SCOPES: Smart Cameras Object Position Estimation System

Ankur Kamthe, Lun Jiang and Alberto Cerpa
Computer Science and Engineering, School of Engineering,
University of California, Merced; Merced, CA 95344
akamthe,ljiang2,acerpa@ucmerced.edu

September 14, 2007

Abstract

In this paper we present SCOPES, a distributed Smart Camera Object Position Estimation sensor network System that provides maps of distribution of people in indoors environments. Each node in the system is comprised of a cyclops camera that performs local detection and processing of the visual information and a tmote sensor node, which provides multihop communication. SCOPES uses local adaptive techniques that enables intelligent duty-cycling [1] *between the active sensing and the information processing tasks performed at each node. The system switches between the fast and simple background subtraction algorithms for object detection, to the more computationally intensive object grouping algorithms for estimating the number and direction of travel of multiple persons in the local field of view. By aggregating meta-information generated by each node, SCOPES is able to minimize the total data transmitted in the network and still be able to generate an accurate density estimation map of the coverage area.*

Using analysis, simulation and experimentation, we show that the system is able to provide a small global error estimate of the spatio-temporal distribution of people in indoors environments despite the absence of continuous sensing when doing local information processing and sparse coverage. Moreover, we show that performance of the system degrades gracefully in the presence of memory and people mobility constraints. In the paper we show the results of people density estimation, power consumption, memory usage, latency and detection probability on a real system deployment comprised of 16 nodes running in a research building at the University of California, Merced.

1 INTRODUCTION

In the past decade, the day-to-day life of every human has been invaded by a plethora of sensors, of all shapes, sizes and purpose. Sensors have developed greater capabilities, with every new generation showcasing better computational and communication paradigms. Surveillance and monitoring

[1]*In this paper, the term duty-cycle refers to ratio of the active sensing time of the camera to the total camera uptime. It is not to be confused with the conventional engineering definition which is the proportion of time an electronic device is operated.*

applications have benefited vastly from developments in the field of sensors. Networks comprising 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.

Wireless sensor networks have eased the burden of deployment, but the principal challenge of getting meaningful data from these sensor networks within the constraints of the application, still remains a largely unsolved problem. For example for surveillance applications, the sensor should have the computation capapbility to filter the noise in the video/image data. In order to extract maximum information/features from the captured data, a large number of sensors are usually deployed. Lastly, network connectivity and power consumption per node control the effective lifetimes in wireless networks. Thus, data fidelity, cost-effectiveness and network lifetimes pose conflicting requirements that need to be balanced. In this paper, we chiefly concentrate on the first i.e. data-fidelity, in view of an object detection and tracking application using image data from Cyclops, a low-power camera sensor board [1]. Object detection and tracking dictates the use of image processing algorithms which are expected to filter and extract meaningful data from the image data generated by Cyclops. The time required for data processing should not be too high, else it will degrade the information being collected by the network (increase in missed detections).

In this paper, SCOPES, a distributed smart camera object position estimation sensor network system is presented. We develop a density estimation application from the object detection and tracking data obtained from the system that provides maps of the distribution of people in indoors environments. SCOPES uses local adaptive techniques that enable each node to maximize its active sensing period while keeping the information processing time to a minimum. The system switches between the fast and simple background subtraction algorithms for object detection, to the more computationally intensive grouping algorithms for estimating the number and direction of travel of persons in the local field of view. Using the information gathered from multiple nodes, we are able to construct a map showing the density of people in a section of the building.

The main contributions of our paper are as follows:

- developing a prototype implementation of an personnel tracking systems that allows the position estimation of people in indoor environments.
- developing local adaptive algorithms that combine object detection and background update and minimize the amount of information transmitted for processing.
- actual deployment results for a representative application, together with extensive performance evaluation.

The rest of this paper is organized as follows: in Section 2, we present a brief summary of object detection and tracking systems from a hardware and software perspective. In Section 3, we describe the hardware and software architecture of SCOPES, along with details of the object detection and tracking algorithms, and the system deployment. In Section 4, we provide some simple mathematical analysis through simulation and the experimental results of the performance of the system with respect to density estimation, quality of detection and tracking, operational duty-cycle, power consumption and data delivery latency. Finally, in Section 5, we conclude by summarizing our results and discussing future work.

2 RELATED WORK

Object detection and tracking systems depend on the underlying hardware and software infrastructure for their efficient performance.

The software infrastructure comprises of image processing algorithms for feature extraction and interpretation from the incoming data (images/video). In [2], design of a feature extractor is described for an image thresholding algorithm. It involves extracting the "expressive" and "discriminating" features using a modified Hebbian learning approach. In [3], a principal feature extraction approach is proposed for classifying a pixel as belonging to the background or foreground. Han et al. [4] describe a multiple hypothesis approach for pruning the possibilities of the number and trajectories of objects in the video stream.

However, these techniques are computationally intensive and find use in surveillance and monitoring applications with permanent deployments and no dearth of processing and communications capabilities. Such techniques are not suitable for implementation in resource-constrained, untethered platforms like wireless sensor networks that have to maintain a fine balance between a long-life and operational performance.

The hardware infrastructure for object detection and tracking systems ranges all the way from binary sensors to sensors delivering high quality video over 802.11 networks [5]. Some of the networks [6] demonstrated for use in tracking movements of objects comprise of hundreds of small, densely distributed wireless sensor nodes deployed in the field. They utilize collaboration between neighbors to detect and track a moving target, and alert other sensor nodes along the projected path of the target. In [7, 8], SensEye, a multi- tier camera sensor network is described for surveillance applications. The work aimed at showing that a multi-tier network can balance the conflicting goals of latency and energy-efficiency. However, the experiments conducted in some of these works [7, 8, 6] are not representative of real-world scenarios.

Using a binary sensor network [9], Aslam et al. describe a framework for tracking a moving object by using a particle-filter algorithm. However, these sensors do not provide enough information to localize the object. In [10], VigilNet, a detection and classification system is described which is used for tracking the motion of ferrous objects. It combines information from magnetometer and acoustic (microphone) sensors to infer the location and direction of travel of objects. Information is condensed by the group head node to avoid overwhelming the base station with data. In [11], a similar approach using MICA2 motes along with magnetometer sensors is shown for detecting objects capable of generating magnetic fields. In [12], Ren et al. propose an analytical framework for object detection in sensor networks and develops wave sensing scheduling protocols to maximize network lifetimes while achieving bounded worst-case object detection quality.

In contrast, using smart cameras we can overcome restrictions like magnetic fields. Also, our work addresses issues like maximizing the sensing period for camera sensors using local adaptive image processing algorithms. This enables switching between the fast background subtraction and the computationally intensive pixel grouping algorithms, while accounting for changes in the background. We present results from a real-life deployment of the system regarding quality of detection and tracking, information arrival latency, memory and power consumption in view of a position estimation application which provides maps with distribution of people in indoor environments at any point of time.

3 SYSTEM DESCRIPTION

In this section, we describe the SCOPES system, including the hardware and software architecture, as well as the algorithms and the details of the experimental deployment used.

3.1 Hardware

Our SCOPES implementation comprises of an Agilent Cyclops camera interfaced with a Moteiv Tmote Sky [13] module via an intermediate custom adaptor board. The Cyclops camera performs local detection and processing of the visual information and the Tmote module provides multi-hop communication capability. The Cyclops board and the Tmote share power from the Tmote's battery back.

Cyclops: The Cyclops consists of an Agilent ADCM-1700 imager module, an ATMEL ATmega128L micro-controller (MCU), a Xilinx XC2C256 CoolRunner CPLD, an external SRAM (0.5MB) and an external Flash (0.5MB). The MCU controls the Cyclops sensor, sets the image capture parameters for the imager, instructs the imager to capture a frame, is capable of running simple image processing algorithms and is responsible for communication with host Tmote module. The Cyclops uses the external SRAM to supplement the limited internal MCU memory (4KB). However, as the MCU can address a maximum of 64KB of memory, the external SRAM and Flash is divided into eight 64KB memory banks. The entire 512KB of SRAM can be utilized by switching between memory banks.

Tmote: The Moteiv Tmote Sky module is comprised of an ultra low power Texas Instruments MSP430 F1611 micro-controller featuring 10kB of RAM, 48kB of flash and a Chipcon CC2420 radio for wireless communications. Tmote Sky has two expansion connectors, a 10-pin and a 6-pin IDC header. The Cyclops is connected to the Tmote module through these expansion connectors. The Cyclops is powered by the voltage pins mirrored on the expansion connector. Communication between the Tmote and the Cyclops is carried over the I2C bus. Whenever the Cyclops wants to transfer information to the Tmote, it interrupts the operation of the Tmote using an external trigger pin connected to the User Interrupt pin on the expansion connector. The Tmote then posts a read request to the slave device, which cyclops responds by sending the data over the I2C bus.

Interface Board: The interface board has a 51-pin Hirose connector on one side and a 10-pin and 6-pin female connector on the other side for accommodating the Cyclops board and the Tmote Sky module, respectively. The interface board also contains pull-up resistors for the clock and data lines for the I2C bus on the Tmote.

3.2 Software

The Cyclops and the Tmote run the event-based TinyOS operating system for wireless sensor network platforms. The Cyclops and the Tmote run the event-based TinyOS operating system for wireless sensor network platforms. TinyOS is written in nesC. It is characterized by a small memory footprint and direct-access to the underlying hardware. TinyOS has a synchronous execution model wherein a task will run to completion if not preempted. Due to this, long computations are split into

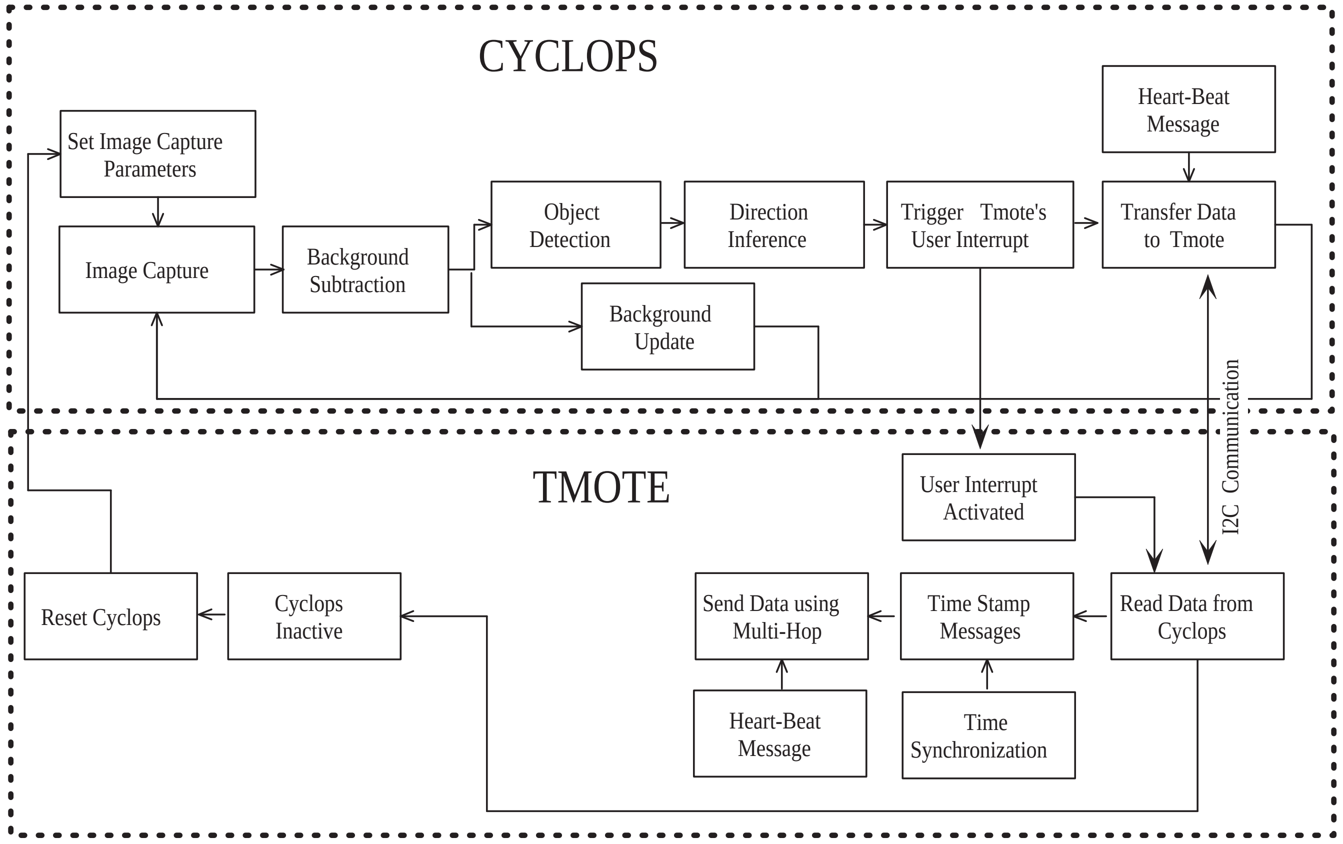


Figure 1: Software Block Diagram of SCOPES. The figure shows the different operations being executed on the Cyclops and Tmote modules. The arrows indicate the logical sequence of operations and interactions between the two devices.

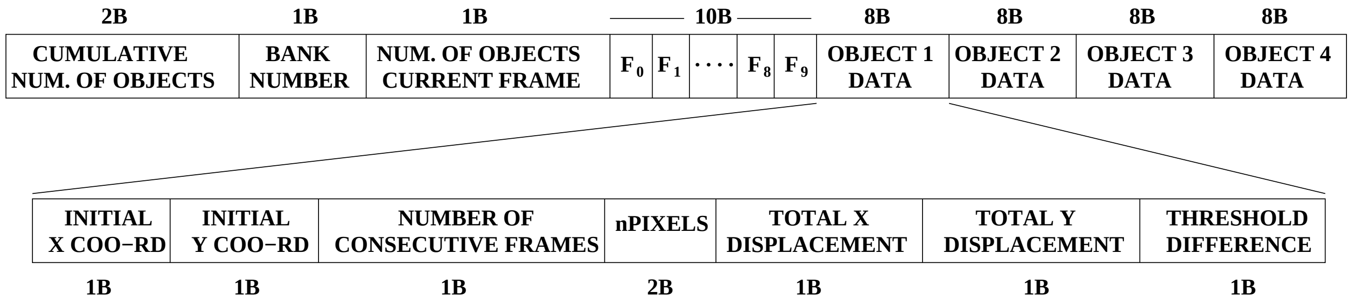


Figure 2: Cyclops Message Format. This is the payload of the TOS_Msg sent by the radio to the base station. (Note: 1. B stands for bytes 2. Figure does not show the other fields used for recording additional details like battery reading, routing tree information, time stamp, etc.)

smaller tasks that are posted sequentially. The TinyOS scheduler handles all the hardware resources and ensures fairness among the different modules. A block diagram of the software framework is shown in Figure 1.

Cyclops: The image processing (object detection and tracking) algorithms running on the Cyclops for SCOPES are implemented within the matrix operations libraries. We added two new functions, *group* and *abssub2thres*, in the *matrixArithmetics* interface. The *abssub2thres* function is responsible for background subtraction and returns the cumulative absolute difference in pixel intensity for an image. The *group* function groups neighboring pixels which exceed a preset threshold into objects and stores information regarding their centroid, size and average threshold difference.

Whenever the Cyclops has anything significant to report, it triggers the UserInterrupt pin on the Tmote. It waits for the Tmote to post an I2C read request to which it responds by sending the payload comprising of centroid, size, threshold difference and direction information regarding detected objects.

The structure of the cyclops payload (see Figure 2) is explained as follows: Cumulative Number of Objects is the number of objects detected by the Cyclops till that time and acts as an implicit sequence number, Bank Number is the current memory bank number, Number of Objects in Current Frame is the number of objects regarding which we are transferring data in the payload and F_0,....,F_9 contain the number of objects detected in each frame. The object data is comprised of the initial X and Y coordinates of the object (centroid), Number of Consecutive Frames is the number of consecutive frames the object appeared in, nPixels is the object size in pixels, Total X Displacement and Total Y Displacement are the displacements of the object along the X and Y axes and Threshold Difference is the average absolute difference in threshold. This information is used by the centralized Density Estimation Algorithm (see Section 3.3).

In addition to this, the Cyclops sends an “is active” status message to the Tmote every minute, over the I2C bus.

Tmote: The Tmote runs a derivative of the TinyOS operating system called Boomerang provided by Moteiv Inc., which provides extra function primitives like resource (I2C, radio) handlers, MultiHop communication (MultiHopLQI) and time synchronization (NetSyncM) native to the Tmote module. The I2C, SPI, UART and USART0 lines are common in the Tmote. In order to exclusively access any of these buses, a resource handle needs to be granted.

Whenever, the Cyclops activates the UserInterrupt pin on the Tmote, the interrupt handler posts a resource request for proceeding with an I2C read operation. When the I2C resource is granted, the Tmote requests data from the Cyclops using the appropriate slave device address. After the read operation is completed, the resource is released by the Tmote. To ensure that the Cyclops is functioning correctly, the Tmote expects an “is active” or “heart-beat” message from the Cyclops every minute. If the Cyclops misses four consecutive “heart-beats”, then it is reset by the Tmote. At the same time, every four minutes, the Tmote sends a “heart-beat” message to the base -station with a special payload to differentiate it from the object detection payload from the Cyclops. The TinyOS data packet sent over the radio is 76 bytes long ($TOSH\_DATA\_LENGTH$).

Cyclops Action	**nFrames**				
	1	3	5	7	10
Image Capture Only	0.68s	1.04s	1.43s	1.85s	2.51s
Image Capture w/ Background Subtraction (BS)	0.7s	1.15s	1.54s	2.01s	2.74s
Image Capture w/ (BS + Object Grouping)	1.3s	2.76s	4.71s	6.49s	9.50s

Table 1: Average Time (in second) required for executing various image processing operations $nFrames$ images at a time for the Cyclops camera. The time information was recorded by executing each of the operations 100 times on two different Cyclops camera modules.

3.3 Algorithms

Object detection is of great importance in surveillance and monitoring applications. It should be robust to noise, adaptive to gradual changes in background –such as illumination changes– and have low processing latency in order to capture significant events. The first step is background subtraction, followed by object detection and grouping, and finally, the direction inference part. Table 1 shows the time taken to execute some of these operations.

Background Subtraction: In order to achieve low processing latency and to maximize the probability of capturing significant objects, we implemented a modified version of the simple background subtraction algorithm to determine the presence of an object in the foreground. Here, we compute the absolute difference in value for each pixel in the current image and the background. The sum of difference in values for all the pixels and the difference per pixel in the current image provides a measure of change in the foreground with respect to the background. If this change is significant, it acts as a weak indicator of the presence of an object. In our approach, we maintain a moving average of the mean and deviation of the difference in value per pixel for all images.

$$Mean = (1 - \delta) \times Mean + \delta \times DifferencePerPixel$$
$$\begin{aligned} Deviation &= (1 - \delta) \times Deviation + \\ &\quad + \delta \times |DifferencePerPixel - Mean| \end{aligned}$$

Depending on the value of the $Mean$ and $Deviation$, the cyclops decides whether to process the image for presence of object.

$$\begin{array}{ll} \text{Object Detection} & \text{if } (Mean > 1.5 \times Mean_0 \text{ and} \\ & \ Deviation > 0.1 + Deviation_0) \\ \text{Background Update} & \text{if } (Mean > 1.5 \times Mean_0 \text{ and} \\ & \ Deviation < 0.1 + Deviation_0) \\ \text{No Operation} & \text{otherwise} \end{array}$$

For our purpose, the value of δ is set to 0.8 so as to impart a greater weight to the current image information as compared to the history information from old images. $Mean_0$ and $Deviation_0$ are preset to 2 and 0, respectively [2].

[2]The *group* and *abssub2thres* functions were implemented in MATLAB and applied to a set of 16,000 images

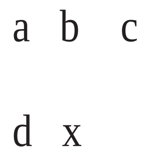


Figure 3: Grouping Pixels

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

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

Object Detection: To detect the presence of significant objects in the foreground, we group neighboring pixels with absolute difference in intensity greater than the threshold. This is done by raster scanning the image starting from the top left corner. A pixel is considered to be part of an object if the absolute difference in intensity of the current pixel along-with that of its left neighbor, top neighbor, top-left neighbor and top-right neighbor exceed a pre-specified threshold (42, in our case). 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. Then, if absolute difference in intensity of x, a, b, c and d exceeds a pre-specified threshold, then x is considered to be part of an object.

Group id's labels are assigned for each pixel using a raster scan. Assigning group id's is done using the rules explained in Table 2. After grouping pixels based on individual and spatial threshold criteria, each of these primary groups which are in close proximity of each other are merged together to account for fragmentation of a big object. For each of the objects, we maintain information regarding the centroid (x and y coordinates), number of pixels and average difference in threshold.

Direction Inference: The next step after object detection is to infer its direction information. In order to do that, we match objects in successive frames according to their size (in pixels). Any object in current frame that cannot be matched to another in the previous frame is stored as a new object. Objects from previous frames that disappear or cannot be matched to objects in current frame are moved into a data structure. We maintain information regarding the original position, displacement and number of consecutive frames for each object that appears across successive images in the current memory bank. After processing all the images in the current bank, we generate an array of structures that contains information on a maximum of four most significant objects based on appearance in maximum consecutive frames.

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, the path and final destination is inferred by accounting for the presence of walls and doors in the vicinity of the node. Objects with no direction information are considered as noise and filtered out. Since, the entry point and exit vector of an object is computed, the distribution of people in sections of the building can be determined.

captured sequentially by the Cyclops. We used the MATLAB implementation as a sandbox to test the effect of different parameters (δ, $Mean_0$ and $Deviation_0$) on the performance of the object detection and grouping algorithms. The initial values for these parameters were set based on the MATLAB results.

Term	**Meaning**
$nFrames$	number of images captured consecutively
$nBanks$	number of memory banks (see Sec. 3.1)
T_S	total cyclops sensing time (camera ON), which depends upon $nBanks$
$T_S^{nFrames}$	cyclops sensing time for $nFrames$ images
T_{BS}	total background subtraction processing time for $nBanks \times nFrames$ images
T_{OG}	object grouping processing time per image
P	power consumption per node
DP	detection probability per node
DFP	detection failure prob. per node, $1 - DP$

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

4 PERFORMANCE EVALUATION

In this section, we report results from a performance evaluation of SCOPES. A combination of simple mathematical models using simulations and real-life deployment results is used to understand the behavior and to validate the performance evaluation of the system.

4.1 Objective Functions, Notation and Constants

Our aim for SCOPES was to function as a surveillance system to monitor the occupancy in indoor environments such as office buildings. Surveillance systems are characterized by their ability to detect and track the movement of people. The performance of the system is determined by its ability to effectively monitor the target area for movement of individuals. Some metrics/objective functions which are of interest are as follows:

Global/Local Density Estimate: How good can we estimate the occupancy in the total area covered? How accurately in each partition area?

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

Memory Usage: How much memory is required to achieve acceptable performance? How is the performance affected as a function of the memory size?

Detection Latency: How long does the system take to estimate the position of a person?

Detection Probability: How good is the estimate of flow movement across different partition areas within the building?

Table 3 shows the basic notation for many parameters used in the remaining of the paper. Many of these parameters may affect one or more objective functions. Our goal is to maximize the people density estimation and the detection probability, while minimizing the detection latency, memory usage and power consumption, but they pose contradictory requirements. Analyzing these trade-offs when proposing a solution is on the the key contributions of this paper.

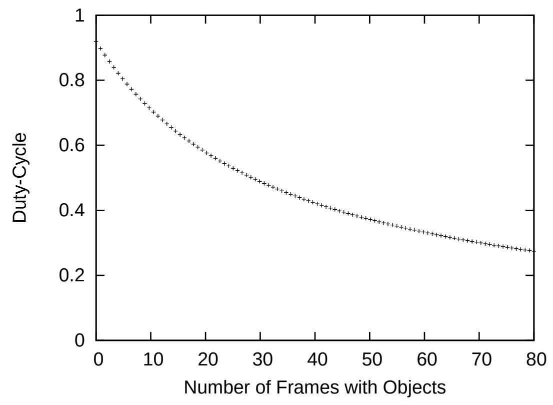


Figure 4: Duty Cycle as a function of the total number of frames with an object. As the number of frames/images increases, the duty cycle decreases since each SCOPES node spends more time processing the images using the CPU intensive grouping algorithm.

4.2 Analysis and Simulation

To understand the relationship between our objective functions and the different parameters we present some simple mathematical analysis and simulation.

We model the arrival times of people passing under a camera with an exponential distribution. The probability density function (pdf) of an exponential distribution has the form

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

where $\frac{1}{\lambda}$ is the mean arrival time for an event. A timeline of events is reconstructed from these arrival times. 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 positioned to observe these events.

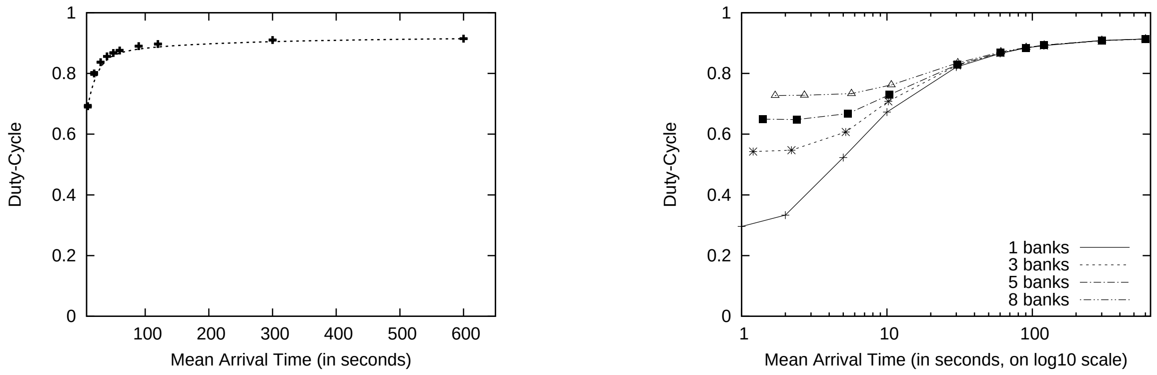
Duty Cycle (DC): is the ratio of total sensing time to the total uptime of the camera. Understanding the Duty Cycle is critical, since it affects many of our objective functions, including global/local position estimation, power consumption, detection probability and detection latency.

In Figure 4, we observe the variation in duty-cycle of a node with respect to the number of frames in which an object is detected. It can be expressed as:

$$DC = \frac{T_S}{T_S + T_{BS} + N \times T_{OG}} \tag{1}$$

The number of grouping function calls N vary from 1 to 80 which is the total number of images captured in one duty-cycle using all available memory. Since the grouping function incurs the highest processing cost in terms of time, the duty-cycle of a node chiefly depends upon the time spent in grouping pixels for object detection.

In Figure 5(a), we observe the variation in duty-cycle of a node with respect to the mean arrival time $\frac{1}{\lambda}$ of an individual observed beneath a camera. This relationship can be expressed as follows:



(a) Duty Cycle vs Mean Arrival Time (b) Duty Cycle vs Memory Usage

Figure 5: Fig. 5(a) shows the *Duty Cycle* as a function of the mean arrival time. The duty cycle increases as the mean arrival time increases due to a reduction in processing time since there are not that many objects to process. Fig. 5(b) shows the *Duty Cycle* as a function of the mean arrival times for different number of memory banks available. The duty cycle gets significantly affected by the amount of memory for small arrival times. For large mean arrival times, the amount of memory does not play a significant role.

$$DC = \frac{nT_S}{n(T_S + T_{BS}) + \sum_{i=1}^{n} \alpha_i \beta_i T_{OG}} \tag{2}$$

where n is the number of times we capture a set of 80 images, α_i is the number of times an event is observed and β_i is the average number of frames occupied by an object in the current (i^{th})sensing cycle. When the mean arrival time is low, events occur at a faster rate, and the camera spends a longer time processing the image data, leading to a low duty-cycle.

In Figure 5(b), we observe the variation in duty-cycle of a node with respect to the number of memory banks *nBanks* as a function of the mean arrival times of an individual observed beneath a camera. The only factor that is affected by the number of memory banks is T_S as $T_S = nBanks \times T_S^{nFrames}$. This relationship can be expressed as follows:

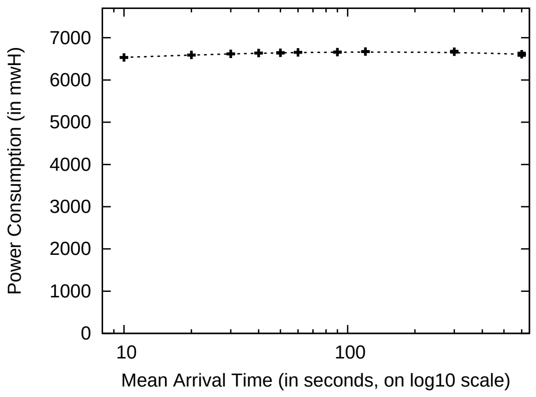


Figure 6: Power Consumption as function of the mean arrival times. The power consumption is dominated by the total sensing time, which does not significantly change with the arrival times.

Device Operation	**Power Consumption**
Cyclops Image Capture	45mW
Cyclops Micro-Controller Active	15mW
Tmote MCU + Radio RX	65.4mW
Tmote MCU + Radio TX (0 dBm)	58.3mW

Table 4: Power Consumption of the Cyclops and Tmote Sky Modules (for a 3V battery).

$$DC = \frac{n(nBanksT_S^{nFrames})}{n(nBanksT_S^{nFrames} + T_{BS}) + \sum_{i=1}^{n} \alpha_i \beta_i T_{OG}} \tag{3}$$

When the number of memory banks is small and the mean arrival time is also low, a node spends a greater percentage of its time processing the image data. The *Duty Cycle* increases as either the amount of available memory (*nBanks*) increases or if the mean arrival time is high. If the system is expected to be deployed in a busy area, it should be provided with sufficient memory resources.

Power Consumption (P): in Figure 6, we observe the change in power consumption as a function of the mean arrival times of an individual observed beneath a camera. Here the transmission and reception power of the Tmote is set to maximum. This relationship can be expressed as follows:

$$P = P_{cyclops}^{sensing} + P_{cyclops}^{proc} + P_{tmote}^{RX} + P_{tmote}^{TX} \tag{4}$$

where $P_{cyclops}^{sensing}$ is the power consumed during image capture, $P_{cyclops}^{proc}$ is the power consumed during image processing, P_{tmote}^{RX} and P_{tmote}^{TX} is the power consumed by the radio during reception/idle and transmission, respectively.

Most of the time the Cyclops is capturing images which accounts for most of the power consumption. Also, the Tmote spends almost all of its time in receiving/idle mode for routing purposes and this is independent of the mean arrival time of an object beneath the camera. For this paper, we experimented mainly with the detection and tracking aspects of the system and the nodes were active (on) all the time which led to this kind of behavior as regards power consumption. The power consumption of the Cyclops and Tmote Sky modules is given in Table 4.

Detection Failure Probability (DFP): This is the probability that none of the camera nodes covering a transition area detects the movement of a person. The main dependency is the *Duty Cycle*, since it will directly affect the probability of detection of each node. The relationship can be expressed as:

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

where n is the total number of nodes covering the same transition area. Figure 7 shows the expected failure probability of detection decreases both as the *Duty Cycle* increases and the number of nodes n increases. The numbers shown in the graph are the lower bound for the DFP i.e., in real-life the DFP will be higher because of the errors introduced by the limited computational capability of the nodes.

4.3 Experimental Deployment

We deployed 16 nodes on the ceiling in the corridors of the Second Floor of the Science and Engineering Building at the University of California, Merced. We separated the floorplan into 5 sections by deploying the nodes in sets of 4 nodes at entry/exit points. We employ multiple nodes as the field of view for a single node is not enough to cover the whole corridor and 4 nodes help ensure a reliable reading in the presence of interleaved sensing and image-processing periods. We also attached relay nodes on the ceiling for the sole purpose of improving the connectivity of the network and to ensure that we do not drop packets in the event of network congestion. The messages coming from the nodes are timestamped and stored in a log file at the base-station.

Moreover, we installed two Panasonic KX-HCM280A networked web cameras at strategic locations in the corridor to record the movement of people passing beneath the cameras. They are capable of capturing images at 10fps at a resolution of 640×480 and include built-in server software for controlling the camera from a standard web browser. The server uploads the images, periodically, into a laptop attached to the provided ethernet port using FTP. These images are timestamped using the local time of the laptop which have been synchronized using NTP. The images form the ground truth with respect to which we compare the data sent by the nodes.

4.4 Experimental Results

In this section we discuss the performance results of our experiments based on the different objective functions and goals.

The first aspect we address is the evaluation of the SCOPES system to build density estimation maps of area occupancy in a building, as well as transitions across the different sections.

Figure 8 shows the results of the different density estimation maps built at different times of the day. For these results, we use 8 memory banks ($nBanks = 8$) each recording 10 images ($nFrames = 10$). On each section, we plot the estimated number of people occupying the section followed by the ground truth value we got from the manual webcams video analysis. We also show the estimated number of incoming and outgoing movement of people across the different sections with arrows indicating the direction of travel. From the results we see that with a modest number

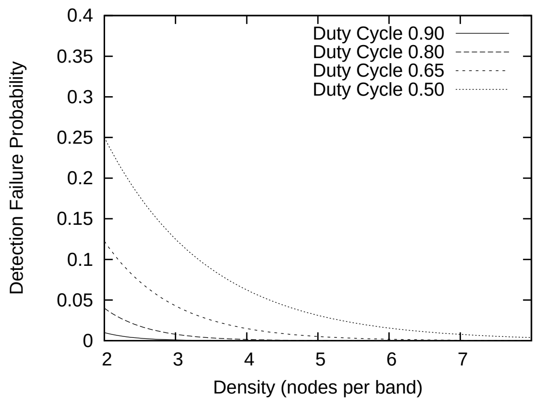
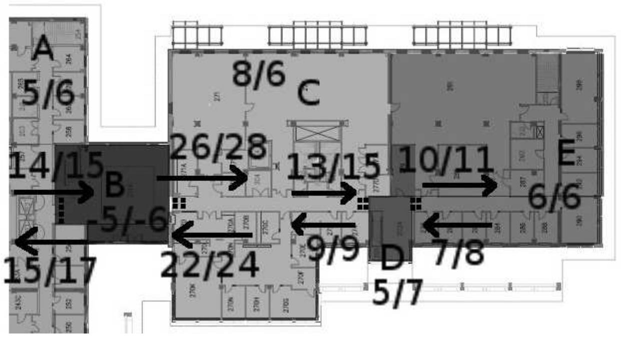
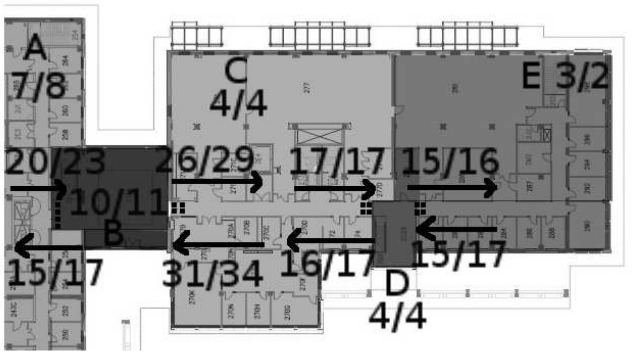


Figure 7: Detection Failure Probability as a function of density of nodes covering the same area and the $DutyCycle$.



(a) Morning Map



(b) Noon Map

Figure 8: Occupancy and Flow Estimation Maps created from data acquired from the cameras. (The arrows indicate the direction of motion. The number above the arrow indicates the number of people travelling in that direction as observed by the SCOPES node (number on the left) and by the Panasonic network cameras (number on the right).)

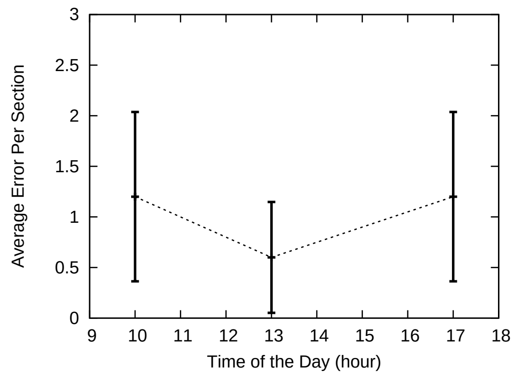
of sensor nodes, we are able to track the movement of people over more than 73 sq. meters of the second floor south wing part of the Engineering Building with a small reasonable error.

Figure 9(a) shows the average density estimation error of all the sections at different times of the day. The error bars show the standard deviation of the results. Since, the error bars are overlapping, we can say that the error does not change significantly in the statistical sense. We see that error remains bounded to reasonable values (under 2) at all times of the day.

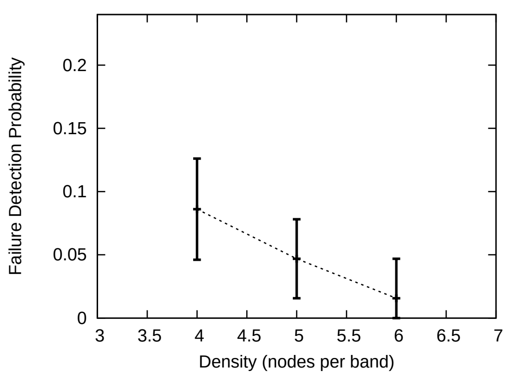
We would also like to explore the minimum error possible SCOPES could achieve as a function of the density of the deployment. Figure 9(b) shows experiments done with 4, 5 and 6 nodes per band covering the different sections. It is clear that deploying 6 or more nodes would result in a negligible failure rate.

In the next graphs, we explore how the detection latency and the detection probability perform as we constraint the amount of memory available to the nodes. Figure 10(a) shows the detection latency, i.e. the time it takes for SCOPES to know about a person transitioning among different areas. We see that as we decreased the amount of memory, the detection latency actually decreases, from 18 seconds when using 8 memory banks, to 10 seconds when using only 1 bank. The reason for this behavior is mainly due to the fact that the maximum processing time as we reduce the total number of banks gets significantly reduced, allowing the nodes to inform of the presence of a person moving much faster. Note that a possible performance optimization with applications that require the lowest detection latency could be to process all the information bank by bank, despite the total number of banks available, and in case of a person being detected, immediately transmit the meta data toward the base station. The only problem might be additional network traffic caused by data that could not be aggregated at the local node.

While the reduced number of banks could in principle reduce the detection latency, we would like to evaluate the probability of detection in this case. Figure 10(b) shows that detection probability gets significantly higher as we increase the number of memory banks available. From Figure 5(b) we learned that as numbers of memory banks available increases, duty cycle also tend to increase. A higher duty cycle implies a larger probability of detection, which is what happens in this case.

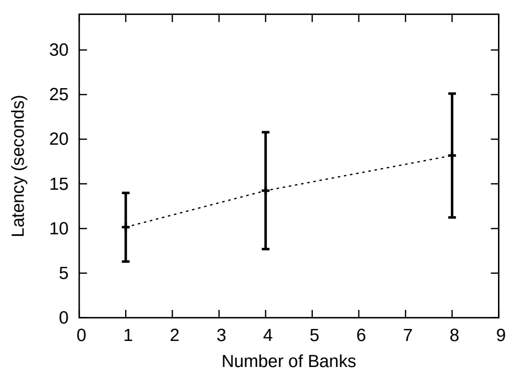


(a) Position Estimation Error

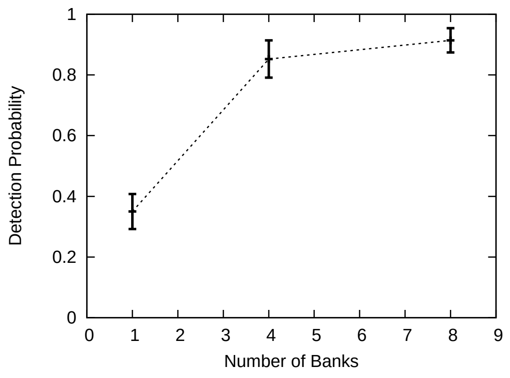


(b) Detection Failure Probability

Figure 9: Fig. 9(a) shows the Average Position Estimation Error per section for different times of the day. The position estimation error tend to be stable and small over time. The error shown on the Y-axis is in terms of number of people. Fig. 9(b) shows Detection Failure Probability as a function of the density of nodes covering the same area. The detection failure probability converges to a very small value after 6 nodes. NOTE CORRECT Y AXIS FOR DETECTION FAILUR PROBABILITY GRAPH



(a) Detection Latency



(b) Detection Probability

Figure 10: Fig. 10(a) shows the Detection Latency as a function of the number of memory banks. The latency tends to increase as we increase the available memory due to larger sensing and processing periods before transmission. Fig. 10(b) shows the Detection Probability as a function of the number of memory banks. The detection probability tends to increase as we increase the available memory due to larger duty cycle and higher probability of a person walking down the camera while sensing.

5 Future Work and Conclusions

In this paper, we presented SCOPES, a distributed smart camera object position estimation sensor network system. We developed a density estimation application from the object detection and tracking data obtained from the camera sensors that provides maps of the distribution of people in indoors environments. Using analysis, simulation and extensive experimentation, we show that the system is able to provide a small global error estimate of the spatio-temporal distribution of people in indoors environments despite the absence of continuous sensing when doing local information processing and sparse coverage. We also investigated the performance of the system with respect to memory usage, detection latency and detection probability on a real system deployment.

As part of future work, we plan to study several aspects of camera networks not covered in this paper. We would like to continue the experimental profiling of aspects we did analytically. We would also like to investigate and provide analysis of missed detections. We aim to develop sensing scheduling algorithms which would enable us to improve the lifetime of the network without sacrificing performance. We would also like to investigate the tradeoff between the detection latency and the additional network traffic generated by sending individual object data messages.

References

[1] Mohammad Rahimi, Rick Baer, Obimdinachi I. Iroezi, Juan C. Garcia, Jay Warrior, Deborah Estrin, and Mani Srivastava. Cyclops: in situ image sensing and interpretation in wireless sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 192–204, New York, NY, USA, 2005. ACM Press.

[2] Yong-Jian Zheng and Bir Bhanu. Adaptive object detection based on modified hebbian learning. In *Proceedings of the 13th International Conference on Pattern Recognition*, 1996.

[3] Liyuan Li, Weimin Huang, Irene Yu-Hua Gu, and Qi Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13:1459–1472, 2004.

[4] Mei Han, A. Sethi, Wei Hua, and Yihong Gong. A detection-based multiple object tracking method. In *ICIP '04: International Conference on Image Processing.*, 2004.

[5] Wu chi Feng, Brian Code, Ed Kaiser, Mike Shea, Wu chang Feng, and Louis Bavoil. Panoptes: scalable low-power video sensor networking technologies. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 562–571, New York, NY, USA, 2003. ACM Press.

[6] Rahul Gupta and Samir R. Das. Tracking moving targets in a smart sensor network. In *VTC '03: Proceedings of 57th IEEE Vehicular Technology Conference*, pages 3035–3039, 2003.

[7] Purushottam Kulkarni, Deepak Ganesan, and Prashant Shenoy. The case for multi-tier camera sensor networks. In *NOSSDAV '05: Proceedings of the international workshop on network and operating systems support for digital audio and video*, pages 141–146, New York, NY, USA, 2005. ACM Press.

[8] Purushottam Kulkarni, Deepak Ganesan, Prashant Shenoy, and Qifeng Lu. Senseye: a multi-tier camera sensor network. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 229–238, New York, NY, USA, 2005. ACM Press.

[9] Javed Aslam, Zack Butler, Florin Constantin, Valentino Crespi, George Cybenko, and Daniela Rus. Tracking a moving object with a binary sensor network. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 150–161, New York, NY, USA, 2003. ACM Press.

[10] Lin Gu, Dong Jia, Pascal Vicaire, Ting Yan, Liqian Luo, Ajay Tirumala, Qing Cao, Tian He, John A. Stankovic, Tarek Abdelzaher, and Bruce H. Krogh. Lightweight detection and classification for wireless sensor networks in realistic environments. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 205–217, New York, NY, USA, 2005. ACM Press.

[11] Tian He, Sudha Krishnamurthy, John A. Stankovic, Tarek Abdelzaher, Liqian Luo, Radu Stoleru, Ting Yan, Lin Gu, Jonathan Hui, and Bruce Krogh. Energy-efficient surveillance system using wireless sensor networks. In *MobiSys '04: Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services*, pages 270–283, New York, NY, USA, 2004. ACM Press.

[12] Shansi Ren, Qun Li, Haining Wang, Xin Chen, and Xiaodong Zhang. Design and analysis of sensing scheduling algorithms under partial coverage for object detection in sensor networks. *IEEE Transactions on Parallel and Distributed Systems.*, 18:334–350, 2007.

[13] Moteiv Corporation. http://www.moteiv.com/.