure prob. per node, $1 - DP$

Information regarding the original position, displacement and number of consecutive frames is maintained for each object that appears across successive images in the current memory bank. After processing all the images in the current bank, an array of data structures containing information on a maximum of four objects is transferred to the base station via the Tmote.

Density Estimation Algorithm: On the base station, the packets coming from the various nodes are deconstructed. Messages are classified by source (node id) and time of occurrence. From the object data in the Cyclops payload, we can infer the direction of motion from the initial position and the displacement vector. Since, we have prior information about the deployment of the nodes, counting the transitions of objects enables the base station to compute the distribution of people in different sections of the building over time (assuming some initial distribution). Objects with no direction information are filtered out.

3 Performance Evaluation of SCOPES

In this section, we present results quantifying the trade-offs between data processing latency, memory usage, power consumption and detection performance. Table 3 shows the notations for the parameters used in the discussion sections.

3.1 Objective Functions

Our goal for SCOPES was to function as a surveillance system to monitor the occupancy of indoor environments such as office buildings. Some metrics and objective functions of interest are as follows:

Global/Local Density Estimate: How accurately can we estimate the occupancy of each section and of the total area covered?

Power Consumption: What is the system lifetime when powered by batteries?

Memory Usage: How much memory is required to achieve acceptable performance? How is the performance affected by memory size?

Detection Latency: How long does the system take to report data?

Detection Probability: How good is the estimate of the movement of people across different sections in the building?

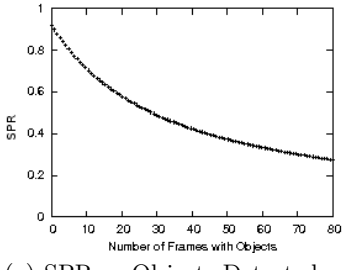
3.2 Simulation and Analysis

Since people passing under a camera can be regarded as a Poisson process, we model their inter-arrival times with an exponential distribution as follows:

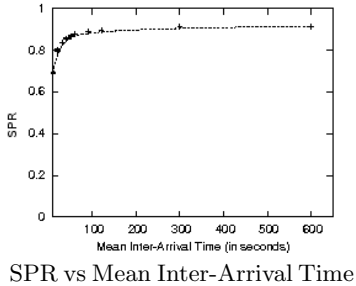
$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} , & x > 0 \\ 0 , & x < 0 \end{cases}$$

where $\frac{1}{\lambda}$ is the mean inter-arrival time for an event. The number of frames in which an object appears is modeled by a uniform distribution (min=2;max=10) to account for variation in speed of people. We simulate the operation of a SCOPES node in the GNU R statistical computing package.

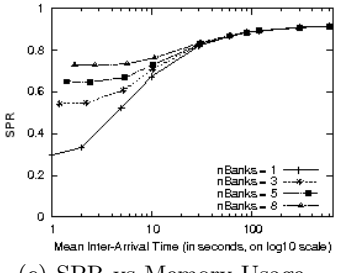
Sensing-Processing Ratio (SPR): In SCOPES, the camera is not operated in trigger-driven or schedule-driven modes discussed in previous studies [7]. Instead, each camera is either in active period, capturing and processing images, or in idle period, wait interval between successive active periods. *In this paper, we refer to the ratio of time taken to capture images by the camera and the total active period i.e., the sum of the image capture and processing times, as sensing-processing ratio.* In our discussions, the sensing-processing ratio is the penalty incurred by the system as a result of the data processing latency. A high sensing-processing ratio is an indicator of lower data processing latency and vice-versa.



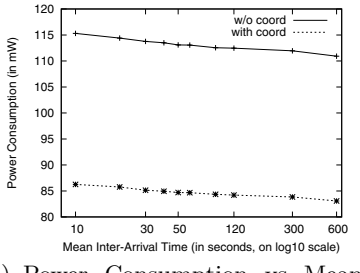
(a) SPR vs Objects Detected



(b) SPR vs Mean Inter-Arrival Time



(c) SPR vs Memory Usage



(d) Power Consumption vs Mean Inter-Arrival Time

Fig. 4. (a) shows the variation in SPR of a node as a function of the number of image frames containing an object. (b) shows the SPR as a function of the mean inter-arrival time. (c) shows the change in SPR as a function of the mean inter-arrival times for different memory usages. (d) shows Power Consumption as function of the mean inter-arrival times.

There are two main reasons for operating the camera as described above; first, the camera hardware in current sensor networks does not allow concurrent image capture and processing of data and second, the object recognition algorithm introduces long latencies between successive image capture periods. Understanding sensing-processing ratio is important, since it affects many of our objective functions, including global/local position estimation, power consumption, detection probability and detection latency.

In Figure 4(a), we observe the variation in SPR of a node with respect to the total number of images N in which an object (person) is detected. This relationship can be expressed as follows:

$$SPR = \frac{T_S}{T_S + T_{OD} + N \times T_{OR}} \tag{1}$$

Background subtraction needs to be performed for all the images resulting in a fixed cost T_{OD}. The object recognition function needs to be executed on only the N frames in which an object is detected. As object recognition incurs the highest processing cost (T_{OR}, see Table 1), the SPR of a node is affected by the time taken for object recognition in each image. During the image processing latency period, an object passing beneath the camera will be missed as the Cyclops is not capable of simultaneously capturing and processing images. Thus, a low SPR will result in lower detection probability.

Figure 4(b) shows the variation in SPR with respect to the mean inter-arrival time $\frac{1}{\lambda}$ of an object detected by the camera with fixed amount of memory ($nBanks = 8$ and $nFrames = 10$, refer Section 2.1 for cause of $nBanks$). This relationship can be expressed as follows:

$$SPR = \frac{nT_S}{n(T_S + T_{OD}) + \sum_{i=1}^{n} \alpha_i \beta_i T_{OR}} \tag{2}$$

where n is the number of times we capture a set of 80 images, α_i is the number of times an object is detected and β_i is the average number of frames occupied by the object in the current (i^{th}) set of images. When the mean inter-arrival time is low, more objects are detected and the camera spends a longer time processing the image data, leading to a low SPR. Hence, longer data processing time leads to lower detection probability as the camera cannot capture images during that period.As the inter-arrival time increases, the SPR increases because the relative proportion of image processing time decreases.

In Figure 4(c), we observe the variation in SPR with respect to the mean inter-arrival time as a function of memory usage (varying $nBanks$). The amount of available memory dictates the space available to store images captured in time T_S ($=nBanks \times T_S^{nFrames}$). This relationship can be expressed as:

$$SPR = \frac{n(nBanksT_S^{nFrames})}{n(nBanksT_S^{nFrames} + T_{OD}) + \sum_{i=1}^{n} \alpha_i \beta_i T_{OR}} \tag{3}$$

As the mean inter-arrival times varies from low to high, the SPR increases with memory usage since a node captures more images while spending a lower percentage of its time processing the image data. Hence, in general, more memory

leads to a better SPR. For long mean inter-arrival times, the amount of memory does not have a significant effect on the SPR of a node.

Power Consumption (P): The lifetime of battery powered sensor nodes is directly affected by the power consumption of the system. The power consumed by the Cyclops and the Tmote Sky in different modes of operation is given in Table 4. The relationship between the power consumption and the different modes of operation of the node can be expressed as follows:

$$P = P^{sensing}_{cyclops} + P^{proc}_{cyclops} + P^{sleep}_{cyclops} + P^{RX}_{tmote} + P^{TX}_{tmote} \quad (4)$$

We analyse the power consumption under two different scenarios for node operation: (i) without coordination (multiple nodes sense the area at the same time) and (ii) with coordination (multiple nodes sense the area in non-overlapping intervals of time, see Section 4 for node coordination scheme details). Figure 4(d) shows the variation in power consumption for each of 3 nodes deployed to sense an area as a function of the mean inter-arrival time of an object.

For case (i), the Cyclops on each of the nodes is capturing and processing data all the time, i.e., there are no idle periods (Sleep mode). The power consumed by the Cyclops is slightly higher when it is processing image data stored in the external SRAM as compared to the power consumed in Image Capture mode (refer Table 4). This leads to higher power consumption when the inter-arrival time is low as the Cyclops spends a higher proportion of its active time processing data. For case (ii), with sensing coordination between the three nodes, the power consumed by each node is significantly lower than in the first case as the Cyclops has idle periods while waiting for its turn to sense the area.

Detection Failure Probability (DFP): Detection failures occur in the form of false negatives and false positives. However, we define Detection Failure Probability (DFP) as the probability that none of the camera nodes covering a section report the presence of a person passing under the camera. DFP quantifies the effect of *only false negatives* on our system. False negatives mainly

Table 4. Power Consumption of the Cyclops and Tmote Sky Modules. (For more details, refer to [8] and the Tmote Sky data sheet.)

Device Operation	Notation	Power
Cyclops		
- Image Capture	$P^{sensing}_{cyclops}$	42mW
- Extended Memory Access	$P^{proc}_{cyclops}$	51.5mW
- Sleep	$P^{sleep}_{cyclops}$	0.7mW
Tmote		
- MCU + Radio RX	P^{RX}_{tmote}	65.4mW
- MCU + Radio TX (0dBm)	P^{RX}_{tmote}	58.3mW

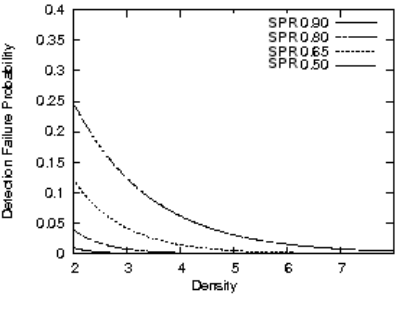
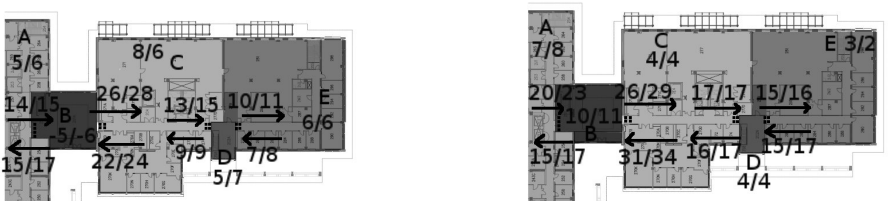


Fig. 5. Detection Failure Probability as a function of density of nodes covering the same area and SPR



(a) Morning Map (b) Noon Map

Fig. 6. Occupancy and Transition Maps. The arrows indicate the direction of motion. The different sections of the floorplan are shaded and labelled with different alphabets. The numbers indicate counts from SCOPES (left) and from ground truth (right).

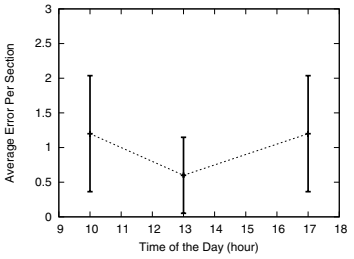
depends on the SPR and to a lesser extent on the static thresholds in the object detection algorithm. Detection failures, due to static thresholds, are difficult to simulate as they might not provide an accurate representation of real-life conditions. We only analyse the relationship between DFP and multiple nodes n sensing an area at the same time (sensing without coordination) as a function of varying SPR. This relationship can be expressed as:

$$DFP = (1 - SPR)^n \tag{5}$$

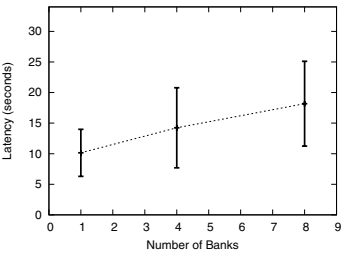
Figure 5 shows that the DFP decreases with higher SPR and number of nodes n. Since, the worst case processing time is bounded because of memory constraints, the SPR cannot fall below a certain number. Having coordination among the nodes will improve detection failure probability by eliminating the chances of a missed detection due to SPR. However, there will still be some missed detections because of the deficiencies of the hardware and software in the underlying platform (See Section 4.3).

3.3 Experimental Deployment

We deployed 16 nodes on the ceiling of the corridors in an office building. The deployment of nodes was done in this fashion to reduce privacy concerns of individuals by sensing in the common areas of the floorplan. The floorplan was separated into 5 sections by deploying the nodes in groups at transition points. *In each group, multiple nodes sense the same area at the same time, i.e., operating without coordination.* For collecting the ground truth, we installed two Panasonic KX-HCM280A network web cameras to record the movement of people. They are capable of capturing 10 frames per second (fps) at a resolution of 640×480 pixels. These images are timestamped using an NTP synchronized machine. The ground truth data is processed using haar cascades implemented in the OpenCV library [9] to provide a list of images in which a human being is detected. We manually corrected the OpenCV output for the false positives and false negatives in the processed ground truth. For computing detection probability and latency, we compare the manually processed and corrected ground truth data with the



(a) Position Estimation Error vs Time of the Day



(b) Detection Latency vs Memory Usage

Fig. 7. Fig. 7(a) shows the Average Position Estimation Error (number of people) per section for different times of the day. Fig. 7(b) shows the Detection Latency as a function of the number of memory banks.

data collected from the SCOPES logs. The following is a list of experiments that we conducted for the performance evaluation:

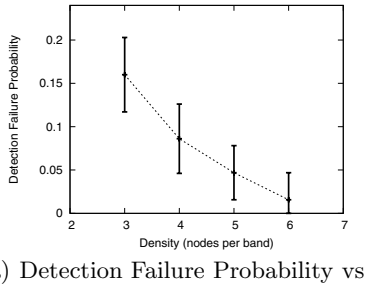
Occupancy and Flow Estimation: 4 groups of nodes with 4 nodes in each group were deployed ($nBanks = 8$, $nFrames = 10$). The experiment was conducted twice for different two-hour periods of the day

Memory Usage and Detection Latency: 3 groups of nodes with 4 nodes in each group were deployed for each value of $nBanks$ ($nFrames = 10$).

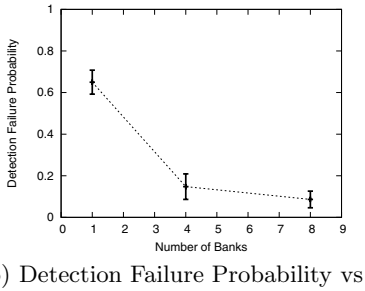
Detection Probability: experiment repeated thrice with 3, 4, 5 and 6 nodes deployed in each group ($nBanks = 8$, $nFrames = 10$).

3.4 Experimental Results

Density Estimation: The first aspect we address is the evaluation of SCOPES for building density estimation maps of area occupancy by counting the



(a) Detection Failure Probability vs Density of Nodes



(b) Detection Failure Probability vs Memory Usage

Fig. 8. Fig. 8(a) shows Detection Failure Probability as a function of the density of nodes covering the same area. Fig. 8(b) shows the Detection Failure Probability as a function of the number of memory banks.

transitions across the different sections. Figure 6 shows the occupancy and transition estimation maps created from data acquired from individual SCOPES experiments. From our results, we see that, we are able to track the movement of people over more than 73 sq. meters of the Engineering Building with a small, reasonable error. Figure 7(a) shows the average density estimation error for all the sections at different times of the day. Since, the error bars are overlapping with the mean values, we can say that there is no statistically significant change in average error which remains bounded (under 2) at different times of the day. The result shows that with suitable number of nodes and deployment location, embedded camera sensor networks such as SCOPES can provide adequate performance for density estimation purposes in real-world scenarios.

Detection Latency: Detection Latency is the time it takes for a node to report a person transitioning among different areas to the base station. In the Cyclops, the 512KB of available memory is partitioned into eight 64KB banks. Each node first captures $nBanks \times nFrames$ images and then it starts processing the image data in each bank. When the Cyclops is able to infer direction for an object from images in a certain bank, it will transfer the summarized object data to the Tmote for that bank. The Tmote routes the data to the base station via the radio. In our experiments, the base station was located 2 hops away from the farthest group of nodes. Figure 7(b) shows the variation in detection latency as a function of the number of memory banks used for storing images. In Figure 7(b), we observe that the detection latency is directly proportional to the amount of memory utilized for storing images. The detection latency is 10 seconds when $nBanks = 1$. It increases to 18 seconds when $nBanks = 8$. As the amount of available memory ($nBanks$) increases, the Cyclops can capture a higher number of images before processing the image data, resulting in longer detection latency. This shows for camera sensor networks that capture sequences of images before processing them, storing more image data can increase the detection latency, which could adversely affect the responsiveness of the surveillance system.

Detection Failure Probability (DFP): Figure 8(a) shows the variation in DFP i.e., false negatives as a function of the number of nodes used to cover an area. In general, deploying more nodes improves the DFP which agrees with our simulation results (see Figure 5). However, beyond 6 nodes we do not see an improvement in DFP because of the limited capabilities of our nodes. In Figure 8(b), we see the variation in DFP under memory constraints. In these experiments, we vary the number of memory banks used for storing image data. Figure 8(b) shows that DFP gets significantly reduced as we increase the number of memory banks. From Figures 8(b) and 4(c), we confirm that as the numbers of memory banks increases, SPR increases, leading to lower DFP.

Our experimental evaluation highlights and quantifies the trade-off between detection latency and detection performance as a function of memory usage and node density. For camera sensor networks like SCOPES, increasing the number of nodes would keep the detection latency low and improve detection performance at the expense of increased deployment cost. Also, by deploying sufficient number

of nodes to cope with worst case SPR, we can enable lower detection latency and hence, a more responsive system.

4 Improving SCOPES Using Node Coordination

To ameliorate the effects of concurrent data processing latency periods for nodes working in a uncoordinated manner, we decided to implement a scheduling and coordinated sensing scheme. The goal of the scheme is to improve the performance of the existing system by reducing the detection failure as well to decreasing the power consumption of the nodes (refer Section 3.2). The design requirements for our node coordination scheme are two-fold: (1) Nodes covering the same area or nodes in close proximity should provide near-continuous sensing coverage for the area, and (2) Nodes should provide near-continuous sensing coverage for an area even if some nodes stop functioning.

4.1 Clustering Algorithm

The clustering algorithm (see Figure 9) executes periodically once every hour. At the beginning, the nodes change their RF power level to reduce the transmission distance ($RF2$). Each node then starts a one shot timer (Timer1) with a random interval up to a maximum of T1 seconds. After the Timer1 fires, a node sends a $GROUP\_ASSOC$ message declaring itself as the group head. It then starts Timer2 with an interval of T2 seconds. All nodes that are within close proximity of the group head respond by sending a $GROUP\_INFORM\_CH$ message. After Timer2 fires, the group head broadcasts a message containing information regarding the group head and the associated group members. Since, these radio messages do not propagate beyond a certain distance, we ensure that nodes in

```
INITIALIZE()
    Change RF Power Level to RF2.
    Set Timer1 to fire after a random interval
TIMER1.FIRED()
    If no GROUP_ASSOC packet received,
        broadcast a GROUP_ASSOC message with group head = current node and set isCH = TRUE
        Set Timer2 to fire after a specific interval
TIMER2.FIRED()
    Broadcast a GROUP_INFORM_MEM message with
    information such as group head and other group members.
RECEIVEMSG.RECEIVE()
1   GROUP_ASSOC message received,
        isCh = FALSE
        send GROUP_INFORM_CH message to associate with a group head
2   GROUP_INFORM_CH message received,
        associate sender node id as part of group
        If Timer2 is not running, set Timer2 to fire after a specific interval
3   GROUP_INFORM_MEM message received,
        copy group data from packet (group head and other members information)
```

Fig. 9. Grouping Algorithm Pseudo-Code

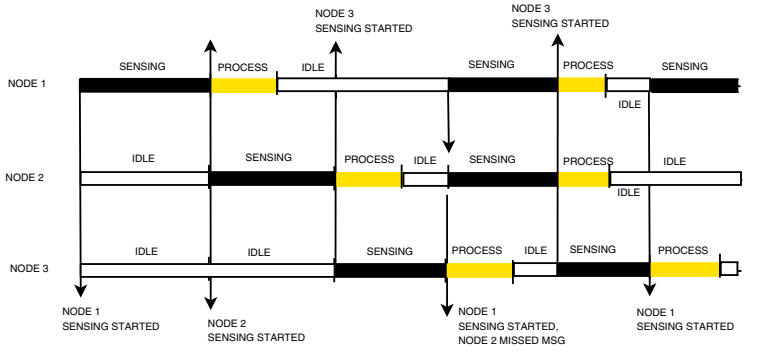


Fig. 10. Illustration of the Group Coordination Algorithm

close proximity are part of the same group. This approach will work if the distance between groups of nodes is greater than the radio propagation distance for the set RF power level. For SCOPES, we empirically set T1=8s and T2=60s. The RF power level for group communication was set to $< -25dBm$.

4.2 Distributed Coordination and Scheduling

Once groups are formed, a node coordination scheme enables non-overlapping, continuous sensing coverage. Our notion of node coordination uses "soft-state" [10] to achieve continuous sensing coverage for a particular area. An illustration of the working on the scheme is shown in Figure 10. Here, each node sends an update message to its group members before it starts sensing. When the other nodes in the same group receive an update message, they advance their "start sensing" timers by one sensing period. When the sensing timer expires, another node sends an update message informing that it has started sensing the area. This way, we can achieve continuous sensing of an area, given sufficient deployment. As all the coordination messages are sent over the radio, we can never discount the possibility that an update message was missed by a particular node. By design, if an update message is lost, the system does not break down. In the event of a lost update message, nodes are expected to start sensing when their sensing timer expires. This also helps to provide continuous coverage in the event of node failures. When an update message is lost, multiple nodes will sense the area in that cycle but the schedule is resumed as soon as the new update messages are received in the following cycle. The current scheme adapts to node failures while ensuring that cameras are providing continuous sensing coverage all the time and non-overlapping coverage for a majority of the time.

4.3 Performance Evaluation

We performed experiments to evaluate the performance of the SCOPES when nodes work in coordination, sensing in non-overlapping intervals of time. The results are presented (refer Table 5) for a single, two hour experiment involving

Table 5. Comparison of detection performance with and without node coordination

Operation Mode	DFP/False Negative %	False Positive %	Number of Objects
Without Coordination	20.0%	21%	85
With Coordination	15.3%	18.5%	103

Table 6. Counting the number of occurrences of false positives and false negatives along with their causes (NOTE: 1. Data is acquired from a single, two hour experiment with 3 nodes working together with coordination. 2. Under false negatives, 103 was the total number of people passing beneath the SCOPES nodes. 3. Under false positives, 193 is the total number of messages received at the base station).

Mis-Detections	Reason	Occurrences	Percentage
False Negatives	Object detected at memory bank border	4 out of 103	3.8%
	Object detected at limit of Sensing Range	8 out of 103	7.7%
	Software Inadequacy	4 out of 103	3.8%
False Positives	Objects in background	15 out of 193	7.7%
	Over-counting due to split-objects	21 out of 193	10.8%

3 nodes. We compare the performance of the SCOPES system in the presence and absence of coordination (see Section 3.4). Here, we see that DFP is reduced to 15% when nodes work with coordination as compared to 20% without any coordination. The DFP is also the false negative percentage for the system. DFP shows significant improvement with node coordination and we would like to continue that evaluation in future work. We have shown earlier (see Figure 4(d)) that we can significantly reduce the power consumption by using node coordination. Also, the relationship between detection performance, detection latency and memory usage (see Figures 7(b) and 8(b)) is independent of the presence or absence of coordination and hence, we expect these results to show similar trends. To complete our evaluation of the system, we provide a quantitative analysis of the detection failures in SCOPES.

Analysis of Detection Failures: In Table 6, we enumerate the number of occurrences and their respective percentages along with the associated reason for misdetection, in the presence of node coordination.

False Negatives: We report a false negative when the ground truth indicates that there is an object in the foreground whereas our system reports none.

Objects missed due to software: Under-counting occurs when the object detection algorithm is unable to differentiate the object from the background. This happens because the colors of objects in the foreground do not contrast enough against the background to trigger object detection. Another scenario where under-counting could occur is when the object recognition algorithm (see Section 2.2) merges two objects that are in close proximity to each other.

Under-counting objects due to hardware: This occurs due to the following reasons: (a) SPR of nodes and (b) limitations of the sensing hardware. Detection failures due to SPR are avoided by using node coordination. However, the camera fails to detect

an object due to loss of image data when the camera is switching memory banks. Since, the object is seen in only one frame in each bank, the algorithm would report no direction information as it does not combine information from successive banks. This problem could be resolved if memory was continuous and not split into banks. Objects that move close to the sensing range of the camera are missed because the camera is not able to cover the entire object from its point of view.

False Positives: False positives result mainly, due to the high sensitivity of the simple background subtraction algorithms used to detect the presence of objects in the image foreground from the fixed thresholds. This might be due to over counting of objects in the foreground and camera hardware calibration.

Over-Counting Objects: Over-counting occurs because the object recognition algorithm might split one object into two objects. It also occurs when a foreign object becomes part of the background for a short time.

Camera Hardware: In SCOPES, when the camera starts capturing images, at times, the image at the start of the burst exhibits higher brightness as compared to all the rest due to calibration issues. This behavior of the imager results in false positives. However, the resulting message contains information about an object with disproportionately large number of pixels and no direction information. We neglect such objects when computing the false positives for our system.

As part of future work, we would like to provide detection failure comparisons between the output of OpenCV program, using computationally complex algorithms, and our system.

5 Comparisons with Previous Work

In this section, we compare SCOPES with related work in the area of embedded camera sensor networks on issues like processing algorithms, latency, memory usage, detection probability and evaluation methods.

Kulkarni *et al.* [5] presented the design, implementation and evaluation of SensEye, a multi-tier camera sensor network for surveillance applications. The work aimed at showing that a multi-tier network can balance the conflicting goals of latency and energy-efficiency. In the evaluation experiments, circular objects were projected onto a wall with an area of $3m \times 1.65m$. Objects appeared at different location for a certain time duration with only one object present at a time. SensEye detected 42 out of 50 object appearances. It achieved 100% detection probability when objects are in view for 9 seconds which decreases to 52% when object time duration is 5 seconds. For moving objects, speeds were varied from 0.2m/s (all objects detected) to 0.6m/s (38% objects detected).

As seen in SensEye, a camera-based surveillance system fails when the speed of the object exceeds the capability of the system. From empirical data, it is said that humans move at speeds ranging from 1-1.5m/s [11]. The main difference between the evaluation of SensEye and SCOPES is that SCOPES was evaluated in uncontrolled real-life conditions where it had to account for variations in light

conditions, shadows, occlusions, and the size and speed of people moving in the environment. SCOPES still has an average detection probability of 84% when we deploy 3 camera nodes to cover an area, which improves to 98% for 6 nodes. Based on the image capture speed of the camera, SCOPES will fail to capture information required to detect an object if the object moves at a speed greater than 8m/s (no image data collected) and will fail to infer the direction if the object moves faster than 4m/s (only one image frame collected).

Teixeira *et al.* [6] proposed a motion histogram approach to count people in indoor spaces. The hardware infrastructure comprised of Intel iMote2 sensor nodes with OmniVision OV7649 imagers. The iMote2 sensor platform operates at 104MHz and is capable of processing 8fps while consuming 322mW (Imote + Camera) of power. Six nodes were deployed with minimum overlap between areas covered by the cameras. Each camera has a field of view of 3m x 2m. The experiments consisted of five people moving inside a lab setting.The system has a detection rate of 89.5% when a single person is present inside the camera network. This drops to 82.48% when two people are present and 79.8% for three. Using the case study of the same camera node, Jung *et al.* [7] present lifetime models for trigger-driven and schedule-driven sensor networks. Their models predict the energy budgets under different application requirements. The results in the paper show the variation in the lifetime of the camera sensor network with respect to the detection probability and object inter-arrival rate.

In comparison, in SCOPES, the Cyclops board operates at 4MHz and is capable of processing 1fps while consuming 115mW of power (Tmote + Cyclops). Inspite of the speed of the Cyclops platform (26 times slower than the iMote2) and the simple image processing algorithms, SCOPES is able to achieve detection probability of 84% with 3 cameras which increases to approx. 98% with 6 cameras covering the same area (see Figure 8). On a faster platform such as the iMote2, the SCOPES image processing algorithms would execute in roughly 26ms, eliminating the need for multiple cameras to provide coverage during the detection latency period of a camera while achieving comparable, if not superior, performance to the motion histogram approach. The performance evaluation of SCOPES highlights the point that by using computationally simple image processing techniques it is possible to achieve detection probabilities comparable to techniques like the motion histogram approach, inspite of differences in the computational capabilities of the underlying platforms. In SCOPES we analysed the power consumption for continuous sensing (without coordination) or interleaved sensing (with coordination) nodes, which differ from the operation models considered in [7]. In addition, in SCOPES, we also provide a detailed analysis and evaluation of the memory usage, detection latency and detection probability as a function of the system parameters.

6 Conclusion and Future Work

In this paper, we argued that previous studies lacked extensive performance evaluation of camera sensor networks in real life conditions, raising doubts regarding

the sustainability of such systems. In contrast, through analysis, simulation and extensive experimentation, we showed that deployment of multiple nodes working in coordination with each other eliminates some of the problems associated with network lifetime, data processing latency and quality of detection performance. In addition, we analysed the detection failures of SCOPES by describing the causes of misdetections and quantifying their effects. Our system provides on par or better detection performance than other approaches that have computationally intensive algorithms and more capable hardware, with a slightly higher deployment cost. In summary, the paper presented a comprehensive design for a embedded camera sensor network, alongwith extensive testing of parameters affecting the system. As part of future work, we would like to pursue the development of lightweight image processing algorithms for camera sensor networks. We would also like to investigate the tradeoff between the network traffic and detection quality by sending the entire image instead of summarized data.

References

1. Viola, P., Jones, M.: Robust real-time object detection. International Journal of Computer Vision (2001)
2. Viola Jones Detector for CMUcam3, `http://www.cmucam.org/wiki/viola-jones`
3. Aslam, J., Butler, Z., Constantin, F., Crespi, V., Cybenko, G., Rus, D.: Tracking a moving object with a binary sensor network. In: SenSys 2003, pp. 150–161. ACM Press, New York (2003)
4. Gu, L., Jia, D., Vicaire, P., Yan, T., Luo, L., Tirumala, A., Cao, Q., He, T., Stankovic, J.A., Abdelzaher, T., Krogh, B.H.: Lightweight detection and classification for wireless sensor networks in realistic environments. In: SenSys 2005, pp. 205–217. ACM Press, New York (2005)
5. Kulkarni, P., Ganesan, D., Shenoy, P., Lu, Q.: Senseye: a multi-tier camera sensor network. In: MULTIMEDIA 2005, pp. 229–238. ACM Press, New York (2005)
6. Teixeira, T., Savvides, A.: Lightweight people counting and localizing in indoor spaces using camera sensor nodes. In: ICDSC 2007, September 25-28 (2007)
7. Jung, D., Teixeira, T., Barton-Sweeney, A., Savvides, A.: Model-based design exploration of wireless sensor node lifetimes. In: Langendoen, K.G., Voigt, T. (eds.) EWSN 2007. LNCS, vol. 4373, pp. 277–292. Springer, Heidelberg (2007)
8. Rahimi, M., Baer, R., Iroezi, O.I., Garcia, J.C., Warrior, J., Estrin, D., Srivastava, M.: Cyclops: in situ image sensing and interpretation in wireless sensor networks. In: SenSys 2005, pp. 192–204. ACM Press, New York (2005)
9. OpenCV, `http://opencvlibrary.sourceforge.net/`
10. Ji, P., Ge, Z., Kurose, J., Towsley, D.: A comparison of hard-state and soft-state signaling protocols. IEEE/ACM Transactions on Networking 15(2), 281–294 (2007)
11. NationMaster Orders of magnitude (speed), `http://www.nationmaster.com/encyclopedia/`