Exploiting Semantics for Scheduling Data Collection from Sensors on Real-Time to Maximize Event Detection

Ronen Vaisenberg, Sharad Mehrotra and Deva Ramanan

School of Information and Computer Sciences
University of California, Irvine
Irvine, CA 92697-3425, USA
{ronen,sharad,dramanan}@ics.uci.edu

ABSTRACT

A distributed camera network allows for many compelling applications such as large-scale tracking or event detection. In most practical systems, resources are constrained. Although one would like to probe every camera at every time instant and store every frame, this is simply not feasible. Constraints arise from network bandwidth restrictions, I/O and disk usage from writing images, and CPU usage needed to extract features from the images.

Assume that, due to resource constraints, only a subset of sensors can be probed at any given time unit. This paper examines the problem of selecting the "best" subset of sensors to probe under some user-specified objective - e.g., detecting as much motion as possible. With this objective, we would like to probe a camera when we expect motion, but would not like to waste resources on a non-active camera. The main idea behind our approach is the use of sensor semantics to guide the scheduling of resources. We learn a dynamic probabilistic model of motion correlations between cameras, and use the model to guide resource allocation for our sensor network.

Although previous work has leveraged probabilistic models for sensor-scheduling, our work is distinct in its focus on real-time building-monitoring using a camera network. We validate our approach on a sensor network of a dozen cameras spread throughout a university building, recording measurements of unscripted human activity over a two week period. We automatically learnt a semantic model of typical behaviors, and show that one can significantly improve efficiency of resource allocation by exploiting this model.

Keywords: Video Surveillance, Algorithms, Sensor Network, Real-Time

1. INTRODUCTION

Consider a camera-based surveillance system deployed throughout a building environment. The task of the system is to process a typically immense quantity of data in real-time, possibly forwarding "suspicious" activities to a human for further inspection. We envision that multiple components are involved in this processing: low-level motion detection algorithms will identify specific regions and cameras of interest, camera sensors must record and transmit images to a central server, and finally, high-level object and activity recognition algorithms should identify suspicious movement patterns that are finally presented to human user. A vital aspect of this processing is that it must be done in **real time**.

Real-time restrictions are particularly limiting because of the enormous computational demands and the resource limitations imposed by a large-scale network. CPU resources limit the number of frames we can process, and image file sizes limit the amount of images that can be transmitted throughout a network. In this work, we consider a fundamental resource limitation due to network bandwidth: we can only probe K out of N cameras at each time instant, where $K \leq N$. Under such restrictions, we examine: what is the system's *optimal* plan of sensor probes?

Our intuition is that much of the behavior and activity in a building, even for the "suspicious" cases, follows certain low-level semantics - people enter and exit through doors, walk along hallways, wait in front of elevators, etc. We demonstrate that one can build a semantic model of such behavior, and use the model to guide efficient resource allocation for the K available sensor probes.

We look at the planning problem as a sub-task selection problem. The sub-tasks were selected to optimize an application-dependant **benefit function (BF)**. For example, a particular application may want all image frames for which there is motion (all motion events are equally important), while another application may define that two images of two different individuals are more important than two of the same person.

Another consideration is the **cost** of a plan, in terms of network resources, referred to as **cost function (CF)**. Different plans may not cost the same in terms of network resources since it may be less expensive to probe the same sensor at the next time instant. In a fully general model, one might also place the number of sensor probes K into the cost function.

Formally, we define a plan for N cameras to be a binary vector of length N that specifies which cameras will be probed in the next time instant. $Plan = \{C_i | 1 \leq i \leq N\}$, where $C_i \in \{0, 1\}$.

Our task is not only to know when and where motion was detected but to probe the cameras for frames for a surveillance application or a human operator. Motion is an underlaying primitive, that is used to select data that might contain interesting events.

To illustrate how the choice of sub-tasks affects a real-time tracking system, consider the following example. Suppose we have $N = 6$ cameras and budget of $K = 1$ probes per time instant. At time t, a frame processed from camera c_j reveals motion. Semantic correlation data collected off-line suggests that motion is expected to continue at c_j at probability $p_{stay}(c_j)$. Suppose further that at time $t - 3$, motion was detected at camera c_k. Semantic correlation data further suggests that given the motion was seen at c_k at $t-3$, motion is expected to be seen at camera $c_{correlated\_to\_k}$ with probability $p\_correlation(c_k, c_{correlated\_to\_k}, 3)$. This probability is a function of the time elapsed and the two cameras in order: from, to. At the same time, the semantic information suggests that motion can start spontaneously at c_{spont} with probability $p_{spont}(c_{spont})$.

These probabilities provide estimates as to where motion is expected (we will see how they can be computed shortly). The actual sub-task selection should be performed based on the specific benefit/cost functions defined. Note that decisions are made based on partial knowledge of where motion is since we only know what we have "seen".

Scheduling under resource constraints for data collection has been previously studied in a variety of contexts;[1,2] e.g., Markov decision processes, in sensor networks to create an accurate system state representation or for optimizing query performance.[3–5] Such work has also exploited semantics of the environment and/or phenomena being sensed to optimize resources (e.g., powering down sensors and/or reducing the rate at which they transmit). While such previous work is very related to our paper (and we discuss this in detail in Section 6), there are several aspects of our work that make it unique. First, we study resource constrained scheduling in the setting where sensors are video cameras which, to the best of our knowledge, has not been studied previously. The nature of cameras and activity these cameras capture provide new opportunities for optimization – e.g., exploiting cross correlation of motion amongst cameras to predict future motion. Second, our goal is to design a principled extensible framework that enables diverse semantics to be incorporated in making scheduling decisions – we illustrate the framework using three semantic concepts (all learnt from data): apriori probability, self-correlation, and cross-correlation. Finally, we illustrate the utility of the technique on real data captured using video cameras deployed at different locations of the building.

We believe that the sub-task selection problem is a central, and frequent, one in multimedia applications. Generally, there is a limit on how many processing tasks can be applied in real-time and an optimal subset of the computation tasks to be selected. In most of these cases, semantic information seems to be a promising way to efficiently direct the available resources.

In order to study the sub-task selection problem, we made several assumptions:

- **Semantics Available** are only of those of **motion**. Although more information can potentially be extracted as semantic information, we focused only on motion.
- **Semantic** system information is learned **off-line** with no budget constraints and is not updated while in operation.

- **Prediction Window** is $t = 1$. Although, it may be important to plan for the next t seconds* extending our approach to support this is part of our planned work for the future, and for this we assume that $t = 1$.
- **Homogeneity** assumption. In order to simplify the algorithms, we assumed that the cameras are homogenous in terms of probe costs. As such, the cost function was always constant. We are leaving the study of the effects of incorporating cost functions to future work.
- **Scheduling Decisions** are made by a (logically†) central node which knows the current state of the system and its history.

Push vs. Pull: Note that our approach is based on a pull-based model wherein data is probed from cameras based on a schedule generated exploiting semantics. An alternate approach could be for cameras to push data whenever an event of interest is observed (push-based model). While push-based models are interesting, they display a few shortcomings in our setting. First, push-based models depend upon cameras to have some computational resources (which is not the case in our infrastructure). Second, such models require basic event detection to be done at the camera which is achievable if events are well defined but more complex in settings such as surveillance where the interest may be in ill-defined events such as "abnormal" event. Of course, push-models could be based simply on motion detection but then scalability becomes an issue since, depending the usage of the monitored space, multiple cameras may detect motion together leading to network resource saturation. Even in our experimental set up, push based model (which we used for collecting training data) led to significant delays (about 30 seconds in some cases). This is specially the case during periods of high activity (e.g., in-between classes) when all sensors are likely to have motion. The delay caused due to network resources adversely affects the real-time nature of the application. Finally, push-based approach at the motion level is not as robust as the pull-based model we use since false positives such as lighting changes from doors swinging, etc. would result in data transmission. In this paper, we concentrate on exploiting semantics in a pull-based approach - a hybrid strategy based on combining both push and pull offers an interesting direction for future research.

As we proceed, we will address the following questions:

1. What are the options in selecting sub-tasks?
2. What kind of semantic information needs to be collected off-line, for the case of motion?
3. Which approach proves to work better on real-data?
4. What are the underlying reasons for the observed differences of sub-task selection strategies?

The rest of the paper is organized as follows: Sec 2 describes our system architecture and the semantics exploited in our framework. In Section 3 we introduce the models that we used to learn the semantics. Section 4 discusses different strategies for sub-task selection. Section 5 describes our implementation of the framework and presents accompanying results. In Section 6 we discuss related work and conclude with future work in Section 7.

2. OUR APPROACH

To study the sub-task selection problem, we postulated a simple model of nodes and tasks. A central process decided the subset of frames to be analyzed for each time unit‡.

To address the scheduling problem, the approach we adopt is to exploit semantics. While the framework we developed (which is based on probabilistic reasoning) is general and can accommodate any type of semantics, we focused on three different semantic concepts with which to illustrate and experiment. We then show how the system can be extended to accommodate other types of semantics as well.

The semantics of interest included:

1. **Apriori** knowledge of where motion is likely to be. Such semantics can be learned from the data by simply detecting which camera is more likely to have an event occurring. In the case of the building, it is likely that the camera at the front door will see more motion than other cameras. Also, cameras in front of classrooms (or meeting facilities) are likely to observe events more often than others.

*For example, due to network delays, deployment has to be done in advance.

†More on this in sec. 7

‡We chose second, but other scales were possible to discrete time.

2. **Self correlation** of camera stream over time. Given that a camera observes an event and given the camera's field of view (FOV), one could predict the probability that the event will continue. Of course, this is a camera specific property since it depends upon FOV as well as how individuals in the FOV use the space. For instance, a camera focussing on a long corridor will have a person in view for a longer period of time compared to a camera that is focused on an exit door.

3. **Cross Correlations** between cameras. Clearly a person who exits a FOV of one camera will be captured by another depending upon the trajectory of the individual and the placement of the cameras. Given that a person exits the FOV of camera A, there is an implied probability that he/she will enter into the FOV of camera B.

Given the above, our approach to scheduling is as follows:

1. We learn the above semantics from data.

2. We develop a framework, where given the state of the system currently, we project the state in the near future by determining probabilities of event (motion in FOV). These probabilities are based on all the three semantic concepts above. The three main modules of our framework are:

- **Task Processing Module-** Takes as an input a set of cameras to probe (query plan) performs the querying of the frames from the cameras and returns a list of cameras where motion was detected.
- **System Semantics Module-** Uses a pre-generated log, containing list of cameras and time at which motion was detected at each camera and outputs semantic information in a form of correlation matrix which is used by the scheduler.
- **Scheduling Module-** Uses the current state of the system, i.e., the cameras at which motion was detected, semantic information, BF - Benefit Function, CF - Cost Function and k - Constraint on the number of sensors to probe and returns a list of k cameras to query (a query plan) which maximizes BF and minimizes CF.

The **semantic module** computes the probabilities of the following three events predicted to occur in the future:

(a) $E(apriori(i))$ motion starts at sensor i. (b)$E(Self.Correlation(i,p))$ motion continues at sensor i, given that there was motion with probability p, one second ago. (c) $E(Cross.Correlation(i,M))$ motion at sensor i, given M - a matrix containing the probability of motion for all sensors for the last t seconds.

Based on the probabilities of the above events, given by the **semantic module** and the current state of the system, detected by the **task processing module** the **scheduling module** generates a schedule for the next time unit under the constraints, that minimizes CF and maximizes BF.

We observe that this probabilistic framework is very general. If we had some other semantics we could further incorporate it as long as we can map the semantics to a conditional probability between events over time.

Later on, we will offer other examples of semantics which may be useful. The key technical challenge and contributions of this paper were as follows:

1. Techniques to learn probabilities from past (as well as current) data.

2. Techniques to schedule based on probabilities. The key issue here is efficiency of computing the schedule without a complex probabilistic analysis.

The next section describes our learning algorithms and the scheduling algorithm followed by an experimental setup, testing and validation.

3. LEARNING ALGORITHMS

The following sections describe the models that were constructed in order to capture semantics of motion. All of the semantics extracted were learned from training data (see experimental setup section for information about how the data was collected). In the following sections, subscript will denote time while super script will denote camera.

3.1 Apriori

Arguably the simplest level of semantics exploits the fact that certain cameras are more likely to see motion, and so they should be probed more often. We compute the probability of seeing motion at a camera from training data as follows: we divide the number of time instants where motion was observed by the total time period.

3.2 Self-Correlation

Assume the state of a camera at time t is $s_t \in \{0, 1\}$, signifying a motion event or the lack thereof. Let us call the observed motion reading from a camera at time t as $o_t \in \{0, 1\}$. We assume a *Hidden Markov Model* (HMM)[6] of the state and observations

$$P(o, s) = P(o|s)P(s) \tag{1}$$

$$= P(s_1) \prod_{t>1} P(o_t|s_t)P(s_t|s_{t-1}) \tag{2}$$

The parameters of the HMM are (SC, B, π) where

$$SC = P(s_t|s_{t-1}) = \begin{bmatrix} a & 1-a \\ 1-b & b \end{bmatrix} \tag{3}$$

$$B = P(o_t|s_t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4}$$

$$\pi = P(s_o) = \begin{bmatrix} \pi_1 & \pi_2 \end{bmatrix}^T \tag{5}$$

For simplicity, we use a zero-noise observation model (Eq.4), though we could learn these values as well. Here, $a = P(s_t = 0|s_{t-1} = 0)$ and $b = P(s_t = 1|s_{t-1} = 1)$.

Inference (computing $P(s_t)$): Exact inference in an HMM can be performed with the forward-backward algorithm.[6] Since we are concerned only with the current state for online scheduling, we need only to implement the forward stage. This can be implemented with a *prediction* step that predicts $P(s_t)$ from $P(s_{t-1})$ using the dynamic model A, and a *correction* step that refines $P(s_t)$ using the observations o_t (and B) if available. **Learning (estimating SC, B, π):** One can learn model parameters from noisy data using the Expectation Maximization algorithm.[6] Since our observation model is noise-free, we can learn model parameters directly by frequency estimation. For example, b from Eq.3 is set to the fraction of times one observed motion in a camera given there was motion in the previous second (from training data).

3.3 Cross-Sensor Correlation

We would like to capture the intuition that, given that motion is observed in a particular camera, the scheduler should probe other cameras that lie on typical walking paths emanating from that camera. A important consideration is that it can take many seconds, even minutes, to move between cameras in a typical network. Hence a first-order markov model no longer applies.

For ease of exposition, we first describe a first-order multi-sensor model, but will extend it subsequently to a M^{th} order model that looks behind (or alternatively, looks ahead) M seconds.

3.3.1 First Order Cross-Sensor Model

We would like to capture interactions between N cameras. For now, only consider interactions one second later. Let us write the state of a camera i at time t is s_t^i. The overall state is $s_t = \{s_t^1, \ldots, s_t^N\}$, while the observations are $o_t = \{o_t^1, \ldots, o_t^N\}$. The observation model is naturally written as

$$P(o_t^{1:N}|s_t^{1:N}) = \prod_{i=1}^{N} P(o_t^i|s_t^i) \tag{6}$$

where we use the shorthand $o^{1:N}$ for $\{o^1, \ldots, o^N\}$. We use superscripts to denote cameras, and subscripts to denote time. Let us write the transition model at $P(s_{t+1}^{1:N}|s_t^{1:N})$. The first assumption we will make is that motion events occur independently at each camera

$$P(s_{t+1}^{1:N}|s_t^{1:N}) = \prod_{i=1}^{N} P(s_{t+1}^i|s_t^{1:N}) \tag{7}$$

This model is an instance of a Dynamic Bayes Net (DBN). It consists of N separate HMMs, where the state at time t depends on all N previous states (a *coupled HMM*). The conditional probability table on the right-hand-side has 2^N entries, so a naive representation will require both exponential storage and time $O(2^N)$ when making predictions. Clearly this does not scale.

Let α_{ij} be the probability that a person moves to camera j in the next timestep given they are currently at camera i. If we assume that people move independently throughout our camera network, we want to intuitively "add" such probabilities across i to compute $P(s_t^j)$. We use a *noisy-OR* model to do this.[7] Formally, let us compute the binary state with a logical OR: $s_t^i = c_1 \wedge \ldots \wedge c_N$ where c_j are equivalent to s_{t-1}^j but are randomly flipped to 0 with probability $(1 - \alpha_{ij})$. The transition model now simplifies to

$$P(s_t^i|s_{t-1}^{1:N}) = 1 - \prod_{j=1}^{N} P(c_j = 0) \tag{8}$$

$$P(s_t^i|s_{t-1}^{1:N}) = 1 - \prod_{j=1}^{N} (1 - \alpha_{ij} s_{t-1}^j) \tag{9}$$

The above model has $O(N^2)$ parameters, and so scales much better than a naive representation.

Learning: Assuming noise-free training data makes the hidden states observable, and so we can learn α_{ij} parameters by frequency counting as before.

Inference: Exact inference in a coupled HMM is intractable because the probabilistic model contains cycles of dependance.[7] Nearby cameras will tend to develop correlated states estimates over time, meaning that the joint $P(s_t^{1:N})$ does not factor into $\prod_i P(s_t^i)$. Due to real-time considerations it is difficult to employ typical solutions that computationally expensive, such as variational approxmations or sampling-based algorithms.[1,2,8,9] One natural approximation is to assume that the states at the previous timestep are independent, and "naively" apply prediction/correction updates (Eq.9) to compute $P(s_t^i)$. In the DBN literature, this is known as the *factored frontier* algorithm,[10] or one-pass loopy belief propagation.[7]

3.3.2 M^{th}-Order Cross-Sensor Model

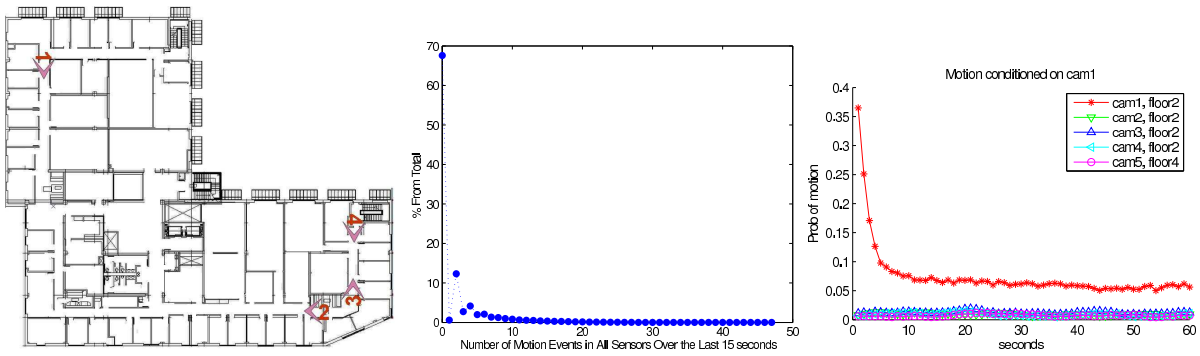
Assume we have a M-order Markov model, where the prediction of state s_t depends on the past M states. Again, a native implementation of $P(s_{t+1}^i|s_{(t-M):t}^{1:N})$ would require a table of size 2^{NM}. If we assume that people move independently, this suggests a noisy-OR model of temporal correlations.

$$P(s_t^i|s_{(t-M):(t-1)}^{1:N}) = 1 - \prod_{o=1}^{M} \prod_{j=1}^{N} (1 - \alpha_{ij}^o s_{t-o}^j) \tag{10}$$

where α_{ij}^o is the probability that a person moves to camera j o-seconds later, given that they are currently at camera i.

The above model states that if there is no motion activity in any camera in the past M-seconds, there can be no motion in any camera currently. To overcome this limitation, we allow for a probability α_i that a person will spontaneously appear at camera i - this is equivalent to a *leaky* noisy OR. We add in $(1 - \alpha_i)$ as an extra term in the product in (10).

Structure learning: We observe that most of the α_{ij}^o dependencies are weak; far-away cameras do not effect each other. To improve speed and enforce sparse connectivity, we zero out the α_{ij}^o values below some threshold. For a given camera i, these prunes away many of the dependencies between $s_{(t-M):(t-1)}^{1:N}$ and s_t^i. This means we are learning the temporal *and* spatial structure of our sensor network, formalized as a DBN.

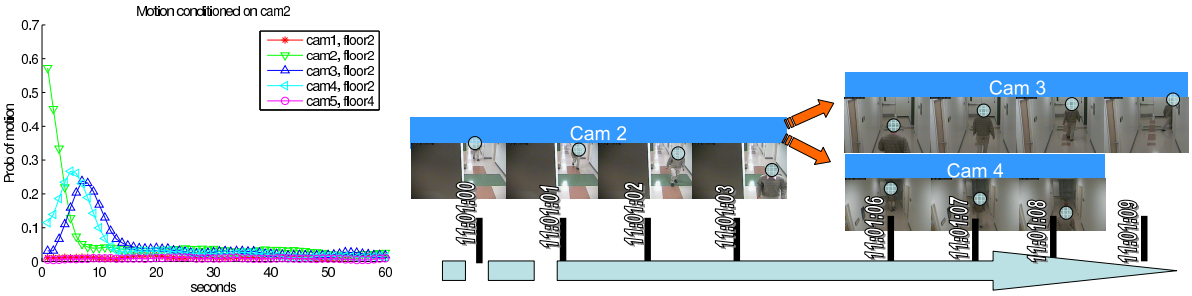


(a) 2nd Floor (b) Motion is Sparse Over Time (c) Only Self Correlated

Figure 1. Physical Instrumentation and Visualization of Motion Semantics.

3.4 Visualization of the Semantics Collected

Fig 1(b) illustrates several semantics of motion: usually, only a small number of motion events occur in a given time frame; Most of the time, there is no motion; When motion occurs, it is very unlikely to occur in only one camera over a 15 second time frame; This teaches us that when motion is detected, there really is a benefit for seeking correlated cameras as motion is rare and when it occurs it doesn't come alone. Camera 2 is deployed in a short hallway that leads in cameras 3 and 4. One can validate our spatio-temporal correlation models by looking at the camera layout (Fig 1(a)). We plot the conditional probability of motion in other cameras given



(a) Cross-Correlated (b) Correlation of Cameras 2,3 and 4

Figure 2. Conditional Correlation of Motion Between Cameras.

that motion was observed at camera 2 in Fig 2(a). Another possible case, is when only self correlation is present as it is the case for camera 1, illustrated in Fig 1(c). This semantic information can be exploited by the scheduler in the following way: Given that motion appears in camera 1, it will estimate motion continuing in camera 1 with high probability. 6 seconds after motion was detected in camera 2, it will estimate motion in the correlated cameras (3, 4), with high probability. A case of such motion pattern is illustrated in the image in Fig 2(b).

4. SUB-TASK SELECTION STRATEGIES

In this section, we propose six strategies for sub-task selection, give their formal definition and algorithm sketch. We consider a system with N sensors $C_1, .., C_N$ and subjects appearing in view of these cameras. A budget of K frames can be processed each second. The statistical attributes were available and were computed off-line, while the real-time attributes are gathered while frames are processed.

First, we presented three ad-hoc, **deterministic** approaches termed as such because the logic they implement is based on ad-hoc intuition of how resources should be allocated.

1. **RR** A based line algorithm, where regardless of the state of the system, the scheduler probes in a Round Robin fashion.

2. **RR + Stay** We observed that while motion tended to be continuous, our next algorithm followed this approach and exploited the continuous nature of the signal. It planned to stay as long as motion was detected and when it was not detected at the previous probe, it moved to the next sensor in a round-robin fashion.
3. **RR + Stay + Go to Nearest Neighbor** We observed that motion is correlated, i.e, motion in one location usually triggered motion in a another location. Thus, when motion was observed, the algorithm would choose to keep probing the sensor. Once the motion ended, we waited for 60 seconds for motion in the nearest neighbor. Note that it would be very hard to come up with a systematic algorithm that will capture the correct *semantics* of the system. The ad-hoc values we choose were selected before we know the actual semantics of the motion in the monitored building. As will be shown in the experimental section, this approach proves to perform even worse than the previous alternative. The probabilistic algorithms that we present next follow a systematic approach based on the actual system semantics and prove to achieve a great improvement in detecting motion events under budget constraints.

The following are **probabilistic** algorithms that exploit the semantics of motion at three different levels: probability of spontaneous motion, motion continuing at the same sensor and motion appearing at a sensor after it appeared in a previous sensor. These probabilities, which we refer to as motion semantics, are computed before the actual execution of the algorithm, based on training data.

1. **Semantic RR** Given the probability distribution of motion at each sensor, we selected a subset of k sensors sampled randomly from the distribution. For example, if cameras i and j were assigned probabilities of 0.1 and 0.2 respectively, camera j is twice as likely to be probed by the algorithm.
2. **Semantic RR+Stay** Our next Algorithm implements the single-sensor Markov model from Sec.3.2. At each time instant, the system maintains a probability of motion at each sensor. This probability can be used, with the dynamic model, to predict the probability of motion in the next time instant. If the scheduler choose to probe this camera, the state is set to 0 or 1. If the probe yields a '1', the predicted probability of motion for the next time instant will be high, and so the scheduler will tend to probe the same sensor until another sensor becomes more likely or, alternatively, no motion is found. Hence the algorithm is naturally interpreted as a probabilistic variant of **RR + Stay**.
3. **Semantic RR+Stay+Correlations**
   Our final algorithm with we refer to as Algorithm 1, implements the multi-sensor Markov model from Sec 3.3. This allows nearby cameras to affect the motion prediction for a particular camera.
   To implement the prediction step, we need to maintain a MN matrix of probabilities for motion of all N sensors for the last M seconds. To compute the predicted probability for camera i, we multiply the matrix with the correlation values a^0_{ij}. Doing this for all N cameras requires MN^2 operations. We observe that most of the values in the MN matrix of state history probabilities are either zero (if probed) or very close to it[§] and thus their contribution is insignificant. By rounding the state probabilities to {0,1} we optimized this step significantly, as most of the multiplication (and the lookup) operations were avoided when the probability is zero.
   Note that the complexity of generating a plan, in this algorithm is $\Theta(N*N*M)$ compared to $\Theta(N*2^{NM})$ required by a naïve implementation that would try to generate the complete state table (see sec 3.3).
   While experimenting with the algorithm we identified two pitfalls: The first is in line 15: we choose to probe based on **sampling**, as opposed to probing based on the order of probabilities (high to low). The ordering alternative will prefer probing cameras where motion is probable and might cause starvation in other sensors and thus will perform sub-optimally.
   The second is in **line 6** where we add to the history table (**H**) only motion that was probed. And thus only motion probed is used for prediction. An alternative would be to use predicted probabilities computed in **line 14** for the process of prediction as well. This alternative would cause the scheduler to continue probing sensors for a very long period, even when motion is no longer present, as the following example illustrates:
   Assume that two sensors: A and B are correlated, $k = 1$ and motion was present and detected by the scheduler at A for 3 seconds, and then ended. Each motion instance in A will cause the probability in B

[§]As most of the time there is no motion

Algorithm 1: Semantic RR+Stay+Correlations

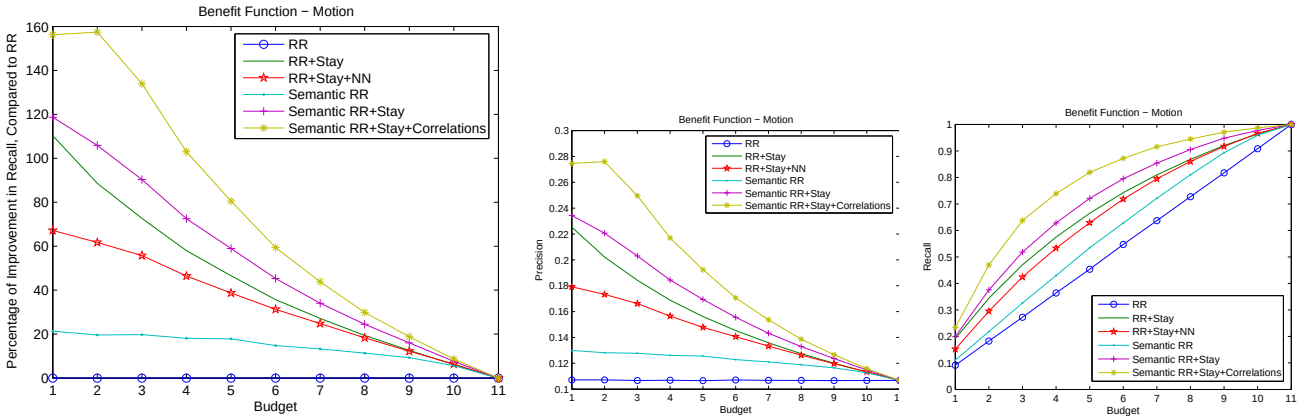
Data: **P** set of cameras probed at the last time unit. **K** Budget to probe. **H** History of motion probabilities for all cameras in the last M seconds. α- For each pair of cameras and $1 \leq t \leq M$ contains the dynamic model as in eq. 10. **current_time** System timestamp (Subtracting two timestamp values gives the difference in seconds.).

Result: **Plan** cameras to probe at time t_i.

begin
  /* *Correction Step*: Update the probability of motion based on what we know */
  for $i \in P$ **do**
    if $p(i).last\_state\_probed = motion\_probed$ **then**
      /* Take into account previous motion seen */
      $H \longleftarrow H \cup \{i.cam, 1, current\_time\}$
  /* *Prediction Step*: Go over the current probabilities and update them (see eq. 10)*/
  /* Sensors is the list of Sensors, MP is initialized with the probability of motion for each sensor. */
  for $j \in Sensors$ **do**
    for $i \in H$ **do**
      /* Compute the cross-correlation probability */
      $MP(j) \leftarrow \prod[1 - j.prob * \alpha_{i.cam,j.cam}^{current\_time - j.time}]$
    /* Add the apriori probability */
    $MP(j) \longleftarrow 1 - (MP(j)) * (1 - \alpha_j)$
  $Plan \longleftarrow Choose\_Based\_on\_Distribution(K, MP)$
end

to increase. At this point, B's probability for motion becomes relatively high (as it is correlated to A). Had we used predicted probabilities for motion prediction, B's probabilities would cause the probability of motion at A to rise (as it is correlated to B). Eventually this will cause the scheduler to keep computing wrong probabilities and be "trapped" between A and B as its prediction is based on false probabilities. We named this phenomena "the gossip motion trap". It can be avoided by not predicting motion based on "gossip" (speculated motion), and only on motion that was actually probed.

5. EXPERIMENTAL SETUP



(a) Improvement Over RR (b) Precision (c) Recall

Figure 3. Comparison of the Different Algorithms - Motion Hits.

We evaluated our algorithms on a network of 11 cameras distributed on two floors in the CS building at UCI. The Data we collected was over a period of two weeks (the first and second weeks of the Spring quarter), all

cameras were located in hallways. We then used the first week to train for optimal collection of the other week. The physical location of the cameras is shown in fig 1. The cameras were Dlink DCS-6620 with built-in motion detection. The cameras were configured to transmit a time-stamped picture to an FTP server, whenever motion was detected. The time interval was set to 1 second between two subsequent images.

Each floor has a Cisco Catalyst 6500 wired switch-router and each camera is plugged in at 100Mbps (wired connection), switched Ethernet each router is connected to each other at 1Gbps. The FTP server collecting the images has runs proftpd over SunOS with 500MB of memory.

Note that we actually collect accurate ground truth (without resource limitations) data during training and test periods; However, the data was collected with serious delays - this does not prove harmful for testing or training because images are tagged with local camera timestamps. Our end goal is to design a **Real-Time** scheduler which avoids such delays using a semantic model.

Furthermore, each cameras was able to process the images locally in hardware, sending only images containing motion. In a general setting where the sensors don't have these **local** processing capabilities, scheduling becomes a vital issue since a central server must also perform motion detection for each time instant for each camera.

We ran the evaluation on a Windows XP PC running Oracle with Intel Pentium 4, 3.0 Mhz processor and 1Gb of RAM.

5.1 Evaluation Methodology

The training data was used to generate three tables: first the probability of unconditioned motion for each camera was computed and stored in a table. The cardinality of that table is 11, one record per camera. Second, for each camera we computed the self-correlation matrix, as discussed in sec 3.2. The cardinality of this table was 11*4, four values per sensor. Finally, we computed the correlation between sensors (as discussed in sec 3.3.2), and stored the correlations in a table of size 11*11*15¶.

To reduce the size of the correlation matrix, we did the following: We computed a threshold probability, $p = 0.05$, and removed all entries of less probability from the table. This value to be visually verified by noting that in Fig. 1(c) and 2(a), everything below 0.05 is noise. Similarly, we found that it is sufficient to look back only 15 seconds, when considering correlations between different cameras.

The correlation matrix was not to be used for self-correlation, which justified the removal of all entries where a camera was self-correlated. Eventually, the cardinality of the table containing these associations was 165 as opposed to 1815 we started with. The different algorithms were implemented in PL/SQL. We simulated the different algorithms, with varying budget: $1 \leq k \leq 11$ by running it over the test data. Each algorithm generated a plan to probe k sensors based on the motion detected by the previous probes. We counted the number of "hits" of each plan and later computed the precision, recall, and relative improvement in recall over round robin. Precision was computed as the fraction of probes containing motion out of the total number of probes. The recall was the fraction of motion detected out of the whole motion in the system.

5.2 Evaluation Results

With a budget of 1, our last semantic algorithm achieved over 150%(!) improvement over RR (see Fig. 3(a)). "RR+Stay+NN" performed *much worse* than "RR+Stay". This proves that without exploiting semantics properly, an ad-hoc approach will prove to perform worse than a simple alternative("RR+Stay"), which in this case better exploits the semantics of the data.

The probabilistic approach "Semantic RR" proved to work better then RR but worse than all other alternatives that took into account the continuation of motion. "Semantic RR+Stay" is the "second-best" candidate exploiting more of the available semantics than all other alternatives and proves to achieve about 100% improvement over the best out of the deterministic approaches.

"Semantic RR+Stay+Correlations" exploiting the largest number of available semantics dominates the chart and offers tremendous improvement over all other alternatives, when the resources are constrained.

¶which is n*n*M where n is the number of sensors and M is the time window to look back

To give an intuition for the reasons behind this vast improvement, imagine that a sensor falsely detects that there is no motion. The deterministic approaches would choose to go to a different sensor, and will not be able to "recover" from that error. The probabilistic approaches elegantly recover from that situation as the training data teaches the algorithm that sometimes it is a good idea to look back and check again for motion, as the probability for motion will not drop to zero immediately after no motion was detected (see Fig 1(c)).

6. RELATED WORK

To the best of our knowledge, we are first to address the problem of scheduling access in a camera network using a semantic model of building activities. Our correlation-based model is simple enough to be implemented in real-time, yet powerful enough to capture meaningful semantics of typical behaviour. However, there is numerous relevant and related work which we now discuss.

The general problem we addressed relates to the problem of **deadline assignment to different jobs**[11] where a global task is to be completed before a deadline and multiple processing components are available for processing. We can think of the task of event capture as a global task which is to be detected until a deadline, the end of the event. Our work mainly as the question we try to answer is which nodes should take part in the processing when different node have different cost/benefit related to the task in hand. Only a specific node (the one where the event is in the FOV) can help the system meet the "deadline" or in our terminology, maximize event capture we exploit semantics of motion to guide the selection of nodes to probe, by definition nodes are not equivalent processing units.

In the context of sensor networks, Yu et al.[3] and Han et al.[4] describe a prediction based approach used to **utilized battery consumption** in sensor networks. Only when the observed value deviates from the expected value based on the prediction model, by a pre-defined error bound, the sensor transmits the observed value to the server. The error bound is defined by the application according to different quality requirements across sensors. Lazaridis et al.[12] proposed an online algorithm for creating approximation of a real valued time series generated by wireless sensors, that guarantees that the compressed representation satisfies a bound on the L_∞ distance. The compressed representation is used to reduce the communication between the sensor and the server in a sensor network.

In the domain of **people and object tracking** Schulz et al.[13] and Bar Shalom et al.[14] address the problem of tracking when the estimates of location are not accurate, i.e., some of the generated estimates are erroneous due to the inaccuracy of the capturing sensor. The task in both papers was to reconstruct the actual trajectory of the object being tracked with statistical models, such as Kalman Filters[15] are utilized to find the most probable trajectory. The kinds of problems that these statistical models are used for try to estimate the location of moving objects, for example tracking planes given a wandering blip that appears every 10 seconds. Work by Song et.al used semantic information to address the problem of inaccuracy feature based tracking algorithms that try to associate subjects seen at different cameras.[16] Here the challenge is again individual tracking and specifically, being able to associate two images from two different cameras to the same person.

There is a large body of work on **schedule optimization** within the sensor modelling literature.[1,2,8,9] A common approach is to build a probabilistic model of object state and probe sensors that reduce the entropy, or uncertainy in state, at the next time frame. Our application is different in that, rather than reducing the uncertainty in building state, we want to collect images of events (even if, and indeed especially if, we are certain of their occurence) for further high-level processing. Directly modeling object state is difficult in our scenario because a large number of people move, interact, and enter/exit, making data association and tracking extremely challenging. We take a different approach and directly model the *environment* itself using a simple, but effective, sensor-correlation model.

Modeling people activity was studied in Ihler et al.[2] It was found that a Poison process provided a robust and accurate model for human behavior, such as people and car counting, and showed that normal behavior can be separated from unusual event behavior. In the context of intrusion detection Scott(2000)[17] proposed a Markov modulated nonhomogeneous Poisson process model for telephone fraud detection. Scott used only event time data, but detectors based on Scott's model may be combined with other intrusion detection algorithms to

accumulate evidence of intrusion across continuous time. Our work differs as we tried to find event of interest on real time under constraints.

One common task in sensor networks is **topology reconstruction** -[18] explicitly do this in the context of non-overlapping cameras. Our correlation model implicitly encodes such a spatial topology, but also captures temporal dependencies between sensors (Fig.1(c)). We know of no previous work on spatio-temporal structure estimation, though we found it vital in our building environment model.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we address the challenge of scheduling data collection from sensors to maximize event detection. We designed and implemented a fully functional system using available off-the-shelf cameras to learn motion semantics for our building and evaluated different scheduling algorithms based on non scripted motion over a period of a week. Our semantic based algorithm that takes into account all available semantic information proves that even under significant resource constraints, we can detect a very large number of events.

We followed a pull based approach, however, for future work plan to investigate a hybrid push/pull approach that will benefit from the advantages of computation at the sensors while meeting the deadlines of a real-time system and supporting the dynamic nature of a surveillance task.

We started this work stating five assumptions. In our future work, we plan to relax these assumptions: **Semantics available:** we will extract features from the images, such as number of people in the FOV learn semantic information based on such information and use semantics at different levels (motion, number of people) for scheduling. **Updating the Semantic Model:** while the system is in operation, up-to-date information is collected about the state of the system. This information can be used to tune the semantic model, for example, due to changes in the semantic behavior (A new coffee-machine was installed and more people tended to stop for coffee). This problem relates to a problem addressed by the database communitee: estimating query selectivity based on the actual query results.[19] We plan to study the affect of different **Prediction Windows** on the recall rates of our algorithms. The scheduler will be given the state of the system as was "seen" in the last t seconds, and will generate a plan for the next t seconds. At that stage, **Distributed Decision Making** might be taken into account as each node might choose to "stay" when motion is "seen" and further "tip" a correlated node to expect as opposed to being loyal to an obsolete plan. Note that our current algorithms can scale with large number of sensor as semantics are local and several decision making nodes can operate in parallel, e.g., we could have installed a different scheduler for each floor in out building case. Different cost functions that depend on the actual network topology, delays and connection overheads is in our plan to relax the **Homogenous** assumption.

8. ACKNOWLEDGMENTS

Support of this research by the National Science Foundation under Award Numbers 0331707 and 0331690 is gratefully acknowledged.

REFERENCES

1. J. Williams, J. Fisher III, and A. Willsky, "An approximate dynamic programming approach to a communication constrained sensor management problem," *Proc. Eighth International Conference of Information Fusion* , 2005.
2. V. Isler and R. Bajcsy, "The sensor selection problem for bounded uncertainty sensing models," *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on* , pp. 151–158, 2005.
3. X. Yu, K. Niyogi, S. Mehrotra, and N. Venkatasubramanian, "Adaptive target tracking in sensor networks," *The 2004 Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'04), San Diego* .
4. Q. Han, S. Mehrotra, and N. Venkatasubramanian, "Energy efficient data collection in distributed sensor environments," *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on* **39**, pp. 590–597, 2004.

5. A. Ihler, J. Hutchins, and P. Smyth, "Adaptive event detection with time-varying poisson processes," in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 207–216, ACM Press, (New York, NY, USA), 2006.
6. L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE* **77**(2), pp. 257–286, 1989.
7. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
8. V. Krishnamurthy, "Algorithms for optimal scheduling and management of hidden Markovmodel sensors," *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]* **50**(6), pp. 1382–1397, 2002.
9. V. Gupta, T. Chung, B. Hassibi, and R. Murray, "On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage," *Automatica* **42**(2), pp. 251–260, 2006.
10. K. P. Murphy and Y. Weiss, "The factored frontier algorithm for approximate inference in dbns," in *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pp. 378–385, Morgan Kaufmann Publishers Inc., (San Francisco, CA, USA), 2001.
11. B. Kao and H. Garcia-Molina, "Deadline assignment in a distributed soft real-time system," *IEEE Transactions on Parallel and Distributed Systems* **8**(12), pp. 1268–1274, 1997.
12. I. Lazaridis and S. Mehrotra, "Capturing sensor-generated time series with quality guarantees," 2003.
13. D. Schulz, D. Fox, and J. Hightower, "People tracking with anonymous and id-sensors using rao-blackwellised particle filters," *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)* , 2003.
14. E. Daeipour and Y. Bar-Shalom, "An interacting multiple model approach for target tracking withglint noise," *Aerospace and Electronic Systems, IEEE Transactions on* **31**(2), pp. 706–715, 1995.
15. R. Brown and P. Hwang, "Introduction to random signals and applied Kalman filtering," 1997.
16. B. Song and A. Roy-Chowdhury, "Stochastic Adaptive Tracking In A Camera Network," *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on* , pp. 1–8, 2007.
17. S. Scott, "Detecting network intrusion using the markov modulated nonhomogeneous poisson process,"
18. D. Makris, T. Ellis, and J. Black, "Bridging the gaps between cameras," *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* **2**.
19. C. Chen and N. Roussopoulos, "Adaptive selectivity estimation using query feedback," *Proceedings of the 1994 ACM SIGMOD international conference on Management of data* , pp. 161–172, 1994.