MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Similarity-based Analysis for Large Networks of Ultra-Low Resolution Sensors

Christopher R. Wren and David C. Minnen and Srinivas G. Rao

TR2005-003 July 2006

Abstract

By analyzing the similarities between bit streams coming from a network of motion detectors, we can recover the network geometry and discover structure in the human behavior being observed. This means that a low-cost network of sensors can provide powerful contextual information to building systems: improving the efficiency of elevators, lighting, heating, and cooling; enhancing safety and security; and opening up new opportunities for human-centered information systems. This paper will show how signal similarity can be used to calibrate a sensor network to accuracies below the resolution of the individual sensors. This is done by analyzing the similarity structures in the unconstrained movement of people in the observed space. We will also present our efficient behavior-learning algorithm that yields 90% correct behavior-detection in data from a sensor network comprised of motion detectors by employing similarity-based clustering to automatically decompose complex activities into detectable sub-classes.

Pattern Recognition 39(10), Special Issue on Similarity-Based Pattern Recognition

December 15, 2004: Submitted to KIMAS 2005

January 16, 2005: To be submitted to **Pattern Recognition**, Special Issue on Similarity-Based Pattern Recognition

April 2005: to appear in the IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems, KIMAS 2005

October 2005: significantly revised in response to reviewer comments.

January 2006: revised version accepted to *Pattern Recognition* Special Issue on Similarity-Based Pattern Recognition to be published by Elsevier, edited by Bicego, Murino, Pelillo and Torsello.

July 2006: Online version available at `http://dx.doi.org/10.1016/j.patcog.2006.04.009` Print version will appear in the October 2006 issue of Pattern Recognition: volume 39 issue 10.

1 Introduction

The occupants of a building generate patterns as they move from place to place, stand at a corner talking, or loiter by the coffee machine. These patterns leave their mark on every object in a building. Even a lowly carpet will eventually be able to tell you something about these patterns by how it wears. However, our automated systems are largely blind to these patterns. Elevator, heating and cooling, lighting, information, safety, and security systems all depend on humans to translate these patterns into action.

A network of sensors can sense these patterns and provide useful information to context sensitive systems in a building. However, these systems need to be cost-effective. We could attempt to extract detailed models of the activities occurring in the building and then reason about them to accomplish our goals. However, we cannot assume that we will have access to the high-fidelity sensors, or the large computing resources, that approach would require. Instead we strive for direct solutions: solutions that operate directly on the signals, employing similarity-based analysis.

This choice also allows us to develop algorithms that are applicable to a wide variety of sensors—not only cameras, but also what we call ultra-low resolution sensors (ULRS): passive-infrared motion detectors, ultrasonic sensors, and other one-bit-per-second devices. In Section 5, we present the results of experiments that validate this approach for building-scale behavior understanding. The experiments show that ubiquitous, simple sensors, combined with similarity-based analytic techniques, compare favorably to the more commonly employed sensor models.

This paper addresses two important problems. First, given no prior information and only the unconstrained motion of building inhabitants, how can a network calibrate itself? Second, given limited, indirect supervising data (possibly garnered from systems such as elevator scheduling computers or key-card systems), how can the network efficiently learn to recognize important behaviors?

Geometric calibration is important because so much of the context within a building is linked to location. Finding ways to make geometric calibration automatic serves two purposes. First, it reduces the installation cost of the network: there is no need to carefully document every sensor or carefully survey the equipment after installation. The network can be deployed in an *ad hoc* manner. Second, automatic calibration also helps make the network more robust by allowing it to adapt to changes in building configuration and to node displacement or replacement. We formulate an automatic-calibration algorithm by realizing that, in a network, sensors that are near each other will exhibit similarity in their data streams. We present our approach to geometric calibration, and our results, in Section 4.

The behavior of the occupants is a rich source of context in a building. Unfortunately, its complex structure is very difficult for a human operator to manually label. Behavior is unique to each building. It is often a time-varying process. All these things point to the need for automatic behavior-discovery.

Unfortunately, the literature on temporal clustering pays little attention to computational complexity. As a result, the classical algorithms that people have used to discover temporal patterns in unstructured databases do not scale at all to the data size and behavioral complexity that one expects to find in a large building. We present a modification to standard temporal pattern discovery algorithms that makes them significantly more efficient. This allows systems to tackle the huge quantity of information and deep richness of behavior that is present in such settings. We present our temporal clustering algorithm, and results, in Section 5.

We show that it is possible to extract rich descriptions of spaces observed by sensor networks, even if the network consists of sensors that are of very limited ability. By using similarity-based techniques and relying only on the patterns generated by the unconstrained occupants of the building, we show how to build low-cost systems that can adapt to a wide variety of settings.

2 Related Work

Wilson and Atkeson [21] also utilize a network of motion detectors in their work. Their system is targeted to the home, where they assume that only a few people will be present. This allows them to pursue a classic track-then-interpret methodology. More people means more ambiguity, and more ambiguity means exponentially more hypotheses that must be considered during tracking. This is only practical in low-census buildings, such as the home. Wilson and Atkeson also assume strategic placement of sensors. That level of specialization is not economical in large buildings or where usage patterns change regularly. We assume that our network will be built into the lights, outlets, and vents, and that it will therefore be installed by professional electricians and ventilation engineers, rather than behavioral or eldercare specialists. We propose our self-calibration method to address this issue.

There is a significant body of literature surrounding the interpretation of human behavior in video [16, 9, 11, 6, 13]. A common thread in all of this work is that tracking is the very first stage of processing. That limits the work to sensor modalities that can provide highly accurate tracking information in the absence of any high-level inference. In particular, the ambiguities inherent in using a motion-detector network can be expected to introduce enough noise in the tracking results to render most of these approaches unusable.

There are a few works that have attempted to step outside this framework [20, 8]. These systems learn task-specific state models that allow the behaviors to be recognized directly from similarity measures on sensor data, without tracking. Our work follows this philosophy and adapts it to the domain of sensor networks.

3 Experimental Design

We have instrumented $175m^2$ of office space with 17 ceiling mounted sensors. Each sensor has an active region that is a roughly $12m^2$ rectangle. The exact field of view of each sensor depends on the height of the ceiling and the presence of occluding objects. The sensors report motion events in their active area at 7.5Hz. They also respond to sudden changes in illumination. They adapt to novel, perfectly stationary objects and other changes in the environment after about 20 seconds.

3.1 The Experimental Domain

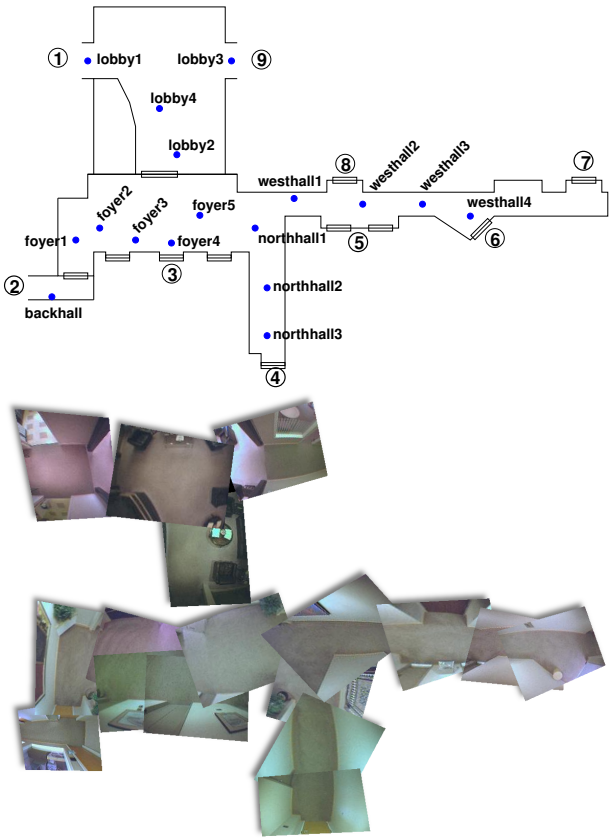

Figure 1: Top: The floor plan of the observed space. Named dots represent sensors. Circled numbers represent entrance and exit points. Bottom: A composite image from the sensor network.

The area covered consists of the high-traffic core of our building: the elevator lobby, reception lobby, restroom entrances, and hallways. See the area map in

Figure 1. Traffic through this area is high since it divides the office into east and west partitions. The east partition (through #1 and #2) contains a few offices and many common services: the kitchen, coffee machine, mail room, supply room, lunch room, and two meeting rooms. The west partition contains the vast majority of the offices and lab spaces. The elevators (at #3) are the only elevators that service this floor and are the principle entrance and exit for the majority of the occupants.

There are a number of places where people can enter and exit the observed space. Specifically: #1 leads to the kitchen, lunch room, and meeting rooms; #2 and #4 lead to offices and lab space; #3 is the bank of elevators, #5 leads to the restrooms; #6 leads to the library, the stairway to lower floor offices, and the administrative wing; #7 is a rarely used stairway; #8 is an alarmed emergency exit; and #9 also leads to the administrative wing. There is an unobserved path that connects #1 and #2, and another that connects #4, #6, and #9.

The area was chosen to test several potential uses of ULRS, including activity recognition and prediction, anomalous event detection, and population estimation and forecasting. Calling an elevator is one activity that happens in this area that is both economically interesting and well defined. Section refsec:models will, in particular, focus on the specific task of detecting and predicting presses of the elevator call button. We anticipate that the methods described below will also be applicable to the other tasks mentioned above.

3.2 Sensors

We define the ULRS as a generalized sensor that generates a stream of boolean values over time in response to human activity. For a given network of sensors, an arbitrary order will be imposed upon the sensor nodes so that they may be unambiguously indexed. The underlying representation of the data will therefore be an *event list*:

$$E_{i,t} = \{0, 1\} \tag{1}$$

A particular entry in the event list will be true if and only if there were a motion event detected by the ith sensor, during the tth temporal bin, where i is the unique index of a particular sensor in the network. Remember that these events are boolean and indicate merely the presence of some kind of motion anywhere in the field of view, but no indication of the number of people, the direction of motion, or any other such secondary information.

This sensor model admits a wide variety of sensor implementations including passive infrared motion detectors, break-beam sensors, pressure pads, and even cameras and microphones.

4 Recovering Geometry

In this section we show that we can recover the geometry of the sensor network simply by observing similarities across time in the patterns created by unconstrained motion of the occupants of a building [23].

We define similarity as the co-occurrence statistic between two streams of events: $C_{i,j,\delta}$. The co-occurrence is the count of events that co-occur at a given temporal offset:

$$C_{i,j,\delta} = \sum_{t=0}^{\infty} E_{i,t} E_{j,t+\delta}$$

where $\delta \geq 0$, and $E_{i,t}$ is a boolean value. Taken together, the $C_{ij\delta}$ for all possible δ are equivalent to the cross-correlation of the event lists for all sensors i and j. However, the entire cross-correlation is not useful and is very memory-intensive to compute, so we will only ever consider relatively small values of δ. In particular, we will focus on values of δ that correspond to the time-scales relevant to human behavior: seconds to minutes.

4.1 Relationship to Geometry

It is perhaps informative to think of these co-occurrence values in the context of audio localization [17]. In audio localization the time difference of arrival (TDOA) is estimated by finding the peak of the cross-correlation between two audio signals. This single number characterizes the spatial relationship between the audio source and the multiple receivers. Broadly, this paper attempts to take a similar approach. If we assume that the dominant activity in the space is movement through the space, then we can estimate the TDOA between various sensors in the system and that will provide information about the relative structure of the sensor network.

We will see in Section 4.3 that this assumption is at least valid in the office environment where we tested it. In particular, we accept that many of the co-occurrences in the counts above will be false co-occurrences, mere coincidence, and not an expression of a structured process (such as people moving about) that would affect the overall shape of the co-occurrence trace.

4.1.1 Position

If we have single-pixel sensors, for example a motion detector, then the TDOA provides only information about the distance between the sensors. Figure 2 illustrates this point. These plots are real co-occurrence plots from a hand-selected subset of the experimental sensors. The plots show the co-occurrence between the `lobby1` (L1) sensor and other sensors (`lobby4`, `lobby2`, `foyer5`, `northhall1`,
`northhall2`, and `northhall3`, as labeled on the left of the figure). For illustrative purposes, These plots have been manually sorted so that the sensor closest to `lobby1` is at the top, and the furthest sensor is at the bottom. The top plot shows auto-correlation of `lobby1` with itself. This plot is proportional to the likelihood of seeing an event from `lobby1` at time $t = T + \delta$, conditioned on the fact that we saw an event from `lobby1` at $t = T$:

$$P_{L1|L1}(\delta) = P(L1_{T+\delta} | L1_T = 1)$$

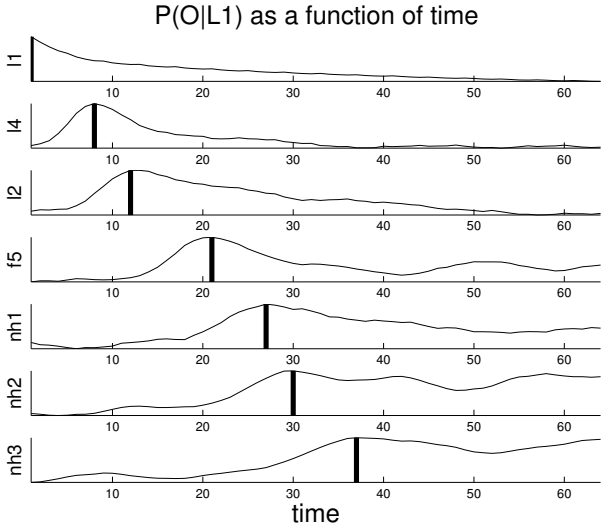

Figure 2: Plots of co-occurrence rate between the first camera and the cameras situated along a contiguous, hand-selected path through the space. Bars represent peak values.

This plot shows that, after seeing an event at the `lobby1` sensor, the probability of seeing a second event there appears to decay exponentially over time. The second plot shows that there is a more interesting relationship between `lobby1` and `lobby4`:

$$P_{L4|L1}(\delta) = P(L4_{T+\delta}|L1_T = 1)$$

this is the probability of seeing an event in `lobby4` some time after seeing an even at `lobby1`. The peak of this plot is around eight seconds and is marked by a black bar in the plot. The sensors `lobby1` and `lobby4` are very near each other. As we move down the figure, we see the conditional likelihood of other nodes: $P_{L2|L1}(\delta)$, $P_{F5|L1}(\delta)$, $P_{NH1|L1}(\delta)$, $P_{NH22|L1}(\delta)$, and $P_{NH3|L1}(\delta)$. These nodes are progressively further from `lobby1`, and we see the peak of the conditional likelihood moving to the right, representing longer and longer average transit times.

The peaks of these plots represent the maximum likelihood estimate of the TDOA between the sensors. We see that the auto-correlation, $P_{L1|L1}(\delta)$, peaks at the origin representing a preponderance of simultaneous co-occurrence. At the bottom the estimated TDOA is around 36 seconds: the time it takes a person to walk from the `lobby1` sensor across the space to the `northhall3` sensor at an average walking speed.

4.1.2 Orientation

Figure 3 illustrates a compound sensor with a known internal geometry between the five receptive fields. Given this kind of sensor, we should be able to estimate the relative orientations, as well as positions. We do this by exploiting the differential in TDOA observed in different receptive fields, relative to other, external sensors.

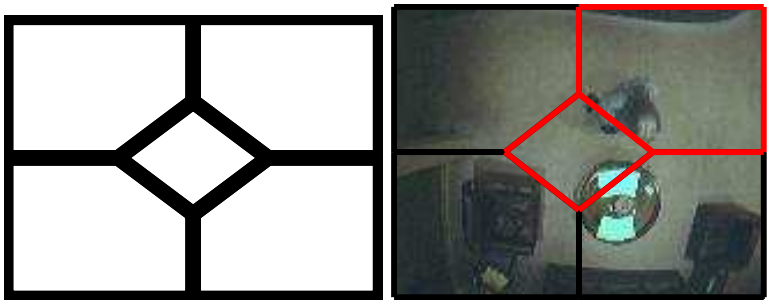

Figure 3: The layout of the foveal receptive field of the compound sensor and an example of a simulated sensor firing in response to a visual stimulus in the video.

If the sensor network is in fact comprised of cameras, as would be the case in a surveillance system for example, then the implementor is free to choose any receptive field geometry. Another possibility would be to trade off hardware cost for computational cost by affixing cheap motion detectors to the camera in a known geometry. These sensors could then directly provide the required event streams.

4.2 Mid-level Vision

The low-level processing of events produces a family of cross-correlation curves: N^2 traces for N sensors. By extracting the peak of each curve, we make an estimate of the N^2 TDOA constraints. There is significant structure in these constraints. They represent the underlying physical reality that ties the N sensors into a single network. An example of these estimates is illustrated in Figure 4.

We exploit these constraints by modeling them in a multi-dimensional scaling (MDS) framework. We refer the reader to the introduction to MDS by Kruskal and Wish [10]. We chose an iterative, gradient-descent approach to MDS where the similarities influence the springs between a set of masses, representing the nodes [7]. The TDOA constraints represent the preferred length of N^2 springs between the N masses. The system is represented by the equation:

$$x_{t+1} = Ax_t + b(x_t)$$

where x is the state of the system (the current estimate of the sensor positions

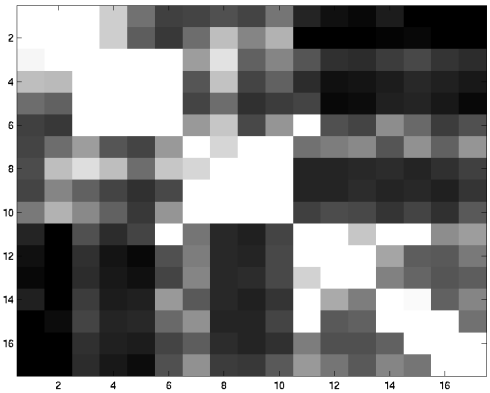

Figure 4: The matrix of TDOA estimates for the 17 sensor condition. White represents a low-TDOA, and black represents a high-TDOA.

and the velocity of the hypothetical masses), A is the system update matrix, and b is the control signal generated by the hypothetical springs based on the current state [2]. This system relaxes to a minimum energy solution that is the optimal estimate of the relative geometry given the constraints. This estimate is accurate up to a global scale, translation, and rotation.

To recover orientation, the N sensors are split into M receptive fields, and the MDS system models the M^2 constraints subject to the requirement that the internal geometry of the compound sensors remains fixed. The dimensionality of x and A do not increase, because the underlying degrees of freedom did not increase. Only the complexity of $b(\cdot)$ increases, with the addition of all the new constraints.

One problem with this framework is that early mistakes in TDOA estimation can have a large impact on the final solution. We find that the TDOA estimates that are the noisiest are the ones that are supported by the least observations: paths rarely taken. We can extract not only the TDOA estimate but a confidence value that is a function of the total number of observations that contributed to that estimate. This confidence is expressed in the model as the spring constant: a gain on the influence exerted by a particular constraint. Without this confidence value, all the spring constants in the MDS simulation are set equal to the same value. The results in the next section show the utility of weighting the estimation process with these confidence values.

4.3 Results

The simple, unweighted TDOA method works well for position. Since the MDS algorithm only recovers the relative geometry of the sensor network, we need to recover the affine transform that best maps the estimated network geometry

	Position	Orientation
Unweighted	2.4m	1.5 rad
Weighted	1.8m	1.3 rad

onto the ground truth geometry. We accomplish this with a standard minimum mean squared error estimator [5]. This is the configuration illustrated in the right-middle pane of Figure 6. The small symbols represent the true sensor positions, and the large symbols represent the estimated sensor positions after the affine transform. The true and estimated positions are connected in the plot by error bars.

Once we have recovered the affine transform, we can measure the accuracy directly simply by computing the root mean squared (RMS) distance error. For the unweighted TDOA method, we see a RMS error of 2.5 meters. Because our sensors observe $3.7m \times 4.9m$ rectangles, this figure is "sub-pixel" in the sense that it is significantly below the raw sensor resolution. When we leave out half of the sensors to create a subset with no overlap, we find that the error only goes up slightly—to 2.7m.

The middle row of Figure 6 shows the position error results. True position and estimated position are connected by error bars. It is possible to see that a large part of this error is due to the global warping of the whole network. This warping has particularly devastating effects at the edges of the network, where it pulls the ends of hallways and the far corners of the lobbies toward the center. By using the weights to discount poorly supported TDOA estimates, we can recover much more accurate estimates, as seem in the left-middle pane of Figure 6. This post-affine transform configuration represents an RMS error of only 1.8m. This lends credence to the idea that these errors are largely caused by noisy estimates of inter-sensor distances for which transitions are rarely seen, such as between sensors on opposite ends of the network.

It is difficult to evaluate the accuracy of orientation estimation when it is jointly performed with position estimation, since what appear to be erroneous orientation estimates may actually be consistent in the context of mistakes in position estimation. We therefore present results from orientation estimation in isolation: where the position of the sensors is assumed known. These results are presented visually in the bottom row of Figure 6. The true and estimated orientations are connected by error arcs. The bold lines indicate the estimated orientation of the sensor, and the thin lines represent truth. As shown in Table 4.3, the unweighted method achieves an RMS error of 1.5 radians. Since the receptive fields divide the sensor into quadrants, this number represents accuracy very close to the $\frac{\pi}{2}$ limit of the raw sensors ability to measure differences in orientation. By introducing the TDOA estimate weights, we improve this number to 1.3 radians, well below the discriminative power of the raw sensors.

It is interesting to note that, in the weighted case, much of the orientation error comes from a small number of outliers. These outliers are sensors that view doors with dampers, elevators doors, and one extremely high traffic junction.

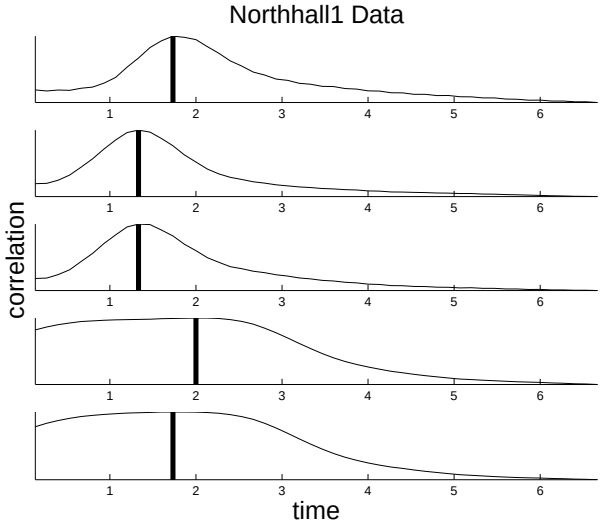

Figure 5: Plots of co-occurrence rate between the active fields of `northhall3` and its neighbor `northhall2`. See the text for a description.

Figure 5 illustrates the pathology of the sensors that is common to the sensors that view damped doors and elevators. The plots show the correlation between the five active fields of `northhall3` and its neighbor `northhall2`. The true geometry places the upper-left and upper-right fields closest to `northhall2`, but the presence of the damped door means that there are persistent, systematic processes that break the underlying assumption of uncorrelated noise. Every time a person passes through this hall, the door is opened and it then creates a stream of events as it slowly closes. This event is, by necessity, as common as someone passing through the door. This causes the wide plateau in the bottom two plots of Figure 5 that confound TDOA estimation despite ample observations. To overcome these errors one would need a significantly more detailed model of the underlying processes.

In order to overcome these errors we would need to take advantage of information contained in the auto-correlation signal, $P_{NH3,NH3}(\delta)$. Sensors that observe damped doors should have significantly different decay profiles, compared to other sensors. It might be possible to use deconvolution to restore the peaks in the smeared cross-correlation functions [3].

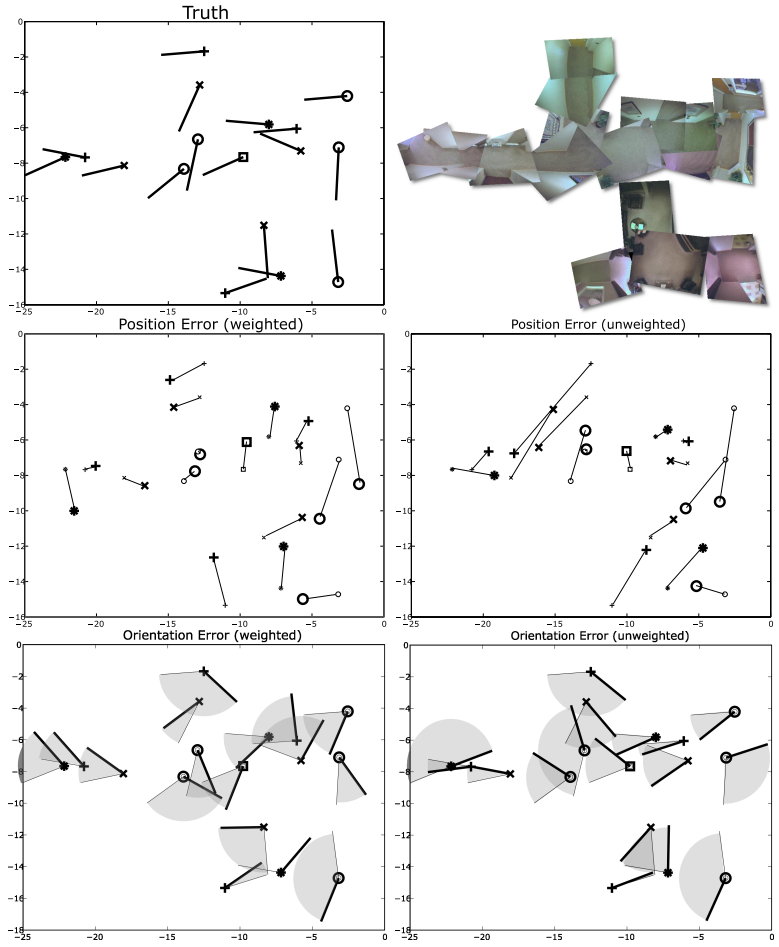

Figure 6: **Top**: The ground truth distance and orientation map (left) and the hand calibrated views from the sensors (right). **Middle**: The estimated (bold) positions of the sensors are connected by error bars to true locations (light), for weighted (left) and unweighted methods (right). **Bottom**: The estimated (bold) orientations of the sensors are connected by error arcs to true orientation (light) for weighted (left) and unweighted methods (right).

5 Modeling Behavior

Given a training set consisting of a collection of hypothetical observation sequences, $E_{i,t}$, and a list of activity labels from the same time period, A_t, our task is to estimate the probability that a new set of observations corresponds to a particular activity:

$$p(A_t|O_{t_0}, O_{t_1}, O_{t_2}, O_{t_3}, \ldots)$$

When $\cdots t_2 < t_1 < t_0 = t$, then the modeling system has the advantage of seeing the entire activity ending at the activity label A_t before making a decision.

The task of parametric, supervised learning in the domain of building-scale behavior recognition is made more difficult by the fact that there are several disjoint processes that result in the same semantic activity label. For example, an individual may approach the elevator button from the left or from the right. In both cases we wish to recognize the impending button press but the two behaviors leading up to the activity will appear very different to any perceptual mechanism. Manually labeling the different processes is fraught with human bias, and would, in any case, be impossible for an automated system that may only have one label available to it (the actual press of the button, in this case). One way to solve this problem is to take a semi-supervised learning approach where the learning algorithm is given the abstract labels but has the responsibility to derive structure directly from the data in a way that is usually associated with unsupervised approaches.

This section discusses a semi-supervised learning technique that recovers trees of Hidden Markov Models (HMMs). The trees are formed by factoring the similarity matrix generated by comparing the training sequences. In this way, the algorithm discovers the sub-behaviors that correspond to the labeled activity. This method is efficient enough to be used on tasks where behavioral complexity is high and training corpi are large. These are the two aspects that characterize the demanding building-wide context domain: rich with complex behaviors, but also rich in data to characterize them.

In the discussion below the assumption is made that the historical observations contain many correctly-labeled positive and negative examples of the activity to be recognized.

5.1 Semi-Supervised Learning

Given a set of observation sequences with the same label, we must find the structure within that class so that we may build efficient models. However, there is no satisfying way to directly compute the similarity of two sequences which may have different lengths. Smyth [15] suggests using HMMs to induce a similarity measure between sequences.

We create a similarity metric for sequences by training one HMM for each observation sequence to be compared. Since there are many more parameters in an HMM than can be trained with a single observation sequence, this amounts to simply initializing the HMMs so that their means and transitions optimally

describe the training sequence. The resulting models are very poor. However, they are good enough to provide a way to compare the similarity of sequences. The similarity between two observations, O_i and O_j,is then computed using the HMMs:

$$\S(O_i, O_j) = \frac{1}{2}[p(O_j \mid \lambda_i) + p(O_i \mid \lambda_j)] \tag{2}$$

where $p(O_j \mid \lambda_i)$ is the probability of generating the j^{th} observation from the i^{th} model. Intuitively, if the two observations are similar, then they should give rise to similar models which would then generate the opposite observation with high likelihood. A similar method was used by Wang et. al. [19] but they proposed a more complicated distance metric:

$$\begin{aligned} D(O_i, O_j) = \frac{1}{2}[\quad & \tfrac{1}{T_i}(P(O_i \mid \lambda_i) - P(O_i \mid \lambda_j)) + \\ & \tfrac{1}{T_j}(P(O_j \mid \lambda_j) - P(O_j \mid \lambda_i))] \end{aligned} \tag{3}$$

Empirical tests, however, showed that this distance metric did not perform as well as Smyth's simpler similarity metric.

Given this definition of a similarity metric, we form the similarity matrix **S** and use it to derive agglomerative clustering of temporal sequences, proceeding as follows:

1. Train an HMM for each of the N observations
2. Assign each observation to its own cluster
3. Compute the triangular matrix of similarities between all pairs of clusters
4. Find the two closest clusters, $A, B : A \neq B$
5. Merge these two clusters
   (a) Update the similarity matrix: $D(i, A) = max(D(i, A), D(i, B))$, for all clusters i
   (b) Remove the B^{th} row and column from the similarity matrix
   (c) Record that these two clusters were merged at this step to form the hierarchy
6. If more than one cluster remains, go to step 4

This procedure will create a full range of decompositions starting at N nodes for N sequences, and ending at a single node model. To determine how many clusters should be used, a separate analysis can be applied to this data structure to find a "natural break" in the clustering (i.e., a merge between significantly more distant clusters than the previous merge). Alternately, a predefined number of clusters can be selected. In the experiments described below, the number of desired clusters was chosen empirically by comparing generalization performance between composite HMMs with different numbers of paths. Note also

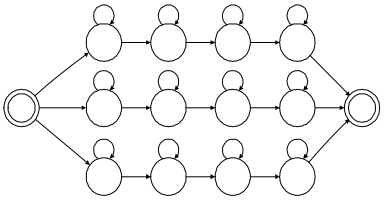

Figure 7: A composite HMM consisting of three path HMMs. The start and end state are non-emitting.

that, as defined above, the similarity between two models will *increase* as they become more similar.

Once the temporal clustering is finished, the M selected clusters are used as the dataset to train new HMMs. Each of these HMMs represents a different underlying process as identified by the clustering algorithm. However, these HMMs are trained after "hard" assignment, where the member function $\mu(O_i, \lambda_j)$ is defined thusly:

$$\mu(O_i, \lambda_j) = \begin{cases} 1 & \text{if} j = \arg\max_k P(O_i \mid \lambda_k) \\ 0 & \text{otherwise} \end{cases}$$

This means that each training observation is a member of one and only one cluster. It has been shown that in many cases "soft" assignment leads to better models due to similarity and ambiguity between classes [1]. Soft assignment allows an observation to be a partial member to many classes:

$$\mu(O_i, \lambda_j) = \frac{P(O_i \mid \lambda_j)}{\sum_k P(O_i \mid \lambda_k)}$$

To accomplish such probabilistic training, a composite HMM is constructed and then retrained on all of the data.

5.2 Composite HMMs

For the composite HMM illustrated in Figure 7, the transition matrix, T_{pq}, would be block diagonal, with each of the three paths contributing a block of non-zero transition probabilities between intra-path states while the inter-path transition probabilities would all be zero. This is the composite HMM structure proposed by Smyth [15].

Constructing the composite HMM is straightforward. Each cluster HMM becomes a distinct path through the composite HMM. This means that if each of the M cluster HMMs has s states, then the composite HMM will have $S = M \times s$ states, leading to an $S \times S$ transition matrix. The transition matrices of each path HMM are copied directly into the transition matrix of the composite HMM along the main diagonal. The prior state probabilities and final transition probabilities (i.e., the probability of exiting each path HMM) are then copied into

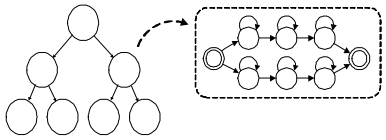

Figure 8: A tree of HMMs with three levels. Each node is composed of two HMMs that determine what subset of the incoming data is used for each child node.

the composite HMM and normalized. Finally, the observation distributions from the path HMMs are also copied into the corresponding states of the composite model. Once the composite model has been constructed, the standard Baum-Welch algorithm can be used to train the paths with soft observation assignments.

Although this algorithm is straightforward, it can be inefficient to retrain the composite model. The standard Baum-Welch algorithm has computational complexity on the order of:

$$O(N \cdot S^2) \tag{4}$$

per iteration, where the model has S states and is trained on N observations. Since the M path HMMs have s states each, training the paths independently should have complexity:

$$O(M \cdot N \cdot s^2) \tag{5}$$

For the full composite model, even though most of the state transitions are known to be zero, they still figure into the calculation, and the composite HMM therefore has the equivalent of the full $S = Ms$ number of states. Substituting Ms back into 4 and taking the ratio of that composite Baum-Welch complexity over the composite, single-path complexity:

$$\frac{\text{complexity(compositemodel)}}{M \cdot \text{complexity(pathmodel)}} = \frac{N \cdot S^2}{M \cdot N \cdot s^2} = \frac{N \cdot (M \cdot s)^2}{N \cdot M \cdot s^2} = M \tag{6}$$

We see that training a composite model composed of M paths will take at least M times longer than training the path HMMs individually. Furthermore, the composite model will almost certainly require many more iterations to converge due to the extra parameters to estimate, which will further increase the training time. Alon et. al. have developed a more efficient algorithm for training with soft assignments [1] that is similar to the Baum-Welch algorithm except that the probability of membership of each observation to each path is taken to be a hidden variable along with the typical hidden state memberships.

5.3 The Tree of Composite HMMs

Two factors motivate the use of trees of HMMs to automatically learn temporal clusters [12]. First, compared to the composite model described above, a tree of HMMs will require less time to train. Second, the tree can potentially decompose

the data more sensibly since each level need only split the dataset into two parts rather than M separate clusters. For datasets that exhibit natural divisions that match this model, we would expect to see an improvement in modeling performance as well as a gain in efficiency.

A tree of HMMs is defined here as a binary tree where each node contains two HMMs (see Figure 8). The purpose of the two HMMs is to model different temporal clusters or groups of clusters in the dataset, thus dividing the data into two distinct parts. Each of these parts is then sent to the child node corresponding to the HMM that better models it. This continued bifurcation of the dataset becomes fully decomposed at the leaves of the tree. It is important to note that the tree of HMMs is *not* a binary decision tree. Instead, it is a hierarchical model of all of the data. Thus, the leaf nodes do not represent decisions, rather they form the final step in the decomposition of the data into distinct clusters.

Each node in the tree will use time proportional to $O(ns^2)$ where each HMM has s states and is trained on n examples. Furthermore, the number of iterations needed for training to converge in each node is far lower than for the composite model, which also contributes to faster training.

5.4 Implementing Trees of Hidden Markov Models

There are a number of decisions that must be made when implementing the tree of HMMs describe above. During the training phase, there is the question of how to initialize the component HMMs. During evaluation there are several possible ways to calculate the probability of an observation. In accordance with empirical findings that we have omitted here, we use left-to-right initialization, evaluation with the Forward algorithm, and overall observation probability determined by the maximum leaf vote [14].

5.5 Hidden Markov Models for Event Lists

Remember from Equation 1 that the observations are boolean vectors with dimensionality equal to the number of cells in the system. Given that multiple cells may simultaneously observe an individual, and that we certainly cannot assume that the space will only be occupied by one person at a time, we cannot *a priori* assume that the cells will be active with any particular structure. That means that a distribution over these observations would have to account for 2^N possibilities for a system with N cells. This is intractable.

Since there is an underlying physical process behind the observations, we can assume, however, that there will be some inherent structure to the observations. To both reduce the dimensionality of this space and capture some of this structure, we will cluster the observations into a set of M alternatives where $M \ll 2^N$. Because we are clustering bit-strings, we use Hamming distance to define the clusters. These M alternative interpretations of the observation vectors are taken as the mutually-exclusive, independent alternatives produced

by a multinomial process. Figure 9 illustrates the 300 clusters found for a 5x5 network of cells overlaid on the `foyer3` camera.

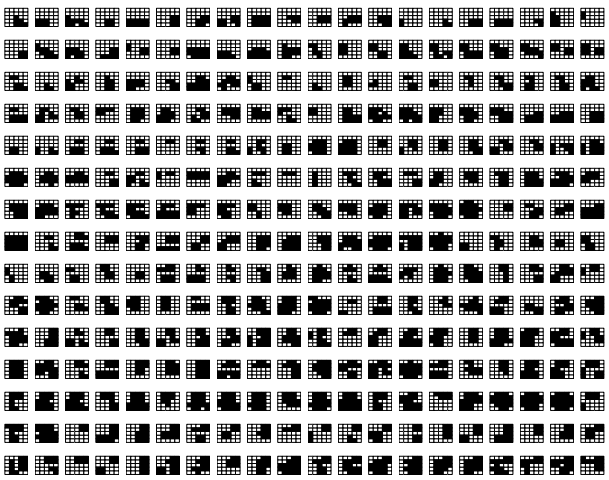

Figure 9: Examples of clusters from a 5x5 cell array on this camera.

In our implementation, both the Smyth and our Tree approach use this multinomial process model to formulate the observation probabilities in the component HMMs.

5.6 Performance Gains

In this section, the training and testing data were drawn from a pool of over 2.7 million observed sequences. Of those, 1956 were positive examples, meaning observations sequences that ended in an elevator press. The rest were taken as negative examples. The data was collected over four weeks. There were no constraints placed on the occupants of the space during that time. Much of the data contains multiple actors in the space at the same time. The algorithms randomly sample from these positive and negative pools to form training sets. The remainder are taken as a test set. The negative test set is also sampled to insure that the cardinality of the negative test set matches the cardinality of the positive test set. Multi-actor data is therefore likely to be part of both test and training data sets.

Both the classification performance and the computational complexity of our Tree method depend on the schedule of model complexity. Simple models at the root drastically reduce computational complexity but may cause over generalization early on that are detrimental to classification performance. The results presented here assume a fixed state schedule: 2, 4, 6, 8, 8, ... This means that the top-level models would have 2 states, the second-level models would have 4 states, and so on up to a maximum of 8 states per model.

Figure 10 shows the results of 20 runs of the Smyth and Tree algorithms

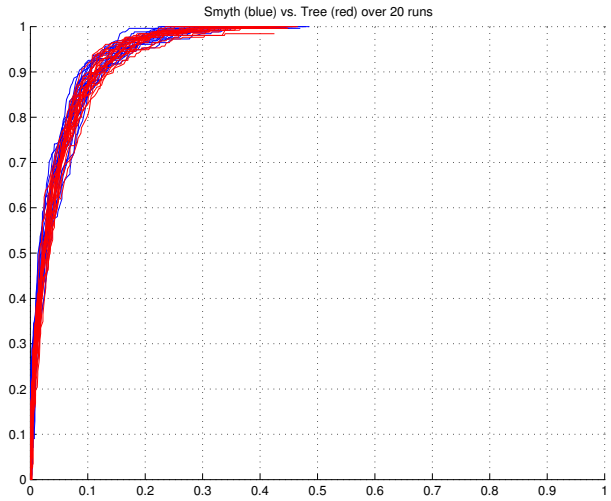

Figure 10: Randomized runs of the Tree and Smyth algorithms on the elevator task. The receiver operating characteristic curves indicate equivalent classification performance within the variance due to randomized initialization.

starting from different random initial conditions (including different random samples of the test and training sets). It shows that the two algorithms have essentially identical performance and stability for this task. The Alon method generates exactly identical results to the Smyth method.

The left side of Figure 11 explores the effect of dataset size on computational performance. We can see that Smyth is super-polynomial in data size: the plot is concave up on a log-log plot, so it's growing faster than any polynomial. We see that the Alon method is a significant improvement over Smyth. Our method is roughly twice as fast as Alon on this dataset. The right side of Figure 11 verifies that there is an advantage to using more data, and that we are not paying a significant classification performance hit for our approximations.

The left side of Figure 12 again shows that the Smyth algorithm is super-polynomial in computational complexity, here in relation to the model complexity. However, we see that the Alon method does not fix this problem: Smyth and Alon are both super-polynomial. The Tree method is much faster than both of the other methods. The right side of Figure 12 shows that there is an advantage to complex models. In fact, for larger spaces we would expect to see many more types of behavior, so this advantage is crucial for real systems.

5.7 An Examination of Context

We tested the above learning methods on the elevator-call task with a variety of sensor configurations and extent of context. The four cases all have roughly the same number of cells but use that perceptual bandwidth in different ways.

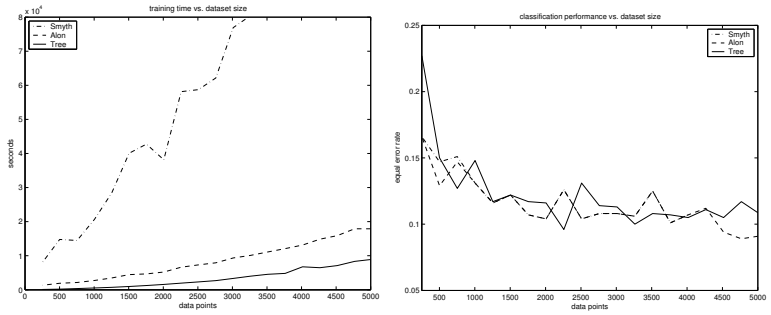

Figure 11: **Left**: The effect of training-set size on computation time (seconds) for learning. Plot lines are, top to bottom, Smyth, Alon, Tree. **Right**: The effect of training set size on classification performance. Smyth and Alon plots overlap.

`f3` : 25 cells from the camera immediately in front of the elevator.

`f3f5f2` : 9 cells from each of three cameras in the vicinity of the elevator.

`f1f2f3f4f5n1` : 4 cells from each of the 6 cameras observing the elevator lobby.

`all` : all 17 cameras in the system, with only one cell each.

The left side of Figure 14 shows the performance of the HMM Trees under these conditions. Two trees were trained for each condition: one for positive examples, and one for negative examples. The two trees were used in a likelihood ratio test framework to classify novel observation sequences. All the positive examples in the training set were used to build the positive tree. However, the overwhelming number of negative examples in the training set forced us to draw a relatively small number of samples to use in estimating the negative tree. It is possible that a bootstrapping mechanism for choosing the negative examples could result in better performance.

It is interesting to note that the `f3` case is the worst performer. This is counter-intuitive in the classification case because the `foyer3` sensor directly observes the elevator call button, and the 5x5 grid should provide quite a lot of discriminative power. For a prediction task one would expect that the very narrow context of `foyer3` would be a hindrance, but these results show that global context is important even for understanding the elevator call activity when observed in its entirety. In fact, the `all` case could be considered to give the best performance, despite the abysmal spatial resolution provided by the one cell per sensor geometry and the fact that more than half the sensors are more than 10 meters from the call button.

In an attempt to validate these startling results, we built another classifier based on a completely different clustering approach than is popular in the literature [8, 4]:

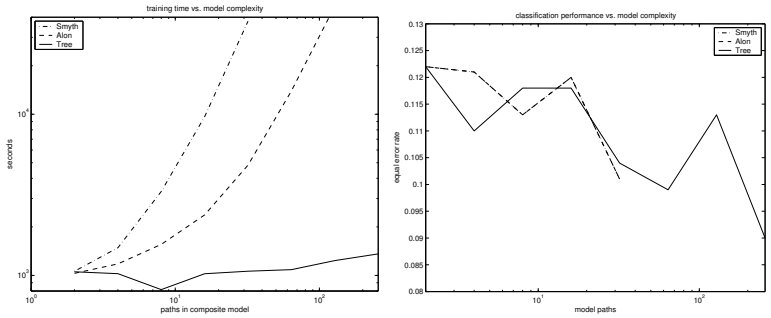

Figure 12: **Left**:The effect of model complexity (paths) on computation time (seconds) for learning. Plot lines are, top to bottom, Smyth, Alon, Tree. This is a log-log plot. **Right**: The effect of model complexity on classification performance. The overlapping Smyth and Alon plots both stop at 30 paths, due to run-time limitations.

1. Training data is arbitrarily segmented and randomly labeled.
2. HMMs are trained on the randomly partitioned classes.
3. Training data is re-labeled using the new HMMs.
4. Training iterates in this way, to convergence.

Given the cluster models, the training sequence is converted to a sequence of discrete labels. We call these data-driven clusters *loiteremes*—they are similar to phoneme models in speech processing. We build a classifier by taking a K-nearest neighbor approach where the distance metric is lexicographic order over the loiteremes sequences that represent a pair of observations. Given the extreme disparity between the prevalence of positive versus negative events, we say that the observation represents a positive event if there is even one positive event within the neighborhood of radius-K.

The results on the right side of Figure 14 were obtained using the KNN-Loitereme classifier, and they echo the surprising result above: that the worst performing condition, `f3`, is the situation that would intuitively seem to provide the most information about the defining moment of the activity—actually pressing the elevator button. Again, this would not be so surprising if we were evaluating prediction performance since the limited temporal horizon of the observations from this one sensor would be an obvious detriment. With this model the result that the `all` condition is the best performer is more clear: outperforming the other conditions across almost the entire ROC curve.

This is surprising. Two very different modeling approaches lead to the same conclusion: that the `all` condition provides some of the best contextual information for the classification task we tested. and the `f3` condition performs among the worst, despite being the intuitive choice for the detection task.

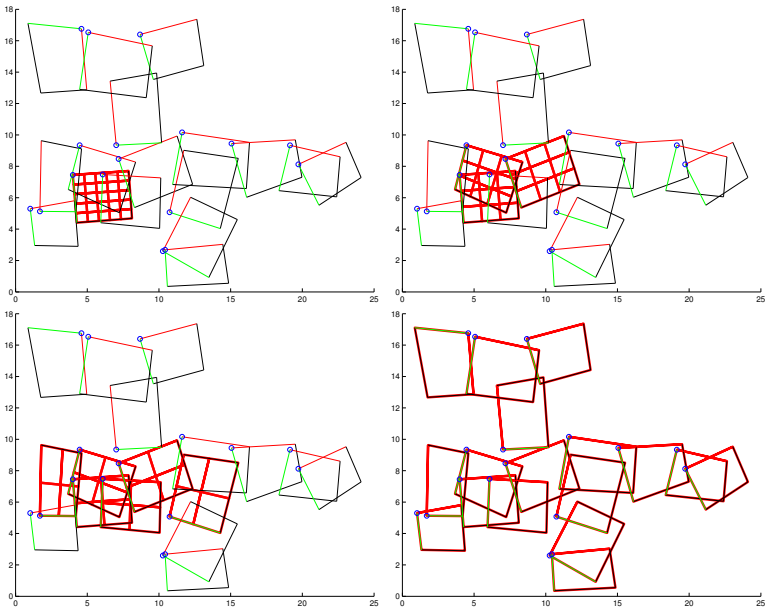

Figure 13: Four different context scenarios with equivalent total bandwidth: from left to right, and top to bottom, more context comes at the expense of larger receptive fields.

This indicates that the notion of global context for human activity understanding in buildings may have some real merit. The surprisingly good results in the `unsup-all` and `tree-all` conditions additionally indicate that having global context may be more important for activity understanding than the quality of the sensors.

6 Conclusion

Networks of ultra-low resolution sensors are a surprisingly powerful tool for building awareness. We have shown that similarity-based methods provide solutions for geometry estimation and behavior discovery in networks of simple sensors, such as motion detectors. This has profound implications for building automation and security. It is possible to perform useful, holistic perceptual tasks in buildings with inexpensive sensors, without extensive installation or maintenance expense, and without the need for invasive sensors.

A Appendix: Sensor Implementation

Even though cameras were used to collect the data, one of the goals of this work is to explore the utility of ultra-cheap, ultra-low resolution sensors. This

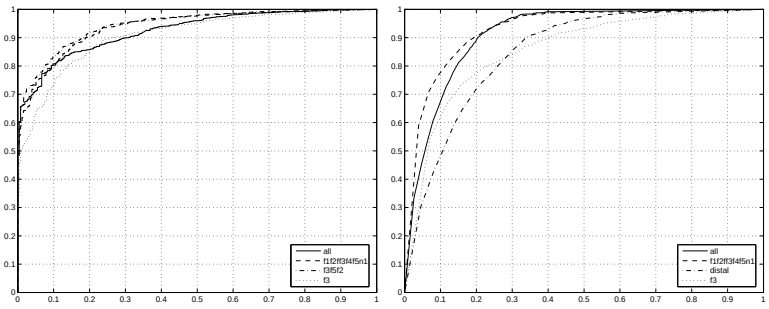

Figure 14: **Left**: HMM Tree learning results for different levels of context. **Right**: Performance of K-Nearest Neighbor classifier built on top of the HMM loitereme library for different levels of context.

section describes the intermediate representations we use that allow us to test our algorithms against an arbitrary set of hypothetical sensor configurations, including configurations that were not considered before data collection.

When the system detects motion it segments the foreground into connected components and records the first and second moments of each component [18]. If the moments are interpreted as the parameters of a two-dimensional Gaussian distribution, then these parameters specify the position, size, eccentricity, and orientation of a family of ellipses corresponding to the iso-probability contours of the Gaussian distribution. The upper-left frame of Figure 16 illustrates the 1σ, 2σ, and 4σ contours for the single connected-component recovered for that frame. Small components, occupying less than 5% of the frame, are dropped.

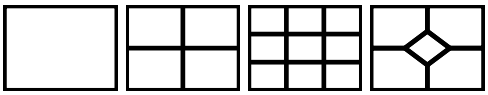

Figure 15: Some example sensor configurations that are used in this work.

The connected component models can be used to generate simulated sensor-readings by intersecting the probability contours with hypothetical sensor configurations. Figure 15 illustrates some of the sensor configurations that we have used in this work. The configuration on the left of Figure 15 is the simplest: it represents a single motion detector with a field of view that is equivalent to the video camera's field of view.The other configurations in Figure 15 are more complex, but within each sensor region the only information that is reported is the boolean presence or absence of motion. For example, the $2x2$ grid of sensors could be built from a collection of simple motion detectors with restricted viewing angles. Figure 16 shows the expression of some example geometries to a real stimulus.

Since these hypothetical sensor readings are computed from the intermediate motion component representation, and not from the video images themselves,

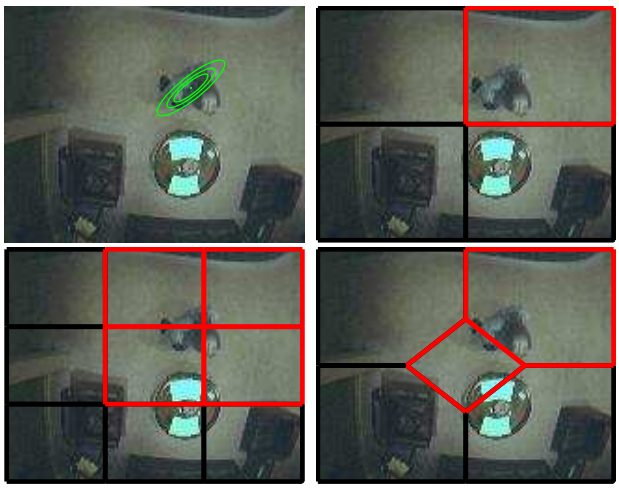

Figure 16: Some examples of sensors extracted from a video frame. The ellipses in the upper-left frame indicate the connected component of the foreground movement used by the cell extractor.

the process of generating a new set of hypothetical observations proceeds much faster than real-time. Generating a month of hypothetical observations typically takes significantly less than an hour for our experimental network.

References

[1] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering clusters in motion time-series data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 375–381, 2003.

[2] The Analytical Sciences Corporation. *Applied Optimal Estimation*, 1996.

[3] A Cichocki and S Amari. *Adaptive Blind Signal and Image Processing*. John Wiley & Sons, 2002.

[4] Brian P. Clarkson and Alex Pentland. Unsupervised clustering of ambulatory audio and video. In *Proceedings of the International Conference of Acoustics Speech and Signal Processing*, pages 3037–40, Phoenix, Arizona, 1999.

[5] A. Criminisi, I. Reid, and A. Zisserman. A plane measuring device. *Image and Vision Computing*, 17(8):625–634, 1999.

[6] Ross Cutler and Larry Davis. Real-time periodic motion detection, analysis and applications. In *Conference on Computer and Pattern Recognition*, pages 326–331, Fort Collins, USA, 1999. IEEE.

[7] Ricahrd O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2nd edition, 2001.

[8] Yuri A. Ivanov and Bruce M. Blumberg. Solving weak transduction with em. *Robotic and Autonomous Systems*, 39(3):129–143, 2002.

[9] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14(8), 1996.

[10] Joseph B Kruskal and Myron Wish. *Multidimensional Scaling*. Sage Publications Inc., 1978.

[11] David Minnen, Irfan Essa, and Thad Starner. Expectation grammars: Leveraging high-level expectations for activity recognition. In *Workshop on Event Mining, Event Detection, and Recognition in Video, held in Conjunction with Computer Vision and Pattern Recognition*, volume 2, page 626. IEEE, 2003.

[12] David C. Minnen and Christopher R. Wren. Finding temporal patterns by data decomposition. In *Sixth International Conference on Automatic Face and Gesture Recognition*, pages 608–613. IEEE, May 2004. also MERL Technical Report TR2004-054.

[13] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81:231–268, 2001.

[14] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–285, 1989.

[15] Padhraic Smyth. Clustering sequences with hidden markov models. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 648. The MIT Press, 1997.

[16] Chris Stauffer and Eric Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 22(8):747–757, 2000.

[17] P. Svaizer, M. Matassoni, and M. Omologo. Acoustic source location in a three-dimensional space using cross-power spectrum phase. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 231–234, Munich, Germany, April 1997.

[18] Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers. Wallflower: Principles and practice of background maintenance. In *ICCV*, pages 255–261. IEEE, 1999.

[19] Tian-Shu Wang, Heung-Yeung Shum, Ying-Qing Xu, and Nan-Ning Zheng. Unsupervised analysis of human gestures. In *IEEE Pacific Rim Conference on Multimedia*, pages 174–181, 2001.

[20] Andrew Wilson and Aaron Bobick. Realtime online adaptive gesture recognition. In *Proceedings of the International Conference on Pattern Recognition*, pages 111–6, Barcelona, Spain, September 2000.

[21] Daniel H. Wilson and Chris Atkeson. Simultaneous tracking & activity recognition (star) using many anonymous, binary sensors. In *The Third International Conference on Pervasive Computing*, pages 62–79, 2005.

[22] Christopher R. Wren and David C. Minnen. Activity mining in sensor networks. In *Workshop on Activity Recognition and Discovery*. NIPS, December 2004. also MERL Technical Report TR2004-135.

[23] Christopher R. Wren and Srinivasa G. Rao. Self-configuring, lightweight sensor networks for ubiquitous computing. In *The Fifth International Conference on Ubiquitous Computing: Adjunct Proceedings*, pages 205–6, October 2003. also MERL Technical Report TR2003-24.