Exploiting Semantics for Sensor Re-Calibration in Event Detection Systems[long]

Ronen Vaisenberg, Shengyue Ji, Bijit Hore, Sharad Mehrotra and Nalini Venkatasubramanian

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

ABSTRACT

Event detection from a video stream is becoming an important and challenging task in surveillance and sentient systems. While computer vision has been extensively studied to solve different kinds of detection problems over time, it is still a hard problem and even in a controlled environment only simple events can be detected with a high degree of accuracy. Instead of struggling to improve event detection using image processing only, we bring in semantics to direct traditional image processing. Semantics are the underlying facts that hide beneath video frames, which can not be "seen" directly by image processing. In this work we demonstrate that time sequence semantics can be exploited to guide unsupervised re-calibration of the event detection system. We present an instantiation of our ideas by using an appliance as an example - Coffee Pot level detection based on video data - to show that semantics can guide the re-calibration of the detection model.

This work exploits time sequence semantics to detect when re-calibration is required to automatically re-learn a new detection model for the newly-evolved system state and to resume monitoring with a higher rate of accuracy.

Keywords: Semantics, Event Detection, Accuracy of Detection, Self-Calibration

1. INTRODUCTION

With increased awareness among public, private and government sectors for need of greater security, surveillance technologies (especially video surveillance) have recently received a lot of attention. Video surveillance systems are being used in a variety of public spaces such as metro stations, airports, shipping docks, etc. The challenge of achieving high Accuracy of Detection (AoD) from large numbers of cameras increases as more cameras go up in more places.

Aside from the numbers of sensors currently being used, some applications require the deployment of sensors in distant and isolated locations. According to Liljegren el. al,[1] the U. S. Department of Energy (DOE) Atmospheric Radiation Measurement (ARM) Program has deployed dual-channel microwave radio-meters in rural Oklahoma and Kansas, the north slope of Alaska, and on islands in the tropical Pacific Ocean. These radio-meters provide continuous measurements of integrated water vapor (IWV) and integrated liquid water (ILW) amounts. Due to the remote nature of these locations, several weeks or months may elapse between maintenance visits by operations personnel. Even then, subtle problems that can adversely affect the instrument calibrations may go undetected. Achieving high accuracy of data generated from sensors of this nature is a real challenge.

In this paper, we propose a general approach to ensure AoD for systems modeled as Finite State Machines (FSM). The challenge lies in designing a robust detection system for specific use when manual intervention is impractical due to the distributed nature of the sensors or their remote locations. We have designed a novel approach to ensure AoD by using semantics of the tracked system for the purpose of automatic re-calibration of the detection algorithms used by the sensors.

For the sensor to ensure AoD, it must accurately interpret the captured feature space to application specific information. Usually, this process involves setting several parameters. For example, OpenCV needs a CASCADE*

*OpenCV[2] uses a statistical approach for object detection, an approach originally developed by Viollan and Johns[3] where features extracted from training sets of data are compressed into a statistical model stored as an XML document.

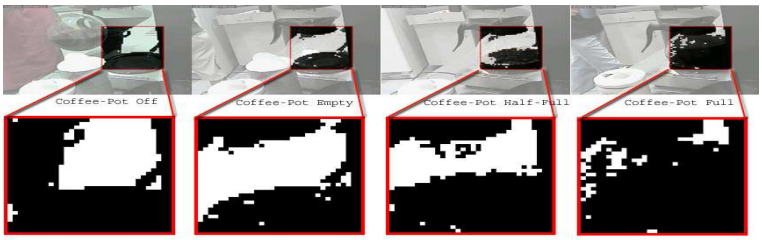

Figure 1. The 4 different states in our example. The black/white cells represent dark/bright average pixel color.

as a parameter to perform face detection and a video camera needs a background model as a parameter. The detection algorithm in the ARM microwave radio-meter has to adjust to changes in the physical location of its mirrors that tend to slip as much as 1 degree on their stepper motor shafts due to continuous use.

Because they are deployed in uncontrolled environments the sensors must overcome problems in order to achieve better AoD. For example, in the case of computer vision, cameras might be replaced, lighting conditions may change and camera views might be altered. Physical changes could also occur due to continuous usage, e.g., the ARM microwave radio-meter example. In order to overcome these difficulties the sensor follows a pre-defined parameterized prediction model (we will clarify this in sec. 3). This model is used to perform the detection based on the sensed feature space but it is inherently imperfect. Specifically, changes due to the nature of the uncontrolled environment in which the sensor operates or due to imperfect matches between the learned model and the actual observed system will result in reduced AoD for that sensor. The parameters are set to reduce the uncertainty of the uncontrolled environment.

Although these parameters enable better state detection by the sensor when set correctly, when they don't match the current situation in the sensed environment they become a problem. Re-calibrating these parameters is necessary to ensure that the detection process meets AoD requirements. We propose a novel approach to re-set the detection algorithm to the new environment by searching the space of parameters to maximize detection consistency with a semantic model.

To the best of our knowledge, the proposed approach is first of its kind that allows parametrization in real time of a given imperfect model based on known system semantics to achieve better AoD. This approach was implemented and evaluated in a real setting and was found to be able to parameterize a model within a very large parameter space.

The paper is structured as follows: In Sec. 2 we give a mathematical definition for the observed system and collected semantics. sec. 3 describes the architecture and outlines the role of semantics in the overall framework. Sec. 4 presents the algorithm for exploiting semantics for sensor re-calibration and improved AoD. Sec. 5 shows the instantiation of our semantic based detection model applied to an appliance: coffee pot level detection based on video data. Sec. 6 presents the results of applying the algorithm on our example application. Sec. 7 discusses related work. Future work and conclusions are presented in sec. 8.

2. SYSTEM MODELING

Many systems of interest and observed environments can be modeled as Finite State Machines. For example, the level of pressure ("Critical","High", "Medium", "Low") on the foundations of a bridge are of crucial importance to maintenance authorities. Real time GPS navigation systems might consult the traffic sensors for average speed of vehicles ("30mph","40mph","50mph","60mph") to determine alternative routes to various destinations.

Four states of interest were specified for the coffee machine in our office kitchen: "Empty", "Half-Full", "Full", "Coffee-Pot Off". Fig. 1 shows the four different states of our system. Our initial goal was to detect the state of the coffee machine at all times and notify the subscribers when there was fresh coffee in the pot. Our final goal is to design a robust system by making it resilient to physical perturbations, for example, slight changes in the field of view of the camera, physical displacement of the coffee machine or replacement of camera used for monitoring using semantics. The reminder of this section defines a system as a finite state machine (FSM) and the semantic model as the transition times of that FSM.

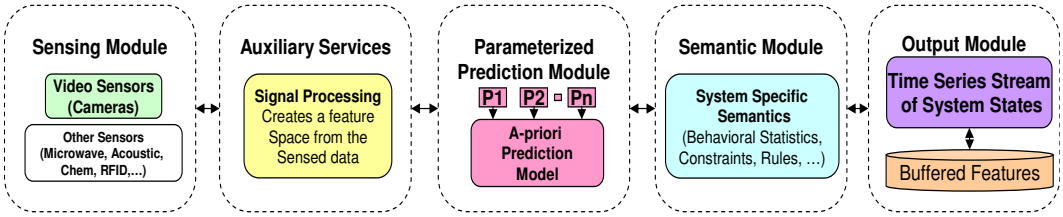

Figure 2. Main Modules Involved in the Detection Process.

2.1 System Model

The system is modeled as a finite state machine (FSM) of n possible states. Observing a system as a finite state machine implies several assumptions that must be taken into consideration. First, many systems are of a continuous nature and a discreteized representation will lose information. Also, setting strict limits between states, might cause the detection to fluctuate between two states when the actual system state stabilizes around the boundary point of the two states. This will result in an incorrect representation of the system's behavior. Second, choosing the incorrect number of states will reduce the reliability of the detection process. For example, discarding Coffee-Pot off, will potentially result in reduced accuracy of detection since there is a possibility that when the system reaches this state it will be classified incorrectly as one of the other three. Third, in uncontrolled environments, even if the number of states selected expresses the complete set of possible system states, outlier states are possible. Outlier states[†] will cause more "noise" to be added to the detection process.

2.2 The Semantic Model

A semantic model for the system is derived by observing its state transition characteristics over time. For example, the average transition time for: Half-Full→Full is 8 minutes and 6 seconds according to the statistics we collected.

2.2.1 Different Time Classes of the Model

Time is broken down into classes. The classes represent varying behavior patterns of the system over several time intervals. For example, the number of people in a conference room, is highly dependant upon the day of week (work day or weekend) and the time of day. (working hours vs non-working hours).

2.2.2 Average and Standard Deviation of Transition Times

For each class C and for each state Si, we calculate the average and standard deviation time it takes to make the transition $Si \rightarrow S_{i+1}$ where $Si \neq S_{i+1}$ and the transition time is in C. The average transition times are used to approximate the probability of a given transition detected to be consistent with the system regular transition times. We expect most transitions to fall in the $\bar{x} - 2\sigma$ and $\bar{x} + 2\sigma$ range. For example, the deviation from the average transition time for the transition: Half-Full→Full is relatively small since it depends on the time it takes the machine to make the coffee which is relatively predictable. We will refer to this statistical model as the *semantic model* of the system from this point on.

3. SYSTEM ARCHITECTURE

Fig 2 depicts a high-level outline of the main modules that are involved in the detection process.

- **Sensing Module:** Processes data from incoming sensors. The sensing module is in-charge of the translation of the physical signal sensed by the sensor to the digital signal processed by the processor.

[†]We refer to outlier states as states that are not part of the normal operation of the system. These states are reached as a result of an unexpected event in the detection processes or system behavior. For example, temporary obscurity in video camera will most probably result in an outlier set of features being captured from the camera.

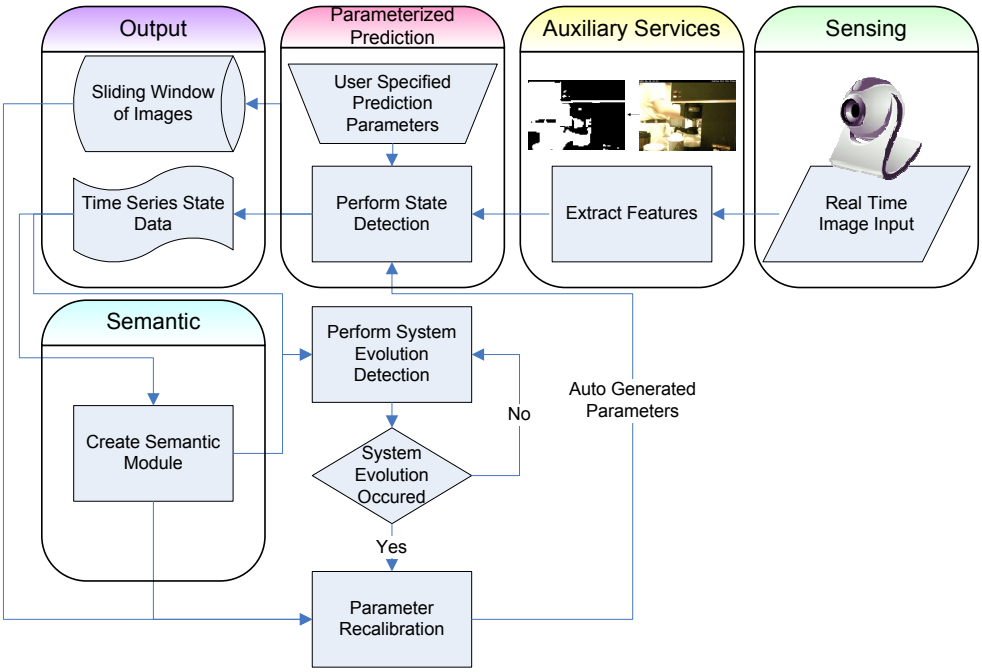

Figure 3. A flowchart of the execution of the algorithm. The modules involved are the same modules described in sec. 3 as well as the processes that execute the algorithm.

- **Auxiliary Services:** A service library containing modules that provide auxiliary services necessary to create the feature space from the digital signal made by the sensing module. The auxiliary services transform the data created by the sensor to form a higher level representation of the system that can be processed by the prediction module. For example, a service that translates the Jpeg image to the corresponding RGB color histogram is considered an auxiliary service.
- **Parameterized Prediction Module:** An a-priori model used for output generation by the sensor. For the sensor to satisfy AoD requirements the prediction module is consulted for prediction based on the features created by the auxiliary services. Since the prediction model performs differently under varying conditions, the model uses the notion of parametrization of the prediction. The added parameters, enable the prediction model to exploit some of the special characteristics of the specific conditions under which it is required to operate.
- **Semantic Module:**

  A model that describes the normal behavior of the system under surveillance, i.e., system semantics. This module is used to monitor the detection process and decide when it deviates from a normal operation expected from the system.

  This module is used for two main tasks: First, it detects the changes in the conditions that causes the system to stop detecting states correctly. Second, it finds new parameters for the new conditions. Once the new parameters are set the detection becomes once again consistent with the semantic model.
- **Output Module:** The output module generates a time series where each entry represents the time instant at which a state transition takes place. This module is also responsible for populating a buffer of sensor readings which is used for re-calibration when the conditions in the sensed environment change.

The flow of execution is presented in Fig. 3. The following are the five main steps of execution:

1. First, a monitored phase of detection is deployed. The user, specifies the prediction parameters and verifies

the output of the detection algorithm. Output: a time series data of system states that was verified for accuracy.

2. Generate the semantic model. Output: a reference model used for system evolution detection and re-calibration.

3. Based on a parameterized prediction model the system executes and detects state as long as the series of states detected complies with the semantic model. Output: a time series data of system states verified against the semantic model.

4. When the states that were detected are no longer consistent with the semantic model, it is assumed that the system had evolved. This means, that the detection can no longer be performed based on the current parameterized prediction model. Output: a notice that the system had evolved and triggering the re-calibration phase.

5. The system tries to build a new reference model based on the semantic model (re-calibration phase). Output: success iff a new model was created that was verified for accuracy against the semantic model.

Step 5, is the key step in the algorithm where the previous model was detected as invalid and a new model is constructed. A successful outcome of this step enables the re-calibration of the detection model based on the semantics of the system. The next section, discusses this step in details.

4. SEMANTIC BASED SYSTEM EVOLUTION DETECTION AND RE-CALIBRATION

DEFINITION 1. *Systemevolution refers to the change in the configuration of the system.*

DEFINITION 2. *$time_{evolution}$ is the time instant at which the system changes completely from the old state to a new, evolved state.*

For example, changes in the physical location of the ARM microwave radio meter's mirrors will fall under our definition of system evolution. Changes in the view of the camera is also an instance of system evolution. We assume that system evolution occurs rarely but instantaneously. Studying the re-calibration of systems that do not follow these two assumptions is out of the scope of this paper[‡].

The following algorithm, scores a sequence of perviously detected time series stream of system states: ψ. The fraction of transition times that take more then two standard deviation units from the average time is returned as the score for ψ.

procedure *evaluation*(ψ) **returns** score
$FaultCount \leftarrow 0$
for $s[i] \in \psi = \langle s[1], s[2], .. \rangle$ **do**
  $trns \leftarrow t[i] - t[i-1]$
  $class \leftarrow class[i]$
  $\bar{x} \leftarrow GET\_AVG\_FROM\_MODEL(s[i], s[i-1], class)$
  $\sigma \leftarrow GET\_STDDEV\_FROM\_MODEL(s[i], s[i-1], class)$
  $range[A, B] \leftarrow [\bar{x} - 2\sigma, \bar{x} + 2\sigma]$
  if $trns \notin range[A, B]$ **then**
    $FaultCount \leftarrow FaultCount + 1$
  end if
end for
return $\frac{FaultCount}{|\psi|}$
end procedure

DEFINITION 3. *$Threshold_{Consistent}$ is a constant, defined by the user. This threshold defines the boundary between fault level that is consistent with the model, to the inconsistent fault level.*

[‡]Systems that change patters of behavior for prolonged periods may cause the algorithm to start a re-calibration phase. That can be accounted using a larger stream ψ or by checking for outlier behavior before re-calibration as described by Scott(2000).[4]

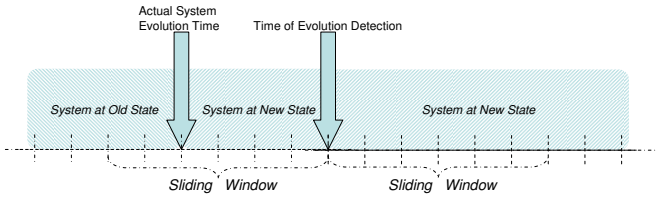

Figure 4. The time gap between the actual system evolution time and the evolution detection time. Before the re-calibration phase begins the algorithm ensures that the sliding window contains only observations that relate to the new system state.

4.1 System Evolution Detection

System evolution is declared iff $evaluation(\psi) > Threshold_{Consistent}$. The length of the stream $|\psi|$ will be referred to as $Window_{Buffer}$.

DEFINITION 4. *$Window_{Buffer}$ is a first-in-first-out queue (sliding window) of features and series of system states (ψ).*

DEFINITION 5. *$Buffer_{Transitions}$ is the number of actual state transitions of the system, captured by the features collected in the window buffer.*

Choosing the correct value for $Threshold_{Consistent}$ and the number of state transition in the window buffer ($Buffer_{Transitions}$) is discussed in sec. 4.3.

4.2 Model Re-Calibration (Search for a New Set of Prediction Parameters)

The re-calibration process is essentially a search for a new set of parameters that optimizes the consistency of the stream generated by the parameterized prediction model with the system semantic model:

procedure *Recalibrate_prediction_model*$(Prediction\_model, Window_{Buffer})$
returns *Parameters* $p1, p2, .., pn$ *for the evolved state.*
$min\_score \leftarrow 1.0$
$\setminus\setminus$ $P1, P2, .., Pn$ *are the domains for* $p1, p2, .., pn$
for $p1, p2, .., pn \in P1, P2, .., Pn$ **do**
  $score \leftarrow evaluation($generateStream[§]$(Prediction_model(p1, p2, .., pn), Window_{Buffer}))$
  if $score < min\_score$ **then**
    $min\_score \leftarrow score$
    $best\_parameters \leftarrow p1, p2, .., pn$
  end if
end for
return *best_parameters*
end procedure

The above algorithm assumes that $Window_{Buffer}$ only contains features that are of the evolved state, otherwise any single set of parameters will fit only a part of the $Window_{Buffer}$. However, although the evolution occurs in a time instance[¶] this does not mean that the buffer includes only features of the evolved state. Fig. 4 illustrates the problem that arises when the exact evolution time cannot be determined. After a system evolution is declared at $time_{detected}$, we know evolution happened before $time_{detected}$. The problem is we don't know exactly when. Since we want to search for new parameters only for the evolved state, we propose waiting until $Window_{Buffer}$ contains only features that were recorded after $time_{detected}$ before engaging in the re-calibration phase. The features received after $time_{evolved}$ and before $time_{detected}$ will be assigned states based on the previous model, and thus the detection in that case will be inaccurate. Achieving correct state detection for this time period is outside the scope of this paper. The next section shows how we applied the ideas presented in this section to our Coffee-Level detection example to achieve re-calibration when the view or zoom changed or the camera was replaced.

[§]generateStream generates the stream ψ using Prediction_model(p1,p2,..,pn)
[¶]see sec. 4

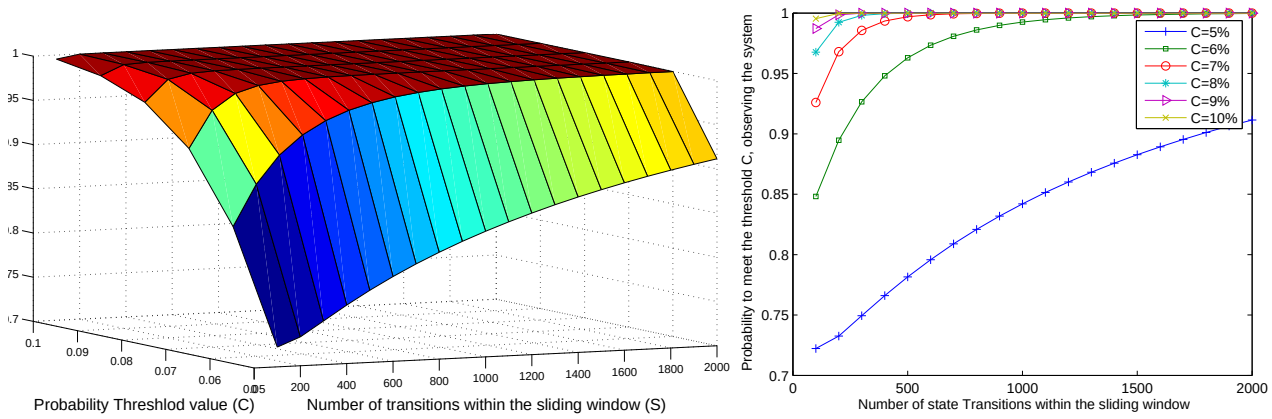

Figure 5. Probability of the observations collected in the $Window_{Buffer}$ to have less than C=$Threshold_{Consistent}$ percent of deviations ($p = 4.4\%$).

4.3 Choosing the Consistency Threshold and Buffer Size

Let p be the probability of a given transition to be *in*consistent with the statistical model (thus, falling outside the range of two standard deviation units from the average). Let us further define $InConsistent(\psi)$ as the number of transitions that are inconsistent with the system semantic model that belong to ψ.

The probability that the algorithm does not incorrectly go into the re-calibration phase when the correct parameters are being used is the probability that $evaluation(\psi)$ returns a score smaller than $Threshold_{Consistent}$. The following formula is the probability of observing up to $Threshold_{Consistent} * Buffer_{Transitions}$ inconsistencies[‖] in independent trials, where the probability of inconsistency in any given trial is p:

$$\sum_{j=0}^{floor(Threshold_{Consistent} * Buffer_{Transitions})} \binom{Buffer_{Transitions}}{j} p^j (1-p)^{Buffer_{Transitions}-j}$$

Fig. 5 plots this probability for the two parameters: $Buffer_{Transitions}$ and $Threshold_{Consistent}$. p was set to 4.4% which is the percentage of data that will be outside this range for a normal distribution. For example, in the case of our coffee-machine we found that $p = 4.59\%$ for our test data. Choosing a high $Threshold_{Consistent}$ value will also increase the probability for false-positives when performing re-calibration (see sec. 4.2).

For a given $Buffer_{Transitions}$, say B_t, the length of time for which the system needs to be observed (i.e., to collect B_t distinct state transitions with a high probability) is decided based on the following two criteria:

1. A minimum of k instances of each state to retrain the prediction model.
2. $Buffer_{transitions}$ number of states needed to reliably re-calibrate the model, see fig. 5.

Therefore, the observation time interval can be computed as follows:

DEFINITION 6. *$Buffer_{TimeLength}$ = MAX(Time period for k occurrences of the most infrequent state, Time period for $Buffer_{transitions}$ occurrences of the most frequent state).*

5. APPLYING THE SEMANTIC BASED STATE DETECTION MODEL TO OUR COFFEE-LEVEL DETECTION EXAMPLE

We applied the model proposed to utilize coffee level detection based on simple set of color layout features that were extracted from a camera that had the coffee pot in view. In this section, we will show how our model was instantiated to a real application. This application should, however, be regarded as a proof of concept only. 6 presents the evaluation results of our appliance system after following the instantiation described in this section.

[‖] $\frac{InConsistent(\psi)}{Buffer_{Transitions}} < Threshold_{Consistent} \rightarrow InConsistent(\psi) < Threshold_{Consistent} * Buffer_{Transitions}$

5.1 Sensing Module

For the experiments we collected a set of images from 10am on June 2007 until 10am June 13th 2007 from two cameras that were focused on the coffee pot ($Buffer_{TimeLength}$=3 days). The images were intended to simulate the buffer of images that would have been created by the proposed algorithm in real time.

If images would have been saved every time unit ($1sec$ in our case) the storage needed would have exceeded 14G**. We therefore followed a more compact approach and saved an image every time a change in the color layout was detected††. This resulted in a set of 1174 images = $Window_{Buffer}$ saved and less then 60Mb of storage used. Later the found that the number of transitions within this buffer was $Buffer_{Transitions}$=203.

5.2 Feature Extraction Using Auxiliary Services

In sec. 2.1 we defined the set of possible states of our appliance example as Coffee-Pot off, Empty, Half Full and Full. The auxiliary services used for detection of these states was Java Advanced Imaging (JAI[5]) library for color layout feature extraction. The image was transformed into a matrix $M(x, y)$ where each cell contains the average color of pixels included in the bounding rectangle ($upperleftX = x * C, upperleftY = y * C, width = C, height = C$) (C defines the number of grid cells extracted from the image). The grid of cells using black and white as the average colors is illustrated in Fig. 1.

5.3 Building the Parameterized Prediction Model

Once the features were extracted, we were able to perform prediction for the actual state represented by each image.

The prediction module we chose was based on color layout similarity. We clustered the images into four different groups based on their color layout. Specifically, we defined a distance function that computed the euclidian distance between the two matrix representations of two images. Based on this distance the images were clustered into $n = 4$ groups (one for each state). A centroid for each group was calculated and manually labeled according to the state it represented. Each image in the $Window_{Buffer}$ was assigned a state based on the label of the nearest centroid found to that image.

Two parameters were required for our prediction model. First, since the data was noisy, in that there were many changes that occur in the background scene, the prediction would have performed better had a background subtraction module being used. A bounding rectangle representation for selecting the system from the background was the first parameter. The second was the actual labeling of the groups of images that were clustered together.

Thus, assuming that the parameters selected expressed most of the possible changes in the uncontrolled environment, when the system evolved, it is a matter of finding the correct new set of parameters to resume detection to meets the AoD requirements.

5.4 Building the Semantic Model

We performed state detection on the set on images collected based on a prediction module where the two parameters were manually seleted. The time series system state stream that was generated was loaded on to a PosGres Database using the following schema:

$Time$	$Class$	$CurrectState$	$PreviousState$	$TransitionTime$
τ	WeekendWorkTime	E	F	4
$\tau + \Delta$	WeekendWorkTime	F	E	4
$\tau + 2\Delta$	WeekendWorkTime	E	F	39
$\tau + 3\Delta$	WeekendWorkTime	H	E	177
.	.	.	.	.
.	.	.	.	.

** $14G = 3_{days} * 24_{hours} * 60_{minutes} * 60_{seconds} * 60k_{jpegSize}$.

††The change detection process itself is very important to the performance of the detection algorithm. All system states need to be captured while only changes that reflect "outliers" should be discarded (see sec. 2.1 for details). We saved images based on the change percentage CP in the color layout features extracted. The CP value was chosen s.t. all possible state transitions will result in a change in the color layout features that is larger then CP

5.5 Performing Re-Calibration

The algorithms below, perform the search for the bounding rectangle and correct labeling. Further details about the two parameters are presented in sections 5.5.1 and 5.5.2 respectively.

procedure *evaluate_rectangle*(*rectangle*) **returns** *score*
clusters ← *clustering*(*rectangle*)
min_score ← 1.0
min_labeled_clusters ← *null*
for *permutation* ∈ *all k! labeling possibilities* **do**
  labled_clusters ← *labeling*(*clusters*, *permutation*)
  score ← *evaluation*(*time_series*(*labeled_clusters*))
  if *score* < *min_score* **then**
    min_score ← *score*
    min_labeled_clusters ← *labeled_clusters*
  end if
end for
return *min_score*
end procedure
rectangle ← *old_rectangle*
while *search continues* **do**
  min_score ← *evaluate_rectangle*(*rectangle*)
  if *min_score* < $Threshold_{Consistent}$ **then**
    return *rectangle*
  end if
  min_score ← 1.0
  min_rectangle ← *null*
  for *rectangle* ∈ *all possible rectangle movement* **do**
    score ← *evaluate_rectangle*(*rectangle*)
    if *score* < *min_score* **then**
      min_score ← *score*
      min_rectangle ← *rectangle*
    end if
  end for
  rectangle ← *min_rectangle*
end while

5.5.1 Parameter1: Bounding Rectangle

Features associated with current bounding rectangle will be extracted to form a set of feature vectors, one for each images from the learning window buffer. A grid divides the rectangle into m by n cells with the same size for each image. Average color is calculated for all pixels inside each cell for every image.

We used classic k-Means[6] algorithm with euclidian distance function for clustering the feature vectors. One thing we did that was different from a standard set-up is that we did not choose the initial k seeds of clusters randomly, rather we used a greedy algorithm which picks initial seeds in a way that none of them is too close to the other. The algorithm is proceeds as follows: choose the pair of points with maximum distance from all pairs of points as the first two seeds s_0 and s_1; $s_i (i > 1)$ is picked from all points such that the minimum distance from $s_j, (0 \leq j < i)$ is maximized. We found that in our setting this greedy algorithm performed better. The reason is that choosing seeds which are distant from each other will increase the probability of each seed representing a different state. Having many observations of one state (in our case, coffee pot empty) caused the centroids generated from a standard set-up to drift towards the random seeds, that were too of the frequently occurring state.

5.5.2 Labeling

Let there be k unlabeled clusters $c_0, c_1, ...c_{k-1}$ and k different labels $l_0, l_1, ...lk-1$ which represent k different states. The labeling procedure is to find the most likely mapping from c_i to l_i. Time series data generated for all

Figure 6. The initial and the resulting bounding rectangle for the two cameras.

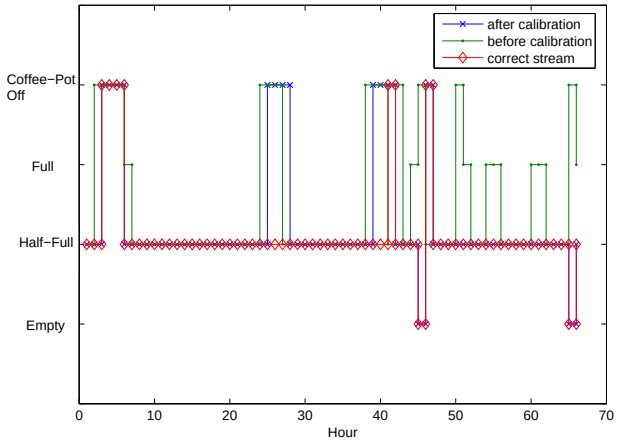

Figure 7. The state at with the system was at for the longest time, for each hour according to the different streams.

different $k!$ labeling possibilities, each labeling instance is evaluated separately. The labeling that generates the lowest score will be considered the most likely labeling. The next section presents the results we achieved when we performed a greedy search in the space of parameters after the system had evolved. The results presented in the next section show, that re-calibration based on a trained model can actually improve AoD.

6. EVALUATION AND RESULTS

We simulated three different situations that might occur: change in the location of the coffee-pot, change in the size of the coffee-pot (zoom in) and change in the camera being used. Fig. 6 shows how, in both our test cases, the starting rectangle was realigned to a position that enabled better detection. The second parameter, labels of the different clusters, was also selected correctly for all groups.

In our experiment we assumed that the new bounding box will be in proximity to the old one. Following this assumption we began our search for a new bounding box from the previous one and expended it iteratively in one of four possible directions: South, North, East and West, according to the greedy algorithm described in sec. 4. Our search thus counted only for the situations where the new bounding box contained the old.

We found that the percentage of transition times that fall outside the range $[avg - 2 * stddev, avg + 2 * stddev]$ for our test data was 4.59%. This value was calculated based on the stream generated in a controlled environment where both the bounding box and the labeling were manually specified. The $Threshold_{Consistent}$ value we used was 0.089 and, along with the 203 state transitions within our buffer, we understood that the probability of the algorithm that meet the threshold was:

$$\sum_{j=0}^{floor(0.089*203)} \binom{203}{j} 0.0459^j (1 - 0.0459)^{203-j} \cong 0.9972$$

Not only were the parameters correctly selected, but also the detection itself also improved. Fig. 7 compares the streams generated before and after the re-calibration took place.

7. RELATED WORK

In the past, time series analysis has been used frequently to study models for system and human behavior. Although we were not able to find work that utilized semantics for re-calibration of prediction models, much work has been done on closely related applications.

In the context of tracking patterns of human activity the proposed models were mainly utilized for abnormal event detection, assuming that the underlying sensing infrastructures work correctly. Our results can be used as an underlying model for better detection on which the abnormal event detection can be detected more accurately. Sweeney et al[7] created a "historical database" for each camera in a set of publicity placed cameras that approximated the number of people present in the camera's view at regular time intervals. The application of the database proposed by the authors was to detect abnormal number of people present (or absent) in a scene. Stauffer et al[8] proposed a probabilistic method for reliable background subtraction. This was used to build a visual monitoring system that observes moving objects on a site and "learns" patters of activity from these observations. These patterns are later clustered and used for the detection of an abnormal pattern of movement of the monitored scene.

Modeling people activity was studied by Ihler et al.[9] It was found that the MMPP provide 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)[4] 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 can be combined with other intrusion detection algorithms to gather evidence of intrusion across continuous time.

In the context of sensor networks, Yu et al[10] and Han et al[11] described a prediction based approach utilizing 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, did the sensor transmit the observed value to the server. The error bound was 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 was used to reduce the communication between the sensor and the server in a sensor network.

In the context of networking, Fu et al[13] exploited time series analysis to model network traffic. They proposed a two-phase information collection process: The first phase was to set a range so that the deviation between the predicted and observed values remained in the range of a given level confidence. Based on the size of the range and the level of confidence, a bound on the sampling rate is determine. In the second phase, the information collection process dynamically adjusted the range as well as the sampling rate based on the briskness of the incoming traffic. The ARMA model was used by Sang et al[14] for network traffic prediction to show how multiplexing can be used to improve traffic prediction. Grossglauser et al[15] studies the impact of the sliding window size of sampling measurements required to minimize the probability that actual arrivals exceed the estimated arrivals in systems following MBAC (measurement-based admission control).

8. FUTURE WORK

Due to the abstract nature of the ideas presented in this paper, many applications can benefit from their use, despite the fact that the application of these ideas was executed in only one domain.

The work should be extended in the following five ways: we showed the applicability of our ideas for our Coffee appliance level detection example only. However, evaluating these ideas on other domains could be studied. Second, we followed a strait forward statistical approach towards the semantics generation. Other models (e.g., Hidden Markov Models (HMM) and rule based model, ..) should be studied in different domains. Third, we only used semantics to validate a given instantiation of parameters, basically performing a greedy search in the parameter space. Optimizing the search using a semantics based heuristic or semantic based search is required. Fourth, the case where the detection is performed using multiple sensors needs to be addressed. Fifth, the semantics collected could be used for prediction as well. We only used the semantics for re-calibration purposes.

It is our belief that building a general-case model for a general system using a general purpose semantic model following a general search strategy is, at the very least, a difficult task. The actual instantiation of these three modules will continue to vary from application to application.

We have presented a new approach for utilizing semantics to achieve better AoD and have proposed the parameterized model concept to improve performance of general case prediction algorithms.

We believe that this approach opens new windows of opportunity in the age of pervasive computing and will prove to be an important building block in a world of unconstrained environments and imperfect prediction models.

ACKNOWLEDGMENTS

Support of this research by the National Science Foundation under Award numbers 0331707 and 0331690 is gratefully acknowledged. We would like to thank the members of RESCUE for their helpful suggestions and assistance in testing the infrastructure.

REFERENCES

1. L. J. C., "Automatic self-calibration of the arm microwave radiometers," 2000.
2. V. P. G Bradski, A Kaehler, "Learning-based computer vision with intels open source computer vision library," 2005.
3. P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *cvpr* **01**, p. 511, 2001.
4. S. Scott, "Detecting network intrusion using the markov modulated nonhomogeneous poisson process,"
5. Sun, "Java advanced imaging," *http://java.sun.com/javase/technologies/desktop/media/jai/* .
6. J. McQueen, "Some methodes for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, University of California Press, (Berkeley, CA, USA), 1967.
7. L. Sweeney and R. Gross., "Mining images in publicly-available cameras for homeland security," 2005.
8. C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(8), pp. 747–757, 2000.
9. 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.
10. 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* .
11. 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.
12. I. Lazaridis and S. Mehrotra, "Capturing sensor-generated time series with quality guarantees," 2003.
13. Z. Fu and N. Venkatasubramanian, "Adaptive parameter collection in dynamic distributed environments," *icdcs* **00**, p. 0469, 2001.
14. A. Sang and S. qi Li, "A predictability analisys of network traffic," *Computer Networks* **39**, pp. 329–345, 2002.
15. M. Grossglauser and D. Tse, "A framework for robust measurement-based admission control," *SIGCOMM Comput. Commun. Rev.* **27**(4), pp. 237–248, 1997.