Lightweight People Counting and Localizing for Easily Deployable Indoors WSNs

Thiago Teixeira and Andreas Savvides

***Abstract*—We describe a lightweight method for counting and localizing people using camera sensor networks. The algorithm makes use of a motion histogram to detect people based on motion and size criteria. The motion histogram is an averaged shifted histogram that estimates the distribution of people in a room given the above-threshold pixels in a frame-differenced "motion" image. The algorithm provides good detection rates at low computational complexity. In this paper, we describe the details of our design and experimentally determine suitable parameters for the proposed histogram. The resulting histogram and counting algorithm are implemented and tested on a network of iMote2 sensor nodes. Our implementation on sensor nodes uses a custom sensor board with a commercial off-the-shelf camera, but the motion histogram is designed to easily adapt to ultralow-power address-event motion imagers.**

***Index Terms*—Assisted living, human counting, wireless sensor networks.**

Fig. 1. Our custom camera board with wide-angle lenses mounted onto the iMote2 sensor node.

I. Introduction

A SYSTEM that counts and localizes people is a common requirement in a broad spectrum of applications, such as assisted living, home care, security, workplace safety, and entertainment. For such a system to work for prolonged periods of time in an indoors environment where new people may enter and leave, and where objects may be introduced, replaced or moved, it cannot rely on wearable sensors or object tags. Also, for scalability purposes, it should be low cost and easy to install, requiring little or no on-site calibration. What is more, for multiple reasons, many applications require the system to observe a certain level of privacy, regarding the people in the scene.

In response to these demands, our research pursues the development of lightweight motion-discriminative sensors that bridge the gap between *specialized* scalar sensors such as passive infrared (PIR) and *generic* array-based ones such as cameras. Our approach to this challenge is to explore biologically inspired address-event (AE) architectures that operate asynchronously at the pixel level to provide feature information instead of just images. AE sensors also have the advantage of being typically ultralow power. This paper describes a lightweight algorithm for people-counting and localizing given the output of a motion-discriminating address-event imager.

Our previous work in [1] provided an initial evaluation of new address-event imager architectures and a model for emulating such architectures on wireless sensor nodes to test AE algorithms. In this paper, we present and evaluate a design for localizing and counting people in indoor spaces with a set of wide-angle camera sensor nodes mounted to the ceiling, facing down. The initial results of this research have been published in [2], where we briefly introduced our people-counting and localizing network. In this paper, we expand on that work by providing further insight on the operation of the person-detector and utilizing an improved tracker on our experiments. Our design targets the architecture presented in [1] and localizes and counts people using a histogram derived from motion and size information. The resulting algorithm runs in real-time on iMote2 camera sensor nodes (Fig. 1) and is currently deployed on a home testbed for assisted living. The main contribution of the work described here is the design and evaluation of the lightweight histogram-based method for localizing people using motion and size information. By employing an address-event motion imager, the computational requirements of the histogram can be further reduced to operate on even smaller processors.

The results of this work are applied in the BehaviorScope project at Yale [3], which uses the information about people's movement in space to infer their behaviors, with assisted living as the driver application. Human locations collected from a wireless sensor network (WSN) deployed inside a house are processed in the context of a building floorplan to recognize the activities of the house inhabitants. The space-time locations of the inhabitants of a house during the course of the day provide

Manuscript received November 1, 2007; revised May 26, 2008. Current version published September 17, 2008. This material is based upon work supported in part by the National Science Foundation under Awards 0622133 and 0721632. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Hamid Aghajan.

The authors are with the Electrical Engineering Department Yale University, New Haven, CT 06520 USA (e-mail: thiago.teixeira@yale.edu; andreas.savvides@yale.edu).

Digital Object Identifier 10.1109/JSTSP.2008.2001426

a set of macro-gestures that are parsed by a framework of hierarchical probabilistic context-free grammars into a set of predefined activities [4][5].

The rest of this paper is organized as follows. Section II provides some background to the problem and surveys the related work. Sections III and IV outline our approach and describe the details of the motion histogram. Section V explains how a sensor node uses the histogram to localize and count, and Section VI presents our experimental results. Video demonstrations of our experiments are available at http://www.eng.yale.edu/enalab/behaviorscope/counting.htm. Section VII concludes the paper.

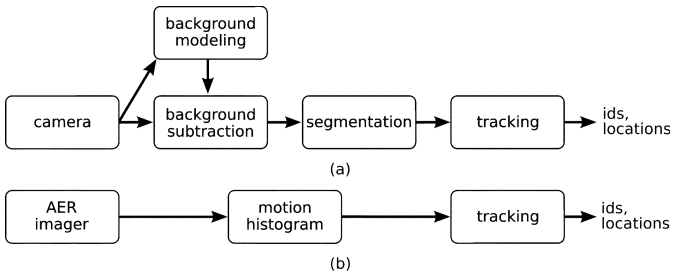

Fig. 2. Advantages of employing address-event (AE) imagers: since the computation starts at the pixel level, initial feature-detection steps can be skipped. Furthermore, the algorithms that operate on AE data are typically simpler.

II. Background And Related Work

There are many systems in the literature that aim to detect, count, localize, and/or track humans. Lately, most of them utilize cameras as the main sensing modality, but there are others that rely on more unusual modalities such as pressure sensors [6] and PIR arrays [7]. In [6], pressure-sensitive floor tiles are installed and consecutive footsteps are monitored to discern the tracks of multiple people. However, this approach requires a laborious installation process that makes it unfit for most existing environments. Meanwhile, the PIR arrays from [7] are used to count the people on a set of stairs. This is similar to [8], where a camera is employed but only for the information contained in its center scanlines. These types of approaches are used to count people at the entrance and exit points of closed spaces, which requires very few sensors. On the other hand, counting errors that occur at detection time end up propagating indefinitely.

As for more traditional camera-based human detectors, there are those who try to segment a human from an image by comparing it to an empty background frame, and those who directly employ some type of pattern matching. The patterns to be matched can be the eigenvectors given by the principal component analysis of a library of human images, or tuples of features such as SIFT [9], [10] and gradient orientation histograms [11]. Pattern-matching approaches depend on extensive training, and the feature extraction and matching processes can be computationally intensive.

The more typical approach is to employ background differencing followed by a series of morphological operations in order to obtain a workable silhouette of a person to be segmented (or "blobbed"). See Fig. 2(a). This silhouette can be used to confirm the blobbed object is indeed a human [12], or to determine the shape of the bounding box from where other features will be extracted for that purpose [13]. Since the low-level morphological operations do not guarantee that each person translate to exactly one blob, a further pass has to be performed where blobs that are close enough are merged together. The end result is that it is common to merge blobs that do not belong together, as well as to separate blobs that compose the same object, as in [14] and [15]. These algorithms usually perform additional steps after blobbing is done, in an attempt to correct such anomalies.

Some researchers utilize stereo cameras to assist in the image segmentation process, as is the case in [14]. In that paper, the authors describe their tracking system for assisted living. Their background model takes into consideration only the pixel intensity oscillations, and would fail in a less controlled environment. More importantly, their system does not handle rooms larger than the single stereo-pair's field-of-view.

Another approach [16] utilizes multiple cameras with largely overlapping field-of-views to get information about the 2-D cross section of a room, its objects and its occupants. The algorithm provides good location precision, but requires the use of multiple cameras to achieve that when covering even a small room. What is more, their approach demands precise calibration of intrinsic and extrinsic camera parameters. In the setting of assisted-living situations, however, the largest issue is that the computed cross section fails to capture a person that is sitting on a couch, lying down, or one that has fallen on the floor.

Finally, there is the problem of maintaining and updating the background model. This is a necessary process due to the presence of a series of change factors in a stream of frames, among which are:

1) natural oscillations in pixel intensity;
2) gradual changes in lighting, such as those imposed by the movement of the sun;
3) presence of repetitive background motion, such as waving foliage;
4) changes in position of static objects, such as furniture.

There are many algorithms that try and cope with these issues [17]. Scenario 1), for example, can be dealt with by modeling each pixel as Gaussian random variable and estimating their mean and standard deviation from a short calibration phase. However, this approach cannot cope with 2), 3) or 4), which require the background to adapt.

The simplest adaptive background-modeling technique is to continuously average all frames. This has the undesirable effect of generating "ghosts" in the areas where there is most activity, which ends up making the subtraction in those areas less reliable. Better adaptive background-modeling approaches are typically computationally expensive, sometimes modeling each single pixel as a mixture of Gaussians [18] or a Kalman filter [19]. Many of these approaches require the field of view (FOV) to be empty at initialization—something that may not be possible in the practical settings we are interested in. Even

then, in the presence of scenario 4), most approaches either fail or recover slowly. In assisted-living and office situations, though, these background changes occur very often. Take as an example the presence of office chairs, which are moved every time someone sits or stands.

In light of the aforementioned problems with background differencing, the work presented in this paper bypasses many of these issues by making use of frame differencing instead [Fig. 2(b)]. Frame differencing consists of subtracting the previous frame from the current one, to detect pixels that changed in intensity. The resulting frame is subsequently thresholded, resulting in a boolean image. This simple computation is robust in scenarios 1), 2), and 4) from the list, which are the most common in indoors deployments. This way, the complex background modeling steps become unnecessary, freeing system resources. Frame differencing, however, can generate images that are harder to segment. This is probably one of the main reasons why the computer-vision community has largely preferred background subtraction approaches. Our solution to this is to constrain certain aspects of our deployment in order to allow us to make simplifying assumptions. Namely, we place our cameras on the ceiling, facing straight down, and assume the ceiling height is known. As will be discussed in Section III, this lets us completely bypass these segmentation issues by employing a motion histogram. Another problem associated with frame-differencing is that people can only be detected while moving. In this paper, we make use of additional features to track stopped targets.

More importantly, our approach is built from ground up with address-event image sensors in mind. These sensors are biomimetic cameras that move the feature detection step into the imager's pixels, taking inspiration from the cornea. Much like neurons in the cornea, each pixel asynchronously emits a pulse (or *spike*) when an event is captured. By "event" it is meant *anything that can be measured*. In the specific case of the imagers for which we designed our motion histogram, an event is signaled whenever a pixel detects a certain intensity variation. The address-event representation (AER) protocol is then used for multiplexing all these spikes into an output data bus. For each incoming spike from pixel x_i, the AER encoder outputs the address of x_i onto the bus. The event magnitude information is not directly encoded. This differs from typical cameras, where the magnitude is the main unit of information, and where large arrays are transmitted regardless of whether the scene is of interest. In AE cameras, magnitude is naturally encoded into the event *frequency*. That is, in a motion-sensitive AE imager, a pixel that detects the most motion fires most often. Another distinction between these two types of cameras is that address-event cameras do not discretize time into *frames*, which leads to high-precision time measurements which can only be obtained by ultrahigh-speed cameras. Surprisingly, the power consumption of AE imagers is typically on the order of milliwatts or hundreds of microwatts [20], [21]. In this paper, we simulate the AE input through traditional imaging techniques. Due to this, in our current setup, we do not get the benefits of the precise time measurements of AE. What is more, for use with address-event hardware, our algorithm must be tweaked to operate at each incoming event, rather than on a frame-by-frame basis. This can easily be done with methods similar to those in [1].

III. Our Approach

Humans can recognize and count other humans based on shape, size, and movement. The background differencing approach attempts to extract and operate on mainly the first two types of information. We choose to focus on the latter two, while at the same time simplifying them by introducing a set of constraints on the deployment and the environment. First, we assume that people inside the room are typically in motion. Even though this does not *always* hold, it is certainly true for each person at some instant in time. Second, in order to cover a large area (requiring fewer sensors) and to minimize occlusions, we choose to place the cameras on the ceiling, facing straight down. In this configuration, and given the ceiling height, it is fair to assume that human size lies within a certain predefined range. Using these two assumptions, our goal is to classify as a human each image entity that meets our movement and size criteria and extract their discrete physical location from our measurements.

To this purpose, we construct a motion histogram from frame-differenced images and utilize that information to pinpoint each person's location. The histogram is designed to consider a typical human size in pixels, given the known characteristics of our camera and the ceiling height, and use it to compute the discrete human locations (histogram peaks) which best explains the moving pixels in the frame-differenced image. These locations can then be processed with higher level algorithms to track each person and recognize their behavior [4], [3], [5]. However, the unique labeling of each human and the association problems that arise are not the focus of this paper, but rather our lightweight sensing algorithm for human detection and localization.

IV. Motion Histogram

A. Overview

For simplicity, consider the 1-D case of detecting a person in the cross section of a background-differenced thresholded image. As shown in Fig. 3(a), the foreground pixels for each person are, ideally, all connected. However, this is usually not the case—especially for frame-differenced images, where above-threshold "motion" pixels are often sparsely distributed. The approach described in this paper gets around these issues by assuming that the size of a human is approximately known. This allows a motion histogram to be created, which is in effect an estimate of the unknown bivariate probability density function of the locations of moving objects. The locations which have the locally highest probability are then selected as human detections.

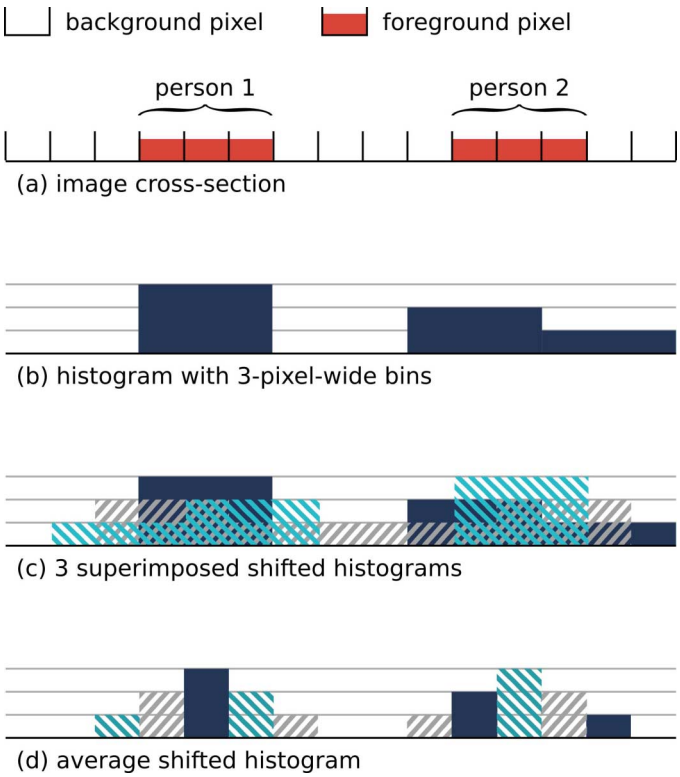

Fig. 3. (a) Cross section of thresholded background-differenced image with two people present. (b) Histogram with three-pixel-wide bins. People are detected at the histogram peaks, but the right-most person's position is ambiguous. (c) Superimposition of three histograms with same bin width but different bin origins. (d) Combining the three histograms from (c), an averaged shifted histogram is formed. Notice how the histogram peak is now a better estimate of the person's position. Since the origin shift was in 1–pixel increments, this last histogram looks like the result of a convolution, but other shifts may be used as described in the text.

Given that the ceiling height is known (i.e., the camera's z coordinate), the dimensions of the bounding box that encloses the image of a human can be approximated by some known average value. Let w be the average width of a human in the cross section of an image frame. Then a histogram may be produced by binning foreground pixels within bins of size w. This is pictured in Fig. 3(b), where $w = 3$. A mode-finding algorithm can then be utilized to discriminate each moving person in the scene. Although the histogram in the figure detects the first person's position very precisely, the position of the second person is ambiguous. This effect is a consequence of the particular choice of bin origin. Of course, in this case the second person's location can be better estimated by cleverly weighting each bin's coordinates according to the bin's values, but that would bring back the connectedness issues seen in typical image blobbing. A better approach is to simultaneously use multiple different bin origins. Thus, a single high-resolution histogram can be composed from multiple shifted histograms, as shown in Fig. 3(c) and (d). This type of histogram is called averaged shifted histogram (ASH).

The modes of the ASH in Fig. 3(d) are much better estimates of each person's location. Although the shifted histograms in Fig. 3(c) are shifted by $\delta = 1$ pixel, larger δs are often used. In the case when $\delta = 1$, the ASH becomes a convolution. Surely, the smaller the δ the higher the achieved resolution, since the worst-case peak location error for a histogram is given by $\delta/2$. However, when employing frame-differenced images, higher resolutions often produce histogram peaks that do not represent the modes of the underlying distribution. The reasoning behind this requires one to consider each thresholded pixel as a Bernoulli variable. In that case, the probability that a pixel x_e at the motion edge is above-threshold is $Pr(x_e) = p_e$, while that for a non-edge pixel x_{ne} is $Pr(x_{ne}) = 0$ (in an ideal noise-free scenario). Then, the motion histogram can be modeled by considering each bin as a binomial distribution binomial(n_i, p_e), where the sample size n_i is the number of edge pixels within the area B_i. Using these assumptions, the expected shape of the histogram for different values of δ can be found by assigning to each bin the expected value of its binomial distribution. For this "expected" histogram, the δ that gives the smallest person-localization error will indeed be $\delta = 1$. However, most *instances* of this histogram will display very jagged lines, which can easily produce false-positive peak detections. This situation can be dealt with by low-passing the histogram (using a Gaussian kernel, for example, or ASH weights) to smoothen these false peaks. However, that introduces the problem of choosing the best cutoff frequency so as not to drop valid peaks. This is where ASHs with $\delta > 1$ become attractive: intuitively, the large bin shifts produce an effect similar to a low pass at no extra cost, with the advantage that the ASH's parameters have a physical interpretation.

Another similarity between ASHs and convolutions is that ASHs may also employ different weights for pixels as a function of their distance to the bin center, in an attempt to further increase the histogram's accuracy. This is analogous to convolving the thresholded image with a given kernel—again, with the main difference being that the δ parameter is always 1 for convolutions.

To summarize, the motion histogram described in this paper is a bivariate ASH with uniform weights (which make the ASH comparable to a convolution with a square mask). The histogram is calculated over the absolute value of the difference between the current image frame and the previous. Moreover, as will be discussed in Section IV-D, when using wide-angle lenses our ASH is built from nonuniform bins, which are modeled after the different shapes people take as they move away from the optical axis. This effect is greatly accentuated when using wide-angle lenses, which is the case in our deployment.

B. Histogram Structure

The primary goal of the motion histogram is to determine the location of each person given the pixels that have changed the most compared to the previous frame. Consider, then, the subdivision of an image I into partially overlapping areas (Fig. 4). Each area B is a subset of the image. Assume, for simplicity, that the union of all such areas exactly cover the image: $\bigcup B_i = I$. Then a motion histogram H of dimensions $H_X \times H_Y$ is built where there is a bin $b = H[j,k], j \in H_X, k \in H_Y$, for each area B_i in the image. Without loss of generality, for the rest of this section we assume that the area centers are arranged in a grid, and that so are the histogram bins.

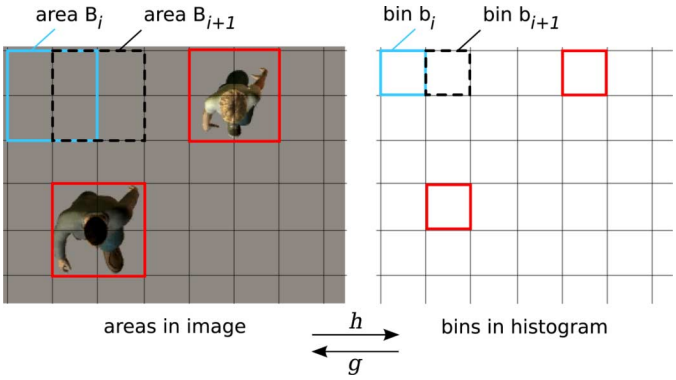

Fig. 4. (a) Histogram structure: histograms are composed of multiple bins defined from overlapping areas in the image (left). The bin size is calculated from human dimensions, and each bin can be uniquely identified by its top-left corner position. Using these positions, a more traditional representation of the histogram may be composed (right).

The value of each histogram bin b_i is the number of above-threshold pixels in the corresponding B_i. In Fig. 4, bin b corresponds to the blue area on the top-left side of the image. Therefore, the relation $g : H \mapsto 2^I$ can be defined, mapping each bin in the histogram to their respective set B_i of pixels in the image I. The notation 2^I is being used to denote the power set (i.e., the set of all subsets) of I. Thus, $g(b_i) = \{x : x \in B_i\}$. Conversely, $h : I \mapsto 2^H$ where $h(x) = \{b_i : x \in B_i\}$ gives for each pixel the set of bins that contain it.

If the bin areas on the left side of Fig. 4 are square with width w, and if the smallest distance between bin centers is δ, then g can be defined as

$$g(b) = \{x : x_x \in [b_x\delta, b_x\delta + w] \wedge x_y \in [b_y\delta, b_y\delta + w]\}$$

where b_x and b_y are the x and y coordinates of bin b in the histogram, and x_x and x_y are the coordinates of pixel x in the image. Similarly, the mapping h for the square-binned ASH described in the figure is:

$$h(x) = \left\{b : b_x \in \left[\frac{x_x}{\delta}, \frac{x_x - w}{\delta}\right] \wedge b_y \in \left[\frac{x_y}{\delta}, \frac{x_y - w}{\delta}\right]\right\}.$$

The relations g and h need not be as trivial as these, and better results may be extracted from irregular bins as we shall describe in Section IV-D.

C. Filling the Histogram

As explained earlier and depicted in Fig. 5, the histogram is filled using motion information from the difference of two consecutive frames. The algorithm for filling the histogram at each frame resets all bins to value 0, and then increments all bins that contain each above-threshold pixel. That is, given an above-threshold pixel x, we increment all bins in the set $h(x)$. The end result is that each histogram bin is assigned a value corresponding to the total number of foreground pixels it encompasses

$$b = |\{x : x \in g(b) \wedge x > T\}|$$

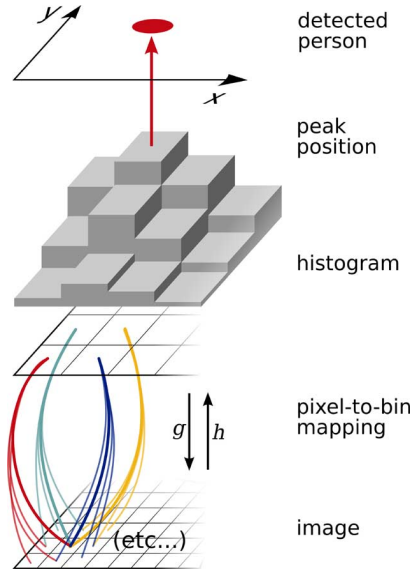

Fig. 5. Detecting positions from motion images: each pixel in the image is mapped to one or more histogram bins (as shown in Fig. 4). Bin values are incremented for each foreground pixel the bin contains. Histogram peaks detect people's positions. Note that, for simplicity, this diagram shows each bin connected to a different four-pixel area. In reality, however, bins encompass many more pixels.

procedure FillHistogram()

for each $b \in H$
  $b \leftarrow b \times \alpha$

for each $x \in I$
  if $x > T$
    for each $b \in h(x)$
      $b \leftarrow b + (1 - \alpha)$

Fig. 6. Pseudocode for histogram computation algorithm.

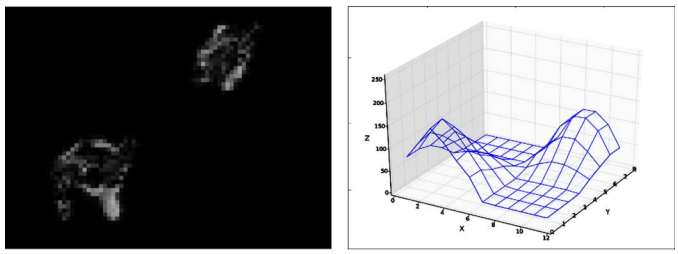

Fig. 7. Example histogram frame. (a) Moving pixels. (b) Resulting histogram. People are detected at histogram peaks.

where the vertical bars denote set cardinality, and T is the threshold on the intensity variation. The location of a person on the image plane can then be computed by running a peak-finding algorithm on the histogram.

For additional robustness against noise, the histogram-filling algorithm may be modified to incorporate information from previous frames. This is accomplished through the use of the variable α as shown in the pseudocode in Fig. 6. This way, each new histogram is superimposed on the previous histograms, and

$\alpha \in [0, 1)$ acts as the blending factor. Hence, the instance where the past histogram values are discarded is a special case of this, with $\alpha = 0$. Fig. 7 shows a histogram produced by this algorithm with $\alpha = 0.4$. An undesirable effect of employing a large α is that a person's peak will tend to lag behind the person's actual location, but we found that values smaller than 0.5 typically produce acceptable results in this regard.

Another measure against histogram noise is to incorporate a threshold after the peak-finding algorithm, such that only the peaks that are tall enough are utilized in people-counting and localizing. This value can be estimated by calculating the motion histogram for an empty scene, and choosing the mean bin height plus two standard deviations. Although this threshold is an empirical value, we find that it does not have to be a large number ($\sim 0.5\%$ the number of pixels in the bin) and this value should rarely require further adjustments.

The complexity of the histogram-filling algorithm in Fig. 6 is $O(I_X I_Y (w/\delta)^2)$, where I_X, I_Y are the image dimensions, and $(w/\delta)^2$ is the average number of bins that contain each pixel. The peak-finding algorithm we used in our experiments is $O(H_X H_Y)$, since it works by search for the local maximum points in each 5×5 area of the histogram. Given that the histogram dimensions are usually much smaller than image dimensions, the entire person-detection algorithm is bounded by the complexity of the histogram-filling.

D. Tuning the Histogram: Wide-Angle Lens Considerations

In the case where each bin maps to equal, but shifted areas in the image, the histogram can be seen as the result of the cross correlation of the image with a human model. In the simplest case, this model is a square, as in our discussion so far. Another possibility is to utilize a more complex function as a kernel, such as a multivariate Gaussian distribution. For the other types of models considered later in this paper, the histogram-producing operation will no longer be a cross correlation, since the kernel shape will vary with its position.

The type of model utilized has an immense effect on the performance of the histogram. This is an extension of the effects that are seen in a cross correlation: the breadth and height of the correlation peaks are the best when the kernel perfectly matches the image. In the case of the motion histogram, if the model is too small, multiple histogram peaks may appear for each person. If, on the other hand, it is too large, then the chance that two people incorrectly produce only one peak increases. Similar considerations must be made when picking the window-shift step size, δ: if the bins are too close, multiple bins may enclose the same person; if too far, the person may be missed entirely. These parameters are initially picked to match the average human dimensions in the described setup, then fine-tuned empirically (Section VI-A).

There are two additional effects that have not yet been accounted for, but which must be considered when building the histogram: perspective and lens distortion. Their effect is especially accentuated for wide-angle lenses and situations where the object distance is fairly small compared to its length (in the direction of the optical axis). Since a person is relatively large compared to the typical ceiling height [Fig. 8(a)], this must be taken into account for our setup. The top camera in the Fig. 8(b) produces very distinct images for each of the people depending on their distance from the center axis of the camera: people near the center of the image appear as seen from the top, while those at the edges are seen diagonally from the side. Hence, the square histogram bins yield good results for subjects near the center of the image, where there is an approximate top-view, but not so much as people wander toward the image edges.

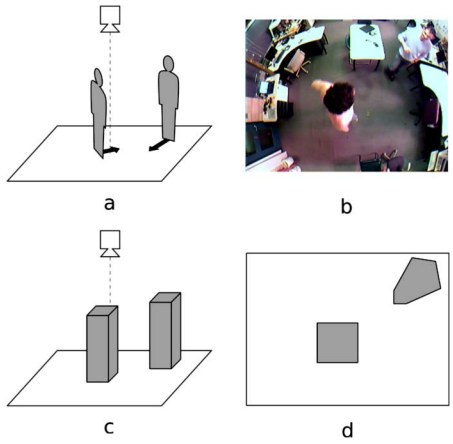

Fig. 8. Effect of perspective and lens distortion on histogram. (a) Ground-truth positions. (b) Image from top-view camera. (c) 3D bin model in the two locations that best match the ground-truth position. (d) Bins projected using camera and deployment parameters. These are the h mappings of the two peak bins of the motion histogram.

Accounting for this, a human model is derived from a 3-D object projected into the image plane using the camera's intrinsic calibration parameters. We take a rectangular cuboid as the 3-D model [Fig. 8(c)], with width and height taken from average human measurements. The model's image is calculated by applying geometric optics equations in conjunction with the Brown–Conrady distortion equations [22] to the coordinates of each of the cuboid's corners. The projections of the models onto the image plane are saved as a boolean bitmaps, which together make up the bin-to-image-area mapping g. The h mapping is computed in a similar fashion. The resulting bins provide a more accurate model as can be seen in Fig. 8(d). The motion histogram can, then, be constructed as follows: for each bin b_i in the histogram, the 3-D model is shifted by an amount δ and the value of g at b_i is mapped to the area B_i within the projected image of the cuboid: $g(b_i) = B_i$. Using this model, the histogram parameters w and δ can be measured in real-world units (such as cm) rather than pixels, making them more intuitive.

V. Camera-Node Level Counting

The motion histogram is designed to detect and locate moving objects. Thus, so long as none of the objects in the camera's FOV stop moving, the number of peaks in the histogram should correspond to the number of people in the room. However, when people stop, something must be done to keep their location consistent. For this reason, we further process with a standard tracking algorithm the locations detected with the motion histogram.

The tracking problem consists of labeling each detected object with the same unique ID in all frames where the object is present. However, for the purposes of this discussion, we relax the required length of the unique ID, since our tracker is employed mainly as a "stopping" detector. For this reason, if an ambiguous situation takes place (two people crossing paths, for example), the tracking algorithm is allowed to assign new IDs to each person involved in the ambiguity.

For these reasons, and given the power and computational limitations of sensor node platforms, the tracker we employ does not filter the sensor data with a Kalman or particle filter. Instead, the approach presented in the remainder of this section is based on bipartite graph matching, such as in [23], [24]. Compared to the lightweight tracker used in our previous work [2], our current tracker replaces the use of a histogram feature (peak height) with an image feature (color histogram). This aims to increase the robustness against "stopping," as is described later in this section.

Our tracker works as follows: at each time instant, the algorithm takes as input the set $Q = \{\vec{\mathbf{q}}_j\}_{j=0}^{m}$ of all peaks from the motion histogram and the set $P' = \{\vec{\mathbf{p}}'_i\}_{i=0}^{n'}$ of all people detected at the previous time step. Note that we denote variables from the previous time instant with a *prime*. The variable $m = |Q|$ is the number of peaks at the current frame, while $n' = |P'|$ is the number of detected people at the previous frame. A complete bipartite graph $G = (P' \cup Q, E)$ can be generated, where the weight of each edge is given by a function $w : P' \times Q \mapsto [0, 1]$. Then, the purpose of the tracker is to select a maximum weighted matching M of G, that is, to find the combination of peak-to-person assignments that globally maximizes a given similarity function w. The matching can be computed using the Hungarian method [25], with complexity upper bounded by $O(n^2m)$ (where n is the number of vertices and m the number of edges). The method described in [26] lowers this complexity to $O(n^2log(m) + nm)$. In this paper, we use the greedy method followed by the exhaustive approach to solve conflicts.

Each peak is represented by a vector $q = (q_x, q_y, q_C)$, and each detected person by $p = (p_{id}, p_x, p_y, p_C)$. These are the person's ID, their x- and y-location (in motion-histogram coordinates) and color histogram of the area in the image where the corresponding bin lies. This area is given by the bin-to-image-plane mapping g described in Section IV-B. The color histogram utilized assigns each pixel in that area into one of 32 bins. These features (location and color histogram) were chosen given their positive contribution to detection rates in similar trackers [24].

The weight function w is, then, defined as

$$w(p', q) = \beta\,\mathrm{Euc}((p'_x, p'_y), (q_x, q_y)) + (1 - \beta)\,\mathrm{Bhat}(p_C, q_C)$$

where $\mathrm{Euc}(\cdot, \cdot)$ is the Euclidian distance normalized by the maximum possible distance (image diagonal), $\mathrm{Bhat}(\cdot, \cdot)$ is the Bhattacharya coefficient, and β is an empirical constant in the interval [0,1]. At this point, an additional constraint is used: people are not allowed move a distance greater than $d = 4bins$ from one frame to the next for them to be properly matched. Given the w and δ of the histogram in our deployment, this allows people to move at a speed of at most 6.7 m/s.

The output of the tracker is the set P of detected people $p = (p'_{id}, q_x, q_y, q_C)$, for all $(p', q) \in M$. That is, each newly matched person is represented by their new position and color histogram along with their previous ID.

As described so far, the tracker assumes that the people in the image are always in motion. This is due to a limitation of the motion histogram, whereas people disappear if they stop moving. To handle this situation, at each time instant a *virtual peak*q^* is added to Q. This virtual peak has the special property that its location depends on which p' it is being matched against. When the weight of edge (p', q^*) is being calculated, q^* becomes (p'_x, p'_y, C^*), where the variable C^* is the color histogram of all pixels in the current frame that are in the set $B = g(H[p'_x, p'_y])$. Thus, each detected person is also compared to the area in the current frame that corresponds to their old location. The β used when weighting an edge of type $(\cdot, p^*)$ is chosen as a smaller number in order to emphasize the effect of the color histogram over that of the location (which was just copied from the detection on the previous frame).

VI. Experimental Results

The accuracy of a histogram is typically measured with a criterion such as the mean integrated squared error or the expected value of the L_1 norm. By minimizing this criterion with respect to the histogram parameters, one is able to find the optimal parameters for a given distribution. However, it is not clear what the distribution of above-threshold pixels for a moving human is. Without this type of information, we optimize the histogram parameters empirically as described in Section VI-A. In Section VI-B the motion histogram's resolution is tested given the parameters found in Section VI-A. Finally, Section VI-C will describe our initial results with a people-counting sensor network deployment.

A. Histogram Parameters

In order to find the best w and δ sizes, the histogram was calculated for a series of videos taken from a ceiling-mounted USB camera. Each video shows a constant number of people, all of whom are always in motion. The number of people in each video ranges from 0 to 5. For each frame where the number of histogram peaks does not match the number of people in the room, an error counter is incremented. The best histogram structure is chosen as the one that provides the least amount of errors (lowest valued counter). Fig. 9 shows the effect of varying w for a fixed $\delta = 15cm$. Meanwhile, Fig. 10 is the result of varying δ while holding w at 50 cm. Both plots were generated using the 3-D bin model, with the cuboid's height set to 170 cm. Since there were no false positives when nobody was in the camera's FOV, the plot for zero people has been omitted from the figure.

The plots in Figs. 9 and 10 show the detection capability of the raw histogram output, before using any tracking or employing any other features. In these conditions, the histogram peaks correctly detected the number of people over 60% of the time for $w \in [30\text{cm}, 50\text{cm}]$ and up to four people. The optimal bin shift is found in the interval $\delta \in [8\text{ cm}, 15\text{ cm}]$. The room where

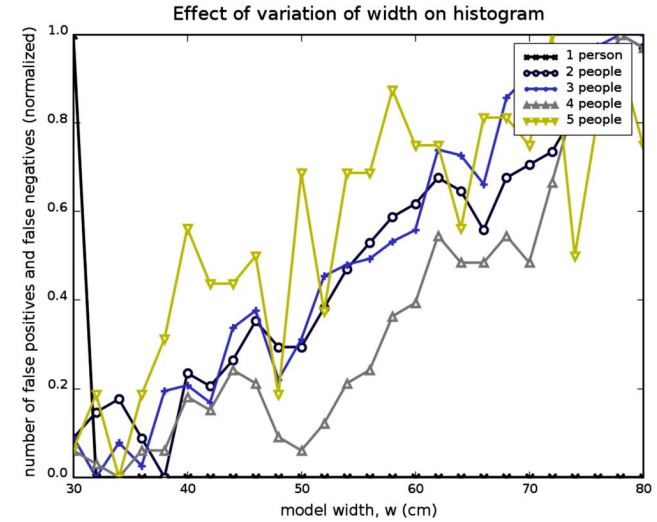

Fig. 9. Effect of varying the bin width w in the 3-D bin model. The value of δ was kept at 15 cm. The y-axis shows the number of detection errors, normalized for easier comparison between experiments with different numbers of people.

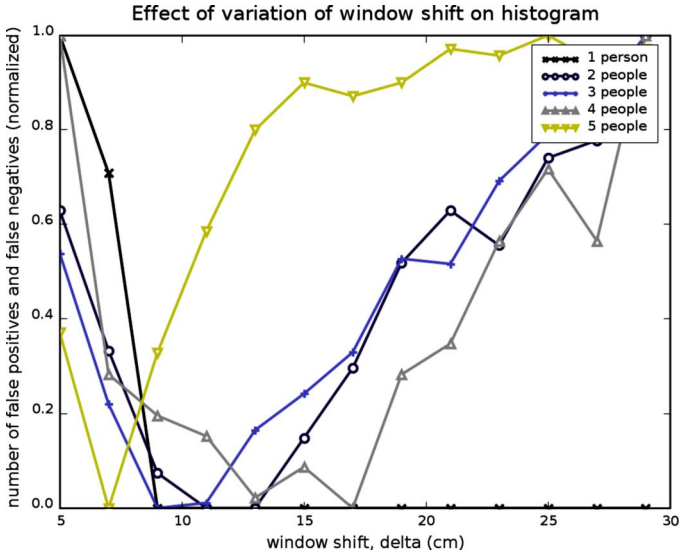

Fig. 10. Effect of varying the model shift δ in the 3-D bin model. The width was kept at 50 cm. The y-axis shows the number of detection errors, normalized for easier comparison between experiments with different numbers of people.

these experiments were performed has dimensions 9 m $\times$ 5 m, with a ceiling height of 3 m. The entire floor was covered by a single camera node with 162° wide-angle lens mounted on the ceiling. If the *usable* field-of-view is defined as the one where a person is seen in their entirety, then the dimensions get reduced to around 3.2 m $\times$ 2.4 m. The people in the room were asked to stay within those bounds, but, in the experiments with five people, they often moved outside due to space constraints. This is reflected in the plots, where the five-person detection results show much higher error rates than the others. Again, it should be emphasized that this is the *raw output* of the motion histogram. Normally, this is coupled with additional data modalities to further improve the results. This is done in the tracker explained in Section V by incorporating the color histogram information.

The shape of the plot in Fig. 10 reinforces the decision to use a histogram instead of a convolution. It is clear that small bin shifts are desirable given that they produce histogram modes with smaller localization error. However, those histograms are also very prone to false positives, as discussed in Section IV-A. This effect is shown in the figure, where error rates are high for small bins, then sharply decrease for midrange bins and increase again for larger ones.

For the rest of this paper, we chose $w = 50$ cm and $\delta = 15$ cm, since they are the largest values in the optimal ranges discussed above. By choosing the largest values we aim to increase robustness to false positives in situations where a person is not standing up. With these histogram parameters, the histogram partitions the image into 50 cm $\times$ 50 cm areas, shifted by $\delta = 15$ cm. Using these values, our tests where people are allowed to sit or lie down showed zero false positives, even though the cuboid utilized is a model of a person that is standing up.

B. Histogram Resolution

We tested the histogram's positional accuracy by having two people walk toward one-another and meet at the center of the image. This was captured by a single camera, in five different runs. The histogram was able to differentiate distances of up to 15 cm 100% of the time. This resolution greatly suits the assisted-living scenario, where the main interest is in the *logical spacial location* (such as "on the sofa," or "by the stove"), instead of more precise coordinates. The same test was performed for locations increasingly farther from the center. The result was the same for distances up to 1 m from the image center (66% of the usable area). At that distance, although the histogram at times produced a single peak for both people, the tracking/counting algorithm was able to disambiguate them. At the farthest position where one is fully covered by the camera (1.5 m), the algorithm missed around 42% of all detections. We believe there is room for improvement in those conditions, by utilizing a better tracker. The histogram achieves its best precision at the center two-thirds of the image, since when people walk closely and side-by-side near the image edges, occlusion often occur. Near the center, the maximal accuracy (15 cm, given that $\delta = 15$ cm) was achieved on five runs of parallel-walking tests (where two people walked side-by-side in different locations). Additionally, on the experiments where the two people crossed paths from different angles, the tracking algorithm was able to keep the correct count and locations regardless of distance from the image center. This is probably thanks to the small duration of the occlusions in this type of experiment. All experiments in this section were performed offline on a desktop computer, using a Python implementation of the algorithms.

C. Network Implementation

We implemented the motion histogram and counting algorithm in a sensor network composed of multiple Intel iMote2 sensor nodes. Each node is suited with a custom-built camera-board (Fig. 1) that contains an OmniVision OV7649 imager. The nodes acquire images at 320 $\times$ 240 resolution, downsample them to 80 $\times$ 60, then run the algorithm described in Fig. 6. The network was time-synchronized at boot-time using a broadcast

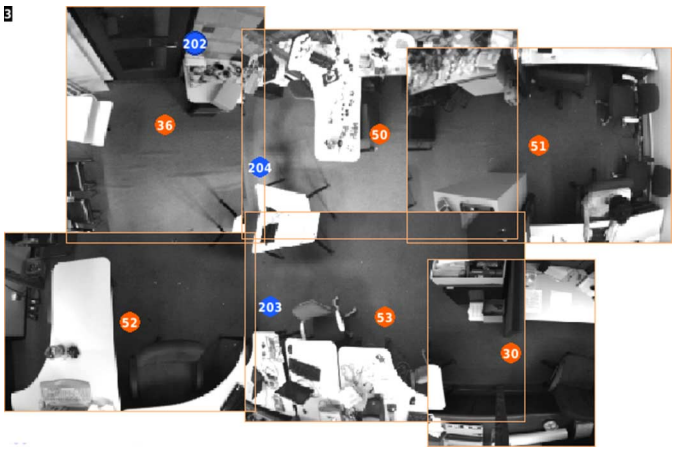

Fig. 11. Snapshot of the real-time visualization GUI: a composite of pretransferred images from all six nodes is used as the background where each detected person's locations is overlayed (blue circles). In the snapshot above there are three people in the scene. An orange circle is placed at each node's location, with the node IDs also displayed. The numbers on the blue circles are each person's temporary ID assigned by the tracker. The count reported by the base is shown in the top-left corner of the image.

from the base node. The mappings g and h are precomputed and kept in the node's memory for fast operation. For each frame, the number of detected peaks $|P|$ along with a timestamp are recorded into a small buffer. For visualization purposes, this buffer can be wirelessly transmitted to the base when full or after a timeout—whichever happens first. This allows people's positions to be verified in real time with a GUI on the client desktop (Fig. 11). This entire process repeats at a frame rate of ~ 14Hz when the processor is configured to run at 208 MHz. Local counts for each node are transmitted whenever they change. The base station aggregates the counts and reports the total count to the gateway computer.

The nodes are placed on the testbed structure on the ceiling of our lab, where they are a single hop away from their base, in a star topology. Given the ceiling height at the lab (240 cm) and the presence of cubicle walls, six nodes are required to cover the entire area. In this configuration, each node has a *usable* FOV of approximately 3 m $\times$ 2 m. The node positions are chosen to minimize field-of-view overlaps, and the images they acquire are cropped until the overlap is virtually zero. This way, we avoid most of the correspondence issues to focus on histogram and counting performance. Moreover, given the nonoverlapping FOVs, the system described here can be effortlessly mapped onto a tree topology. In this configuration, each node sums the count of its children, adds its local count and reports the value to the parent. This can repeat periodically (such as with [27], [28]), or only when the new local count differs from the previous, such as in [29].

For the experiment, people walked around the testbed through every node's FOV. Five runs were performed. The people were allowed to stop at will, as well as to sit down and stand up. Experimental runs were recorded with one to five people in the covered area. The results range from 92.9% correct detections when a single person was in the room, to 64.3% for five people, as shown in Table I. A close examination of the peak

TABLE I
NETWORK IMPLEMENTATION RESULTS

people present	correct detections
1	92.9%
2	88.6%
3	80.1%
4	71.6%
5	64.3%

location data has shown that most errors can be attributed to people temporarily walking between or outside FOVs. Occlusions started to become an issue as the number of people increased, but since most occlusions were brief, the tracker was able to correct the vast majority. Other sources of error are small offsets in time synchronization between sensor nodes, which at times caused the system to count the same person twice for the same wall-clock instant. Hence, better results can be immediately achieved by implementing a time-synchronization protocol such as [30]–[32].

VII. CONCLUSION

We developed an algorithm that uses a motion histogram to detect, count, and localize people. The algorithm is lightweight and operates in real time on wireless sensor nodes. Through the use of low-power AER motion cameras, the overall computational complexity can be further reduced, which in turn allows simpler processors to be employed, lowering the system's power consumption. As there is no calibration step that must be done on-site at the time of deployment, the system can quickly scale by just adding more nodes. The only calibration requirement is that the intrinsic parameters of the camera must be approximately known in the case where wide-angle lenses are used. When all lenses used are the same model, the intrinsic calibration only needs to be performed for one camera, and the parameters can be reused for all nodes.

We implemented the motion histogram and the tracking algorithm described in Section V on sensor nodes, which transmitted their detected person-count to a base for aggregation. The network requires nonoverlapping cameras, but a possible direction for future work is to exploit FOV overlaps for increased robustness to occlusions. A more extensive evaluation of the network and the sources of errors in the algorithm is currently taking place.

REFERENCES

[1] T. Teixeira, E. Culurciello, E. Park, D. Lymberopoulos, and A. Savvides, "Address-event imagers for sensor networks: Evaluation and modeling," in *Proc. Inf. Process. Sens. Netw. IPSN*, Apr. 2006, pp. 458–466.

[2] T. Teixeira and A. Savvides, "Lightweight people counting and localizing in indoor spaces using camera sensor nodes," in *Proc. ACM/IEEE Int. Conf. Distributed Smart Cameras*, Sep. 2007, pp. 36–43.

[3] D. Lymberopoulos, T. Teixeira, and A. Savvides, "The Behaviorscope System and its application to assisted living," Yale Univ., New Haven, CT, Tech. Rep. ENALAB 120407, 2007.

[4] D. Lymberopoulos, A. Ogale, A. Savvides, and Y. Aloimonos, "A sensory grammar for inferring behaviors in sensor networks," in *Proc. Inf. Process. Sens. Netw. IPSN*, Apr. 2006, pp. 251–259.

[5] D. Lymberopoulos, T. Teixeira, and A. Savvides, "Detecting patterns for assisted living: A case study," in *Proc. SensorComm*, 2007, pp. 590–596.

[6] T. Murakita, T. Ikeda, and H. Ishiguro, "Human tracking using floor sensors based on the Markov chain Monte Carlo method," in *Proc. 17th Int. Conf. Pattern Recognition*, Aug. 2004, pp. 917–920.
[7] K. Hashimoto, K. Morinaka, N. Yoshiike, C. Kawaguchi, and S. Matsueda, "People count system using multi-sensing application," in *Proc. IEEE Int. Conf. Solid-State Sens. Actuators*, Jun. 1997, pp. 1291–1294.
[8] G. Conrad and R. Johnsonbaugh, "A real-time people counter," in *Proc. ACM Symp. Appl. Comput. SAC'94:* , New York, 1994, pp. 20–24, ACM Press.
[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.
[10] Y. Ke and R. Sukthankar, "Pea-sift: A more distinctive representation for local image descriptors," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, 2004, pp. 506–513.
[11] N. Dalai and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, 2005, pp. 886–893.
[12] D. J. Lee, P. Zhan, A. Thomas, and R. Schoenberger, "Shape-based human intrusion detection," in *SPIE Int. Symp. Defense and Security, Visual Inf. Process. XIII*, 2004, vol. 5438, pp. 81–91.
[13] A. M. Tabar, A. Keshavarz, and H. Aghajan, "Smart home care network using sensor fusion and distributed vision-based reasoning," in *Proc. 4th ACM Int. Workshop Video Surveillance and Sens. Netw. VSSN'06*, New York, 2006, pp. 145–154, ACM Press.
[14] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer, "Multi-camera multi-person tracking for easyliving," in *Proc. 3rd IEEE Int. Workshop Visual Surveillance (VS'2000)*, Washington, DC, 2000, pp. 3–10, IEEE Comput. Soc..
[15] J. Owens, A. Hunter, and E. Fletcher, "A fast model-free morphology-based object tracking algorithm," in *Proc. British Machine Vision Conf.*, 2002, pp. 767–776.
[16] D. Yang, H. Gonzalez-Banos, and L. Guibas, "Counting people in crowds with a real-time network of simple image sensors," in *Proc. 9th IEEE Int. Conf. Comput. Vision*, Oct. 2003, pp. 122–129.
[17] D. Hall, J. Nascimento, P. Ribeiro, E. Andrade, P. Moreno, S. Pesnel, T. List, R. Emonet, R. B. Fisher, J. Santos Victor, and J. L. Crowley, "Comparison of target detection algorithms using adaptive background models," in *Proc. VS-PETS*, 2005, submitted for publication.
[18] C. Stauffer and E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 747–757, Aug. 2000.
[19] C. Ridder, O. Munkelt, and H. Kirchner, "Adaptive background estimation and foreground detection using kalman filtering," in *Proc. Int. Conf. Recent Advances in Mechatronics*, 1995.
[20] E. Culurciello and A. G. Andreou, "ALOHA CMOS imager," in *Proc. 2004 IEEE Int. Symp. Circuits and Systems ISCAS'04*, May 2004.
[21] P. Lichtsteiner, J. Kramer, and T. Delbruck, "Improved on/off temporally differentiating address-event imager," in *IEEE Int. Conf. Electronics, Circuits and Systems, ICECS*, Dec. 2004, vol. 4, pp. 211–214.
[22] D. C. Brown, "Decentering distortion of lenses," *American Society of Photo Grammetry, Annu. Conv.*, vol. 32, pp. 444–462, Mar. 1965.
[23] K. Shafique and M. Shah, "A non-iterative greedy algorithm for multi-frame point correspondence," in *Proc. ICCV*, 2003.
[24] M. Taj, E. Maggio, and A. Cavallaro, "Multi-feature graph-based object tracking," in *CLEAR*, 2006, pp. 190–199.
[25] H. W. Kuhn, "The Hungarian method for the assignment problem," in *Naval Research Logistic Quarterly*, 1955, vol. 52.
[26] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM*, vol. 34, no. 3, pp. 596–615, 1987.
[27] J. Hellerstein, W. Hong, S. Madden, and M. Franklin, "Tag: A tiny aggregation service for ad-hoc sensor networks," in *5th Symp. Operating Systems Design and Implementation*, 2002.
[28] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 122–173, 2005.
[29] R. Manohar and K. Mani Chandy, "Delta-dataflow networks for event stream processing," in *IASTED Int. Conf. Parallel and Distributed Computing and Systems*, 2004.
[30] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *SenSys'03: Proc. 1st Int. Conf. Embedded Networked Sensor Systems*, 2003, pp. 138–149, ACM.
[31] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *SenSys'04: Proc. 2nd Int. Conf. Embedded Networked Sensor Systems*, 2004, pp. 39–49, ACM.
[32] R. Solis, V. S. Borkar, and P. R. Kumar, "A new distributed time synchronization protocol for multihop wireless networks," in *IEEE Conf. Decision and Control*, 2006, IEEE.

Thiago Teixeira is currently pursuing the Ph.D. degree at Yale University, New Haven, CT. He received the B.S. degree in electrical engineering in 2003, the B.A. degree in mathematics in 2003 and the M.S. degree in engineering in 2005 from The Johns Hopkins University, Baltimore, MD, where he developed an address-event imaging sensor network.

In 2005, he joined the Yale Embedded Networks and Applications Lab (ENALAB) at Yale, where he has been since. His interests include wireless sensor networks, embedded systems, sensors, pattern recognition and data mining.

Andreas Savvides received the B.S in degree in computer engineering from the University of California, San Diego, the M.S. degree in electrical and computer engineering from the University of Massachusetts, Amherst, and the Ph.D. degree in the electrical engineering from the University of California, Los Angeles, in 2003.

He is an Assistant Professor in the Electrical Engineering and Computer Science Departments at Yale University, New Haven, CT. He is the founder of the Embedded Networks and Applications Lab (ENALAB) that specializes in the design and implementation of distributed sensor networks and smart spaces. His research is supported by an NSF CAREER award as well as other federal grants and industrial support. His current research is focusing on self-configuring sensor networks, particularly the ones that can be used to interpret human behavior in order to provide assistive services.