Macroscopic Human Behavior Interpretation Using Distributed Imager and Other Sensors

Dimitrios Lymberopoulos, *Student Member, IEEE,* Thiago Teixeira, *Student Member, IEEE,* and Andreas Savvides, *Member, IEEE*

Abstract

This paper presents BScope, a new system for interpreting human activity patterns using a sensor network. BScope provides a run-time, user-programmable framework that processes streams of timestamped sensor data along with prior context information to infer activities and generate appropriate notifications. The users of the system are able to describe human activities with high level scripts that are directly mapped to hierarchical probabilistic grammars used to parse low level sensor measurements into high level distinguishable activities. Our approach is presented, though not limited, in the context of an assisted living application in which a small, privacy preserving camera sensor network of five nodes is used to monitor activity in the entire house over a period of 25 days. Privacy is preserved by the fact that camera sensors only provide discrete high-level features, such as motion information in the form of image locations, and not actual images. In this deployment, our primary sensing modality is a distributed array of image sensors with wide-angle lens that observe people's locations in the house during the course of the day. We demonstrate that our system can successfully generate summaries of everyday activities and trigger notifications at run-time by using more than 1.3 million location measurements acquired through our real home deployment.

Index Terms

Activity Grammars, PCFG, Behavior Recognition, Camera Sensor Networks.

I. Introduction

The proliferation of wireless sensor networks is rapidly making the collection of overwhelming amounts of data possible. Scientists are already using this data to gain a better understanding of physical processes, large ecosystems and the behavior of different species. In military applications, information from otherwise hard to reach places is facilitating the efficient collection of intelligence related to detection, identification, tracking and observation of interests. From a data interpretation perspective, the above efforts have yielded significant contributions in the efficient transport of raw data and trip-wire detections. In most of these situations, however, the data reaches a human interpreter in raw or aggregate form before it is fully understood. In this paper we argue for a different

D. Lymberopoulos, T. Teixeira and A. Savvides are with the Department of Electrical Engineering, Yale University, New Haven, CT, 06511, USA e-mail: {dimitrios.lymberopoulos,thiago.teixeira,andreas.savvides}@yale.edu

set of applications, closer to everyday life, where sensor data needs to be *understood* by the network, without the human in the loop, in order to provide responsive services.

In this paper we present BScope, a run-time architecture that processes streams of timestamped sensor data along with prior context information to infer macroscale activities and generate appropriate notifications. BScope operates on top of a lightweight camera sensor network whose main task is to detect, localize and track people in a scene [1]. Unlike other camera systems however, our approach does not look at body gestures but at more macroscopic (higher level) activities conducted in the context of a building or a city map. Instead of applying complex image processing techniques to maximize the information extracted from each camera sensor node, we use lightweight algorithms [2] to detect a person's position with respect to a set of predefined areas in a building. The power of our approach comes at the network level, where all the different locations recorded at the different camera sensor nodes are linked together over time to characterize the monitored person's macroscale behavior. In practice, our system finds macroscopic gestures, that take place over larger space and time intervals compared to body gestures, by using basic location information provided by a distributed set of camera sensors.

In BScope, users describe activities as a collection of rules with spatial and temporal characteristics expressed in high level script form. Each activity description has well-defined inputs and outputs enabling the creation of a library of activity components that can be connected together into hierarchies to provide even more complex interpretations. The power of this system is demonstrated using a real camera sensor network deployment for assisted living and a 25-day location dataset collected from this deployment. In this setup, descriptions of human behavior patterns are used to generate summaries of daily activities and to trigger various levels of alarms as well as cell-phone and email notifications. In addition to human observation, another notable aspect of our system is that it uses the same framework to define data and system consistency checks. By combining sensor data with network metadata, the system executes a continuous consistency check that verifies that the network is operating correctly. When the outputs do not match the designer's pre-specified expectations, an additional set of notifications is generated. Although our presentation is described in the context of a home setting environment, our architecture is more general and can be directly applied in other application domains such as, workplace safety, security, entertainment and others.

This paper makes three main contributions. First, it presents a run-time architecture that utilizes the inference power of Probabilistic Context-Free Grammars (PCFGs) organized in hierarchies, to classify spatial and temporal patterns from simple low-level sensor measurements. To achieve that, the architecture defines a set of conventions for dealing with space and time. Second, it demonstrates the feasibility of this approach in a real world assisted living application. The concepts presented here are applied to classify behaviors as well as to supervise the network and verify that it is functioning according to the designer's specification by using a small set of predefined rules. Third, in our architecture the programmer enters the patterns in the form of high level scripts containing rule and grammar specifications, thus abstracting the low-level node programming details. This approach biases the required programming effort towards domain expertise rather than embedded systems, networking and database programming.

Our presentation is organized as follows: Section II provides an overview of the related work and highlights the main contributions of our work. The next two sections motivate the choice of the inference engine and provide some background information. In Section IV we present our camera sensor network deployment in the actual house of an elder person and describe in detail the hardware and software infrastructure used. Section V presents the methodology for converting a sensor network into an interpretation mechanism and shows the main idea of the proposed system architecture. In Section VI an in depth analysis of the design challenges for embedding the selected inference engine into a run-time sensor network architecture is presented and a high-level description of our main system design contributions is given. Section VII describes in detail the BScope architecture through illustrative examples based on an assisted living scenario and Section VIII shows how the same architecture can be used to diagnose system faults. Section IX presents the evaluation of the system on a 25-day dataset acquired from a real home deployment and Section X concludes the paper.

II. Related Work

As in the Semantic Streams work presented in [3] the system presented here generates a set of intermediate level semantics from raw sensor measurements. Our handling of the data across a hierarchy of grammars is similar to the notion of having semantic streams. Our approach however is different in the sense that it provides a structured bottom-up processing of the sensor data that is not application specific. In contrast, the Semantic Streams work follows a top-down approach that focuses more on the end-user programming interface of the sensor network rather than the actual inference engine that generates the high level semantics on which the programming abstraction is based.

The interpretation of human activity has been previously considered in assisted living applications. Researchers at Intel Research have considered assisted living applications using RFID tags [4], [5], [6], [7], [8]. This approach requires a person to wear an RFID reader and extensive tagging of objects or people with RFID tags. Other researchers, attempted similar monitoring with video cameras [9]. While our work is absolutely compatible and it could be transparently used with these types of network setups, it makes a significant contribution: it provides a common, structured framework for describing and identifying spatiotemporal patterns in low level sensor data.

Probabilistic grammars, the centerpiece of our system, have been previously successfully applied by computer vision researchers to detect persons that are picked-up or dropped-off in a parking lot [10], identify human gestures [11] or human interactions [12], [13], recognize high-level narratives of multi-player games and player's strategy [14], and to classify different types of interactions(dancing, hugging etc) between two people [15]. What has not been done before is: (1) the application of such grammars on multimodal sensor networks, and the creation of a hierarchy that reduces reasoning training requirements and (2) the design of an architecture with space and time conventions that can be directly used by grammars.

Sensor networks for abnormal activity detection have also been proposed [16], [17]. In this approach, statistical analysis of long-term real data is used to define what a "normal" activity is. Every activity that deviates from the "normal" activity profile is considered to be "abnormal". While this method can be useful, it does not provide

enough information about the exact service that has to be triggered by the system. Different types of abnormal activities require different types of services to be triggered. Furthermore, in many cases it is very useful to be aware of the exact activities of a person even though these activities are not considered to be "abnormal". For instance, a sensor network that can understand human behaviors could be used to assist elders living alone in multiple ways. It could post reminders and provide warnings when someone engages in unsafe behaviors.

A. Our Work

The purpose of our system is to hide the low level details of the sensor network by exposing a high level interface where multiple users can quickly describe their own activities of interest in a script-like form. This allows the inference engine of the sensor network to become easily configurable and scalable supporting various types of applications and users. In that way, the proposed architecture can be used to generate multiple system instances for different applications while providing a common, powerful interface.

At the same time, the syntactic pattern recognition approach that our system adopts, enables the users to exploit the structure of the activities that have to be identified. Instead of "blindly" training a tool to recognize an activity based on statistical data associations we provide a method for *defining an activity syntax over sensor data*. The hierarchical organization of the inference engine allows to confine training at the lower levels of the hierarchy, facilitating the process of describing (instead of directly training) the syntax of higher level human activities. This approach reduces the training requirements by giving more control to the actual programmer to define the internal structure of each activity. However, this does not limit our training capabilities by any means. When training is required, probabilistic grammars offer the same training capabilities as HMMs or similar pattern recognition tools [18], [19], [20], [21]. These training capabilities have been extensively studied and demonstrated especially in the context of natural language processing and speech processing [18], [19] and are out of the scope of this paper.

In addition to describing the design and implementation of an end-to-end system for mapping the data recorded from a collection of sensors into distinguishable high-level human activities, the work described in this paper makes another significant contribution. It describes a time abstraction architecture that allows the combination of continuous time information with the strict formalism of grammars. Human activities take place over space and time. Even though discretization of time is relatively easy, as we will show in Section VI, discretizing and encoding time in the context of Probabilistic Context-Free Grammars (PCFGs) is a challenging task because PCFGs do not provide the necessary formalism for manipulating continuous numerical values. Our system addresses this problem through a user-configurable and grammar independent time abstraction layer that transparently enables the concurrent support of multiple time scales.

III. Motivation and Background

In order to translate raw sensing data to high level interpretations, such as human activities, an inference engine with the ability to automatically discover patterns in the data and logically label them is required. In general, the different pattern recognition design approaches that can be used, can be divided into two broad categories [19],

[20], [21]: (1) the decision-theoretic approach and (2) the syntactic approach. In the case of the decision-theoretic approach (neural networks etc.) a *decision function* that classifies an input pattern to one of two or more different pattern classes has to be mathematically defined. Usually, the parameters of this function are trained with a well-chosen set of representative patterns and the system is expected to perform satisfactorily on "real" data during normal operation. This approach is ideally suited for applications where patterns can be meaningfully represented in vector form [19]. However, in many applications, as in the case of human activity detection, the structure of a pattern plays an important role in the classification process. For instance, several different human activities can be decomposed to the same set of primitive actions. The order with which these primitive actions take place can be used to differentiate between different activities. In these situations, the decision-theoretic approach has serious drawbacks because it lacks a suitable formalism for handling pattern structures and their relationships. Conversely, the basic concept of the syntactic pattern recognition approach (Probabilistic Context-Free or Context-Sensitive Grammars) is the decomposition of a pattern into sub-patterns or primitives. These primitives are used as a basic vocabulary over which different grammars can be defined, each describing a different pattern. This process is analogous to natural language, where grammars over an alphabet of letters or words can be used to describe different words or sentences respectively. As we will show in section III-B, the probabilistic aspect of the grammars allows them to parse out the most likely parse tree among multiple competing alternatives.

Similar functionality to grammars could be achieved by the more widely-used Hidden Markov Models (HMMs). Mathematically, HMMs are very similar to Probabilistic Context-Free Grammars [19], [20]. In practice, this means that the recognition power and training capabilities of both tools is the same [18]. However, PCFGs provide a much more expressive interface than HMMs. The expressiveness and generative power of grammars allows the derivation of a large number of HMMs from a compact set of rules. This is very important in the case of sensor networks where a high level programing interface is required.

Fusing the powerful grammar formalism with the extensive monitoring capabilities of sensor networks is as challenging as appealing. Sensor networks have been demonstrated to provide continuous streams of data that are distributed over space and time. Grammars alone cannot perform the task of parsing continuous spatiotemporal data. First, a run-time architecture responsible for parsing the data as it comes in to the system is required. Second, a time abstraction for grammars must be developed since grammar's formalism does not directly support handling of numerical time values. In addition, a sensing abstraction mechanism is required that will hide the complexity of the sensing layer by encoding information from multiple sensors into a common form that fits the grammar processing model.

A. Scope of This Paper

The BScope system provides a generic middleware architecture that programmers can use in different application domains and with different sensing modalities to define and extract patterns from low level spatiotemporal sensing data. In this paper, we demonstrate the capabilities of the proposed architecture in the context of an assisted living deployment. We would like , however, to emphasize that the core architecture of the BScope system has been

designed independently of this application domain. The same system could be used with minor modifications in other domains such as security, workplace safety, entertainment and more with a diverse set of sensing modalities. In our pilot deployment cameras were utilized for monitoring a person inside a house because of the rich information and large coverage they provide. However, any other collection of sensors (i.e. RFIDs, ultra-wideband, break-beam, water, contact, pressure etc.) could be used instead of cameras, with no changes in the architecture or the specified grammar definitions.

In the system description that follows the sensor data is centrally processed. This process, however, can take place anywhere in the sensor network as was initially suggested in [22].

B. Probabilistic Context-Free Grammars (PCFGs)

A probabilistic context-free grammar G [23], [18] is an ordered five-tuple $\langle V_N, V_T, Start, Pr, P\rangle$ where:

- V_N is a finite set of non-terminal symbols.
- V_T is a finite set of terminal symbols.
- $V_N \cap V_T = \emptyset$. $V = V_N \cup V_T$ is called the vocabulary. V^* is the set of all strings of symbols in V including the string of length 0.
- $Start \in V_N$ is the start symbol.
- Pr is a finite nonempty subset of $V_N \times V^*$ called the production rules.
- The production rules are paired with a set of probabilities $P = \{p_{ij}\}$ that satisfy the following rules:
  1) For each production $P_{ij} \in Pr$ there is one and only one probability $p_{ij} \in P$.
  2) $0 < p_{ij} \leq 1, \forall i, j$
  3) For every i with $1 \leq i \leq |V_N|$: $\sum_{1 \leq j \leq n_i} p_{ij} = 1$, where n_i is the number of productions with the i^{th} non-terminal on the left-hand side.

Let capital letters: $A, B, C, \ldots$ represent the non-terminal symbols and small letters: $a, b, c, \ldots$ represent the terminal symbols. The production rules of a context-free grammar are then written as: $A \rightarrow a^{(0 \leq p \leq 1)}$, where the left-hand side can be any non-terminal symbol and the right-hand side can be any combination of terminal and non-terminal symbols. The exponent p denotes the probability assigned to the production rule.

Starting from the start symbol $Start$ and by successively applying the same or different production rules, different strings can be generated. In general, we say that string α derives string β ($\alpha \Rightarrow \beta$) if there is a sequence: $\alpha = \alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_n = \beta, n \geq 0$, of strings in V^* such that: $\alpha_0 \rightarrow \alpha_1, \alpha_1 \rightarrow \alpha_2, \ldots, \alpha_{n-1} \rightarrow \alpha_n$. The language $L(G)$ generated by a probabilistic context-free grammar G is the set: $L(G) = \{x | Start \Rightarrow x, x \in V_T^*\}$. In other words, $L(G)$ is the set of all terminal strings derivable from the start symbol $Start$.

Since every production rule is assigned a probability, any string of terminal symbols derivable from the $Start$ symbol is assigned a probability. The basic assumption is that the choice of production rules used in deriving a sentence is "context-free" in the sense that each rule is chosen independently of all the others in the derivation. This allows us to compute the probability of a sentence as the product of the production probabilities that were

(a) House deployment floorplan **(b)** Nodes 4, 6 deployed **(c)** The field of view for nodes 4(bottom) and 10(top)

Fig. 1. Pilot network deployment overview.

used to generate this sentence. If the same string can be derived in more than one ways then the derivation with the highest probability wins.

For instance, given the following PCFG:

$$V_T = \{a, b\}$$
$$V_N = \{A, B\}$$
$$Start \rightarrow A^{(0.5)} \mid B^{(0.5)}$$
$$A \rightarrow A\,a^{(0.5)} \mid a\,b^{(0.5)}$$
$$B \rightarrow B\,b^{(0.5)} \mid a\,b\,a^{(0.5)}$$

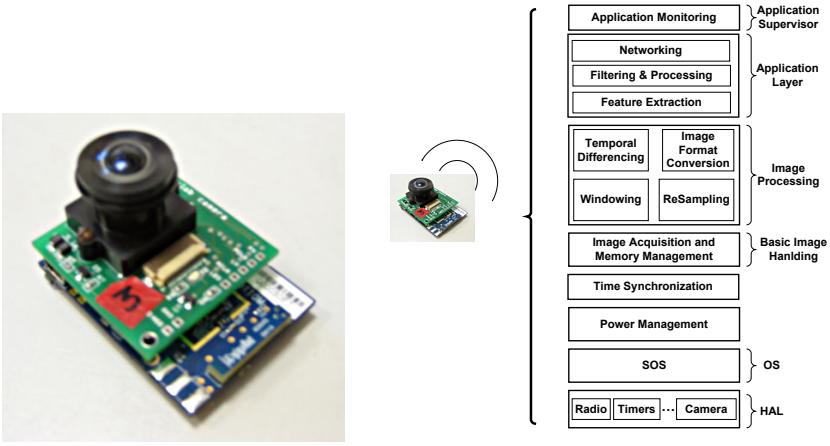
the string of terminals "$a\,b\,a$" can be derived from the $Start$ symbol in two different ways:

(1): $S \rightarrow A^{(0.5)} \rightarrow A\,a^{(0.5)} \rightarrow a\,b\,a^{(0.5)}$, with a derivation probability of: $0.5^3 = 0.125$

(2): $S \rightarrow B^{(0.5)} \rightarrow a\,b\,a^{(0.5)}$, with a derivation probability of: $0.5^2 = 0.25$

In this case the most probable derivation of the input is the second one and therefore the input sequence is mapped to the nonterminal "B". Note, that by changing the probabilities assigned to the production rules, we can change the most probable parse of the input.

In the rest of the paper we do not focus on demonstrating the probabilistic inference of grammars. Instead, we emphasize on how to write a grammar and on how to properly configure the proposed architecture to support multiple, concurrently running grammars. As a result of this and to simplify our discussion, we will always assume a uniform probability distribution for the production rules and the actual probabilities will not be shown. However, a rigorous example that demonstrates the probabilistic inference of grammars can be found in [24].



(a) (b)

Fig. 2. (a) A camera-enabled imote2 wireless sensor node (b) Sensor node software stack.

IV. ASSSISTED LIVING DEPLOYMENT

To evaluate the BScope system in a real environment we created a pilot camera sensor network deployment in a two-floor house. Five Intel $iMote2$ [25] sensor nodes sporting a COTS camera module we have designed for this application and a single basestation node were deployed in the house for 25 days using the configuration shown in Figure 1. The camera nodes are attached to the ceiling with the camera lens pointing down into the room. Each camera node (Figure 2(a)) features an OV7649 VGA camera from Omnivision coupled to a 162 degree lens. Images are acquired at a rate of 8 frames per second, downsampled to an 80×60 resolution and used as input to a lightweight image processing algorithm we have developed to count the number of people and extract their locations inside the house [2]. The computed image locations are then combined with house map information to produce the real word locations/areas visited by the monitored person. These sequences of locations/areas visited over time become the actual input to the BScope system that hierarchically translates them to human activities and behaviors.

The PXA271 processor aboard the iMote2 runs the SOS operating system from UCLA [26] in which we have implemented the corresponding camera drivers, a small image processing library and the software required by the BScope system. Figure 2(b) provides a detailed description of our software stack. To ensure that the ordering of measurements from different nodes is preserved, we have implemented a lightweight time-synchronization mechanism that synchronizes the real-time clocks (RTCs) of all the nodes to that of the base station node within 1 second of accuracy. All the timestamped locations are routed back to the base station node and they are stored into a database. At the application layer the sequence of captured images is used to extract basic features that provide rudimentary information about the monitored person's macroscale activities [1]. To simplify this process we have implemented a small image processing library that offers basic windowing, temporal differencing, image format conversion and re-sampling capabilities. At the top of our software stack an application monitoring mechanism

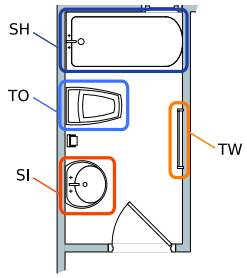


Fig. 3. A typical bathroom layout.

observes the state of the sensor node and upon the detection of exceptions, abnormal behavior or node crashes, it immediately reboots the sensor node. At the same time, the software running on the basestation is also responsible for tracking the state of each sensor node in the system. To make this possible, when a node has no data to transmit, it periodically transmits a *HEARTBEAT* message that contains statistics locally recorded at the node. These are collected at the basestation and compared against the statistics observed at each node to determine where packet losses occur in the system.

V. The Sensor Network As An Interpretation Engine

A. Encoding Sensory Information

According to the formal definition given in the previous section, grammars are defined on top of a *finite set of discrete symbols* called terminal symbols. In the case of sensor networks, terminal symbols correspond to the detected sensing features that we call *phonemes*. Raw data collected from the sensors is directly mapped into *phonemes*: a set of symbols that imply high degree of correlation to the specific action we wish to identify. For instance, consider the case of monitoring the bathroom visits of an elderly person living alone in a house. Given the bathroom layout shown in Figure 3, we would like to identify when and how this person is actually using the bathroom. In a typical scenario the elderly person would initially visit the toilet, then the sink to wash hands and he would finally dry his hands using one or more towels. This high level specification of a bathroom visit can be directly mapped to a very simple grammar description. The *phonemes* over which this grammar is defined will be the toilet, sink and towel areas that we denote with the symbols TO, SI and TW respectively. The actual grammar definition is shown in Figure 4. Two types of bathroom visits are defined: $NormalBathVisit$ and $IncompleteBathVisit$. The former has two possible definitions. It is described either as the ordered sequence of the toilet, sink and towel phonemes or as the ordered sequence of the toilet and sink phonemes. The second definition is used to identify the cases where the person chooses to dry his hands outside the bathroom. The $IncompleteBathVisit$ is defined as a single appearance of the toilet phoneme and it can be used to remind the elder person to wash hands before leaving the bathroom.

In the same way we can identify toilet activity we can also identify shower activity. All we have to do is expand the set of phonemes to include the shower area phoneme (denoted with the symbol SH) and add the following

Input: Any sequence of the phonemes:$\{TO\,, SI\,, TW\}$

Output: A sequence of any of the following non-terminal symbols: $\{NormalBathVisit, IncompleteBathVisit\}$

1. V_N	$=$	$\{Start, NormalBathVisit, IncompleteBathVisit\}$
2. V_T	$=$	$\{TO, SI, TW\}$
3. $Start$	$\rightarrow$	$NormalBathVisit \,\vert\, IncompleteBathVisit$
4. $NormalBathVisit$	$\rightarrow$	$TO\ SI\ TW \,\vert\, TO\ SI$
5. $IncompleteBathVisit$	$\rightarrow$	TO

Fig. 4. A grammar for identifying bathroom visits.

production rule:

$$ShowerVisit \rightarrow ShowerVisit\,SI \,|\, ShowerVisit\,TW \,|\, SH\,TW$$

According to this definition, the core of a shower visit is defined as first using the shower and then a towel to dry the body. However, this definition is using recursion to be able to identify more general shower visits where one first takes a shower and then have visits to the sink to perform another activity such as brushing his hair, shaving etc.

In our simple example, the areas of interest inside the bathroom became the actual phonemes. To detect other activities we may need to specify a different set of phonemes that provide us with rudimentary information about the activity. For instance, in the case of home monitoring, we would like to monitor the usage of basic rooms such as kitchen, bathroom, bedroom, living room, etc. If we require a finer level of sensing granularity we might want to monitor specific areas in a room such as the stove, the dining table, the refrigerator or the sink inside the kitchen. Note that the actual grammars operating on top of these phonemes do not need to know any information about the type of sensor used to detect these phonemes. The only information that is crucial to the grammars is the type of phonemes detected and not how these phonemes are actually detected. For instance, to detect if the monitored person uses the stove, several different sensors could be used varying from current sensors to see if the stove is on, door sensors to see if the stove door was opened/closed, or even a camera sensor which can associate the image location of the person with the location of the stove in the floor plan and infer if the monitored person is actually using the stove. Given a grammar definition that operates on top of the stove phoneme, any type of sensor or even combination of sensors can be used to detect the stove phoneme transparently to the grammar definition. In other words, any changes to the underlying sensing layer are completely transparent to the grammar definitions as long as the type of phonemes detected remains intact.

The different phonemes detected at the sensor level could be used in many different ways to describe basic activities that take place inside a house. This can be better seen in Figure 5 where all the phonemes that were recorded by the camera sensor network, described in Section IV, in a 4-day time-window are shown over time. By simply inspecting the sequence of rooms and areas that the person visited over time inside the house, it is clear

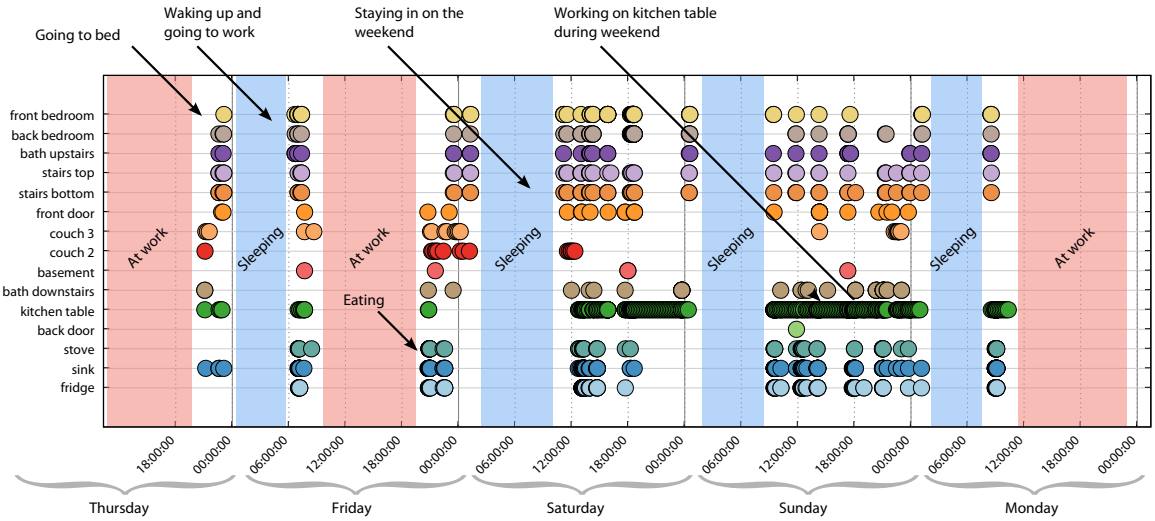


Fig. 5. Various phonemes recorded over a four day period.

that patterns, strongly related to the person's activities, start emerging. The sleeping pattern, the morning and night activity patterns, that take place right after waking up and right before going to sleep respectively, of the monitored person can be clearly seen. The bathroom visits along with the time that the person spends working in the kitchen or the time and duration that the person is away at work during weekdays provides invaluable information about the person's daily living habits.

In general, in the context of our system *behaviors are defined as a set of actions in space and time. The goal of phoneme generation is to instrument the environment with an appropriate set of sensors so that a string of phonemes containing information about specific behaviors can be extracted*. The application designer is free to choose a heterogeneous set of sensors to extract an appropriate set of phonemes from the environment. For example, the same area phonemes we have previously defined could be generated with a heterogeneous set of sensors. The toilet phoneme (TO) could be detected by associating location information to a building map or by using a variety of other sensors such as, pressure, contact or motion sensors. In the same way, the shower and sink phonemes (SH and SI) could be detected by using contact or water sensors. In general, in a home setting, one could use a variety of sensors ranging from RFIDs to door and current sensors to generate phonemes about the interaction of people with objects in their immediate environment, generate opening and closing phonemes and detect appliance usage.

All these different types of *phonemes* form the vocabulary that allows us to specify human behavior languages, by providing a powerful sensing abstraction that enables the use of a heterogeneous set of sensors to extract information about an activity. *The power of the abstraction comes from the fact that it hides the complexity of sensing at a lower layer, and thus does not require us to fuse sensor measurements from multiple modalities*.

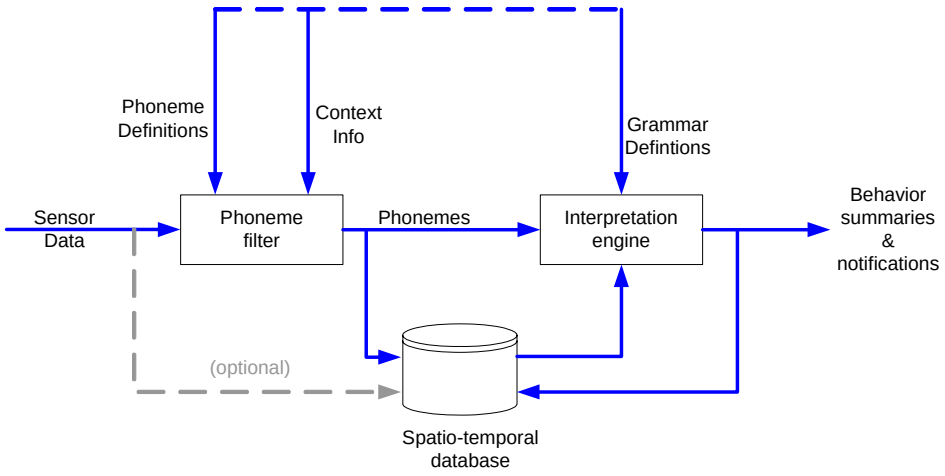


Fig. 6. Data interpretation process.

B. System Process Overview

Figure 6 depicts the main datapath of the proposed system architecture. The time-stamped raw sensor data, collected by the sensor network, passes through two main processing blocks. The first block is the phoneme filter that combines raw sensor data with a set of user-specific phoneme definitions and context information, such as building maps, to generate a stream of time-stamped phonemes. The second block passes the incoming phoneme stream through a set of hierarchical probabilistic parsers defined by the programmer. These parsers automatically discover spatiotemporal phoneme patterns and create higher level, meaningful summaries of activities. At the same time, they detect actions of interest or alarms and they trigger user-specific e-mail or cell-phone notifications.

VI. Considering Space and Time

According to the description provided in Section V-A, phonemes are nothing more than a discretization of the observed human activities. In general, human activities are expanded over two dimensions: space and time. Discretization of space can be achieved if absolute or relative location information is associated to building map information. Using this approach, human activities can be expressed as a trace of spatial phonemes over time. Time discretization, on the other hand, is not straightforward. Time can only be discretized by using numerical values that represent either actual timestamps or time duration between successive timestamps. However, grammars (as well as HMMs) do not provide the necessary formalism for manipulating numerical values. Grammars operate on a set of symbols and they can only check the validity of different sequences of these symbols. Attributes of these symbols, such as numerical values, cannot be internally handled unless a different symbol is defined for every different numerical value. Obviously, this approach is feasible only when a single time scale representation is required across all different grammar hierarchies. However, different grammars often require representation of time at different scales depending on the application they focus on. This requirement leads to a large number of input symbols that makes the process of defining and running grammars infeasible. Thus a more suitable abstraction of

time is required.

For instance, consider the following example where a sensor network is deployed inside a house to monitor an elderly person that lives alone. Several grammars running in the system monitor different activities of that person including, cooking, sleeping, visiting the bathroom, etc. At the same time, another set of grammars is used to identify emergency situations and generate alarms. In the case of cooking activity detection, the timing information needed can vary from minutes up to hours depending on the meal that is prepared. The sleeping grammar requires time information that can vary from approximately 30 minutes to 8 or 10 hours. The grammar used for monitoring bathroom usage might require time information in the order of seconds or tens of minutes. On the other hand, the grammars used for emergency condition detection can use a different time scale to generate alarms using the same spatial phonemes as the previous grammars. For instance, if the person has been in the bathroom for more than an hour then most probably something is wrong. In the same sense, if no movement has been detected for more than 30 minutes in the house and the person is not sleeping then an alarm should be generated. Based on this simple example it is clear that:

(1) Time information plays an important role in the recognition of human activities. The same sequence of spatial phonemes might be interpreted in a different way based on the time associated to it.

(2) Different time scales with large variations have to be concurrently supported.

A. Time Augmented Grammars

The BScope system implements a user-configurable and grammar independent time abstraction layer that transparently enables the concurrent support of multiple time scales. Instead of trying to directly discretize time space and feed it as input to grammars, we associate time information to spatial phonemes *before grammar execution*. Time information is used in the time abstraction layer to generate spatiotemporal symbols based on user-specific parameters. This approach, encodes space and time information into symbols *outside the grammar* and on a *per-grammar basis* providing two main advantages:

(1) Since the input symbols encode both time and space information, grammars can easily generate higher-level spatiotemporal semantics without having to explicitly parse time information.

(2) The time abstraction layer decouples the different time scales that the system has to support from the time scale that a specific grammar requires. In practice this means that the grammar designer has to worry only about the time scale that his grammar requires and nothing else.

As a result of this, the number of different symbols that a grammar is defined upon is dramatically reduced facilitating the process of describing and running grammars.

Figure 7 shows a high-level overview of the timing abstraction mechanism in the BScope system. Area phonemes along with their timestamps are provided to the time abstraction layer which transforms them to spatiotemporal symbols according to user-specific rules. The new spatiotemporal symbols have both start and end timestamps that define their time duration. Note, that only the spatiotemporal symbols are fed as input to the grammar. The pairs of timestamps for each input symbol are used to derive the ground truth start and end timestamps for the

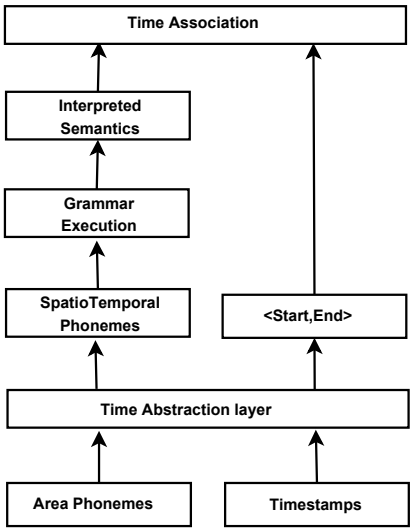


Fig. 7. Overview of the time abstraction layer.

generated higher level semantics. The fact that the output semantics of every grammar are mapped to ground truth time facilitates the creation of time-based semantic hierarchies. In practice, this means that a grammar can always associate time information to its input phonemes independently of its position in the hierarchy. In that way, time information is hierarchically associated to input phonemes creating a hierarchical spatiotemporal semantic generation mechanism.

To better illustrate the capabilities and flexibility of the proposed time abstraction layer let us consider again the example grammar specification shown in Figure 4. While this grammar can provide information about how often a person visits the toilet, it cannot provide any information about the length of individual bathroom visits. For instance, when an elderly's person visit to the toilet lasts more than 15 minutes, then that person either has difficulties getting up or is sick. Using the features of the timing abstraction layer we can easily add temporal information to the detected semantics while minimizing the changes in the actual grammar configuration. Figures 8 and Figures 9 provide two different alternatives.

In Figure 8, time information is encoded to the actual input phonemes as Lines 1 and 2 show. These two lines are **not** part of the grammar description. Instead, these two lines provide configuration parameters to the time abstraction layer. The first line, defines a time condition on the input symbol TO according to which, every TO phoneme with duration period longer than 900 seconds will be renamed to TO_LONG. Note that now, the TO_LONG phoneme encodes both spatial and temporal information. Line 2, instructs the time abstraction layer to pass to the grammar only those phonemes that satisfy at least one of the previous $Time\_Replace$ statements. This will result in ignoring all the TO phonemes with duration time less than 900 seconds. In that way, only the detected TO_LONG phonemes will be fed as input to the grammar. The rest of the lines in Figure 8 are identical to the initial grammar definition. The only difference now, is that the grammar operates on the TO_LONG phonemes instead of the TO phonemes. As a result of this, the output of the grammar shown in Figure 8 will include only

Input: Any sequence of the phonemes:$\{TO\,, SI\,, TW\}$

Output: A sequence of any of the following non-terminal symbols: $\{NormalBathVisit, IncompleteBathVisit\}$

1. $Time\_Rule: TO > 900$	$\rightarrow$	TO_LONG
2. $Use\_Time\_Phonemes\_Only:$	$\rightarrow$	YES
3. V_N	$=$	$\{Start, NormalBathVisit, IncompleteBathVisit\}$
4. V_T	$=$	$\{TO\_LONG, SI, TW\}$
5. $Start$	$\rightarrow$	$NormalBathVisit \mid IncompleteBathVisit$
6. $NormalBathVisit$	$\rightarrow$	$TO\_LONG\ SI\ TW \mid TO\_LONG\ SI$
7. $IncompleteBathVisit$	$\rightarrow$	TO_LONG

Fig. 8. Time augmented grammar for identifying abnormally long bathroom visits.

Input: Any sequence of the phonemes: $\{Normal\,, Incomplete\}$

Output: Any of the following non-terminal symbols: $\{NormalLongBathVisit, IncompleteLongBathVisit\}$

1. $Time\_Rule: NormalBathVisit > 900$	$\rightarrow$	$Normal$
2. $Time\_Rule: IncompleteBathVisit > 900$	$\rightarrow$	$Incomplete$
3. $Use\_Time\_Phonemes\_Only:$	$\rightarrow$	YES
4. V_N	$=$	$\{Start, NormalLongBathVisit, IncompleteLongBathVisit\}$
5. V_T	$=$	$\{NormalLongBathVisit, IncompleteLongBathVisit\}$
6. $Start$	$\rightarrow$	$\{NormalLongBathVisit \mid IncompleteLongBathVisit\}$
7. $NormalLongBathVisit$	$\rightarrow$	$Normal$
8. $IncompleteLongBathVisit$	$\rightarrow$	$Incomplete$

Fig. 9. Using a second level grammar for identifying abnormally long bathroom visits.

those toilet visits that are abnormally long.

Another way to achieve the same functionality is shown in Figure 9. In this case, an additional grammar is created that receives as input the output of the grammar shown in Figure 4. Time information is now directly associated to the input phonemes of the second level grammar in exactly the same way as before. Note, that the new grammar definition turns into a trivial symbol renaming procedure due to the fact that all the low level details are taken care of by the timing abstraction layer. More elaborate examples of using the time abstraction layer are given in the next section.

In the example presented in this section the different time parameters associated to the recorded spatial phonemes have to be manually specified by the grammar designer. However, these parameters might vary across different persons or even for the same person over time. Therefore, it will not always be feasible and efficient to manually specify these parameters. Instead, an automated way of learning these parameters directly from the low level data is required. We have designed and implemented such an approach that enables the automatic extraction of temporal characteristics from sensor data streams [27]. However, describing the details of this approach is out of the scope of this paper.

VII. BSCOPE INTERPRETATION ENGINE

The BScope system provides a high-level interface that enables the programmer to write, configure and execute grammar hierarchies in a very simple way. In order for a user to insert a new grammar hierarchy into the system and execute it, the following steps are required:

(1) Specify a single file that provides the exact grammars to be run and the order at which they should be run.

(2) Provide the actual grammar definitions.

(3) Specify a single file describing the input/output of each grammar.

(4) Specify a set of triggers and associate it to one or more grammar hierarchies. Every time that a hierarchy produces an output semantic, all the triggers associated to this hierarchy will be fired. In the current implementation, users can provide their e-mail address and/or cell phone number where a message will be automatically sent including important information about the detected semantic (i.e. time duration etc.).

Given these four pieces of information the BScope system is able to automatically:

(1) Filter the input stream of phonemes to ensure an appropriate input for the grammar.

(2) Create the parsers for every grammar definition and run them with the appropriate input.

(3) Connect the grammars into a hierarchy.

(4) Plot the output of grammar hierarchies and trigger notifications on the detection of semantics of interest.

This process is shown in detail in Figure 10. The programmer is only responsible for providing a collection of grammar hierarchy definitions along with a set of configuration parameters. The BScope engine is continuously monitoring the database for new phonemes detected by the sensor network. As new phonemes become available they are first passed through a generic filtering stage. The purpose of this stage is to deal with the uncertainty of the underlying sensing layer. Several false positives are removed at this step and the length of the phoneme sequence is minimized to reduce grammar execution time. This is done by removing redundant occurrences of phonemes. At the same time, the initial input sequence of phonemes is broken into multiple grammar-specific phoneme sequences. The grammar-specific phoneme sequences contain only those phonemes that the corresponding grammar can actually handle. At the next level, time information is assigned to the phoneme streams and time-based filtering takes place according to the parameters that the programmer passes to the system. After this step, every phoneme stream is assigned to a grammar parser and starting from level 1, all parsers in all hierarchies are executed. The output of each hierarchy first goes through a notification triggering mechanism. Bscope searches if the detected activities match activities of interest defined by the programmers and if this is true the user-defined notifications are triggered. Finally, the outputs of all the grammars are plotted into a single summary-plot. In that way hundreds of thousands of phonemes can be automatically translated into a small summary of activities and a small number of notifications.

A. BScope Configuration

Programmers can configure the BScope interpretation engine to preprocess the input phoneme streams on a per-grammar basis. This is done by using a set of pre-defined configuration parameters. Every grammar description is associated to a configuration file that contains lines of the form "<parameter_name>: <data>". The purpose of

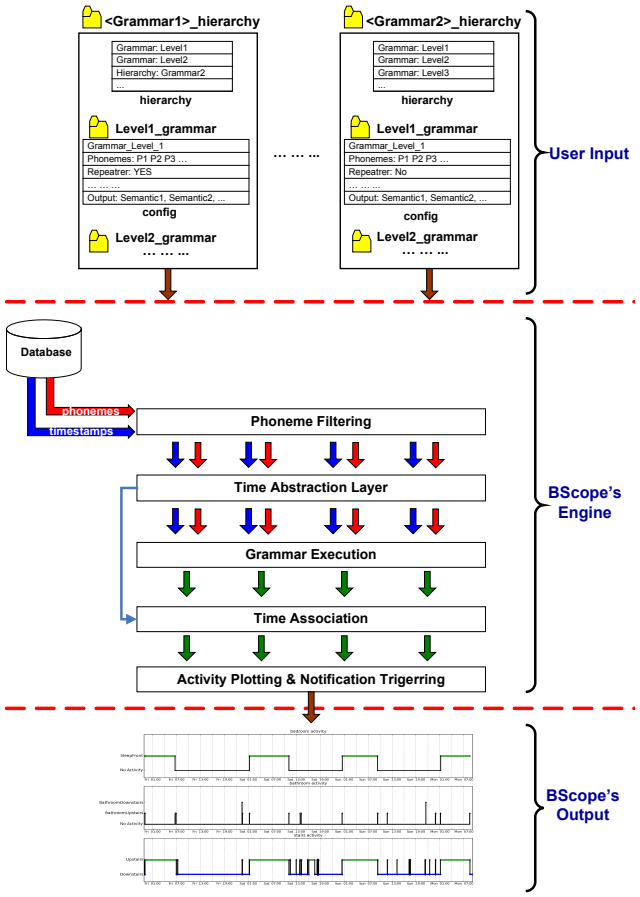


Fig. 10. Overview of the BScope's interpretation engine.

these parameters is to give the ability to the programmer to configure the BScope engine according to the needs of his grammar without forcing him to actually modify the interpretation engine of the framework.

We present the available parameters and their effect on the input of a grammar through two examples: the bathroom grammar shown in Figure 8 and the stairs activity grammar shown in Figure 12. The configuration files for both grammars can be seen in Figures 11 and 13 respectively. The following six parameters are defined:

Phonemes: this parameter defines which of the available phonemes are of interest to the grammar. In the case of the bathroom grammar the phonemes *TO*, *SI* and *TW* are of interest since they represent the toilet, sink and towel areas respectively. In the case of the stairs grammar the phonemes of interest are the *DOWN* and *UP* phonemes denoting the bottom and top areas of the stairs. In practice, this parameter instructs the BScope engine to create a grammar specific phoneme stream for every grammar where only the phonemes of interest are included.

Output: this parameter defines the output semantics produced by the grammar. In the case of the bathroom grammar, *NormalLongBathVisit* and *IncompleteLongBathVisit* are the produced semantics. Note, that the list of output semantics can be a subset of the semantics generated by the grammar. This would force the BScope engine to remove any detected semantics that are not specified in the "Output" parameter from the grammar's output. For

```
# bathroom grammar configuration file
Phonemes: TO SI TW
Merge_Consecutive_Phonemes: NO
Keep_Last_Phoneme: NO
Repeater: NO
Use_Time_Phonemes_Only: YES
Time_Rule: TO > 900 TO_LONG
Output: NormalLongBathVisit IncompleteLongBathVisit
```

Fig. 11. The configuration file for the bathroom monitoring grammar.

instance if we wouldn't like the $NormalLongBathVisit$ to appear in the output of the system, all we have to do is remove it from the *Output* statement without changing the actual grammar definition.

Repeater: when this parameter is set the input is fed to the grammar through a repeater. This process converts the input sequence of phonemes into a new one where every phoneme in the initial input sequence, except the first and last ones, is duplicated. This functionality is extremely important when every phoneme has to be matched with both its immediately previous and next phonemes. For instance, in the case of the stairs activity example, a typical input phoneme stream could be the following: $DOWN\ UP\ DOWN$. This input should be mapped to the following output: $MoveUp\ MoveDown$. However, given the input and the stairs grammar definition the desired output will never be provided unless the input is first passed through a repeater. Now the output of the repeater: $DOWN\ UP$ **UP** $DOWN$ can be successfully mapped to the desired output by matching the first two phonemes into a $MoveUP$ activity and the last two phonemes into a $MoveDown$ activity.

Merge_Consecutive_Phonemes: After creating a grammar-specific phoneme stream according to the "'Phonemes" parameter, phonemes might not be continuous in time. When this parameter is set, consecutive appearances of the same phoneme are grouped into a single phoneme *even if these phonemes were not recorded at consecutive time instants*. The duration of the new phoneme becomes the sum of the durations of the merged phonemes. This parameter is used to minimize the size of the grammar input when fine-grained timing is of no importance to the actual grammar.

Keep_Last_Phoneme: This parameter is used only when the "Merge_Consecutive_Phonemes" parameter is set. When it is set, it has the same effect with the "Keep_Last_Phoneme" parameter with the difference that the time duration assigned to the phoneme produced by the merging process is equal to the time duration of the last merged phoneme. When this parameter is not set it has absolutely no effect on the input.

Time_Rule: this parameter is used to define time rules based on the duration of a phoneme. Every "Time_Rule" statement has the following format: <original_phoneme> <comparison_operator> <duration_in_seconds> <new_phoneme>. If the duration of the <original_phoneme> satisfies the time condition then the <original_phoneme> is *replaced* by the <new_phoneme>. In the case of bedroom grammar, any appearance of the toilet phoneme *TO* that has a duration equal or larger to 900 seconds is replaced by the spatiotemporal phoneme *TO_LONG*. Note that when <duration_in_seconds> is equal to 0 a time rule is basically a renaming rule. This can be very useful when one

Input: Any sequence of the phonemes:$\{DOWN, UP\}$

Output: A sequence of any of the following non-terminal symbols: $\{MoveUp, MoveDown\}$

1. V_N	$=$	$\{Start, MoveUp, MoveDown\}$
2. V_T	$=$	$\{DOWN, UP\}$
3. $Start$	$\rightarrow$	$MoveUp \mid MoveDown$
4. $MoveUp$	$\rightarrow$	$DOWN\,UP$
5. $MoveDown$	$\rightarrow$	$UP\,DOWN$

Fig. 12. Grammar definition for detecting stair traversals.

```
# stairs grammar configuration file
Phonemes: DOWN UP
Merge_Consecutive_Phonemes: YES
Keep_Last_Phoneme: YES
Repeater: YES
Use_Time_Phonemes_Only: NO
Output: MoveUp MoveDown
```

Fig. 13. The configuration file for the stairs grammar.

or more phonemes are translated in exactly the same way by the grammar. In this case, the renaming rule can be used to map all these phonemes to a single one, thus facilitating the grammar definition.

Use_Time_Phonemes_Only: when this parameter is set only the phonemes that were generated by the "Time_Rule" statements (<new_phoneme>) are considered to be valid input for the grammar. When this parameter is not set, the valid set of input phonemes is expanded to include all the phonemes that were **not** mapped to a new spatiotemporal phoneme. For instance, in the case of the bathroom grammar shown in Figure 8, disabling the "Use_Time_Phonemes_Only" parameter would result in providing as input to the grammar two types of toilet phonemes. All the toilet phonemes with a duration that exceeds 900 seconds will be mapped to a TO_LONG phoneme and all other toilet phonemes will still be represented by the initial TO phoneme. In that way, we can easily expand the grammar to identify both normal and abnormally long bathroom visits by also parsing all the TO phonemes.

B. Run-Time Grammar Execution

An important challenge in designing the Bscope system is the ability to know **when** parsing has to take place. Since the input phoneme stream generated by the sensor network is continuous in time, a way to know when to stop acquiring new phonemes and parse the buffered input is required. Providing a generic solution to the problem is difficult due to the fact that different grammars have different input requirements. The proposed architecture addresses this problem by allowing the user to guide the input segmentation process on a per-grammar basis. In particular, another parameter called "Execute_Condition" is introduced. Every *Execute_Condition* parameter consists

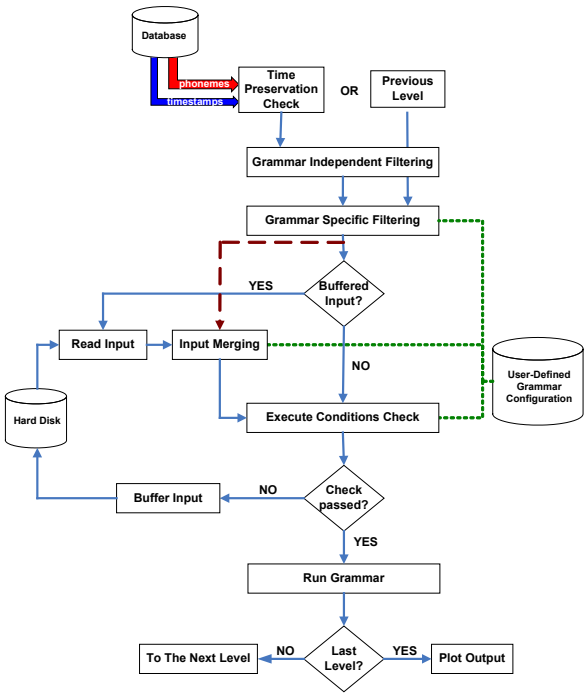


Fig. 14. Overview of the Bscope run-time architecture.

of one or more tuples of the form: $< Phoneme >< number >$. The $< Phoneme >$ can be any input phoneme. The $< number >$ represents the required number of appearances of the $< Phoneme >$ in the input stream, so that the grammar can be safely run. When more than one tuples are given in one *Execute_Condition* statement, the logical OR of all tuples is computed. When more than one *Execute_Condition* parameters are defined then the input is considered to be valid only if all the *Execute_Condition* parameters are satisfied (logical AND). For instance, in the case of the stairs grammar (Figure 12), the input is considered to be valid when it contains at least one $DOWN$ and one UP phonemes:

$$Execute\_Condition : DOWN\ 1,\ Execute\_Condition : UP\ 1$$

Figure 14 provides an overview of the BScope's run-time architecture. If the filtered phoneme stream satisfies the *Execution_Condition* statements described in the grammar configuration file, the grammar is run. Otherwise, the output of the filtering stage is temporarily buffered until more input phonemes are generated in the database. Every time a new sequence of phonemes passes through the filtering stage, the system automatically checks if there is previously buffered input. If this is true, the system merges the old and new phoneme sequences based on the parameters set by the user in the grammar configuration file and then the grammar is run.

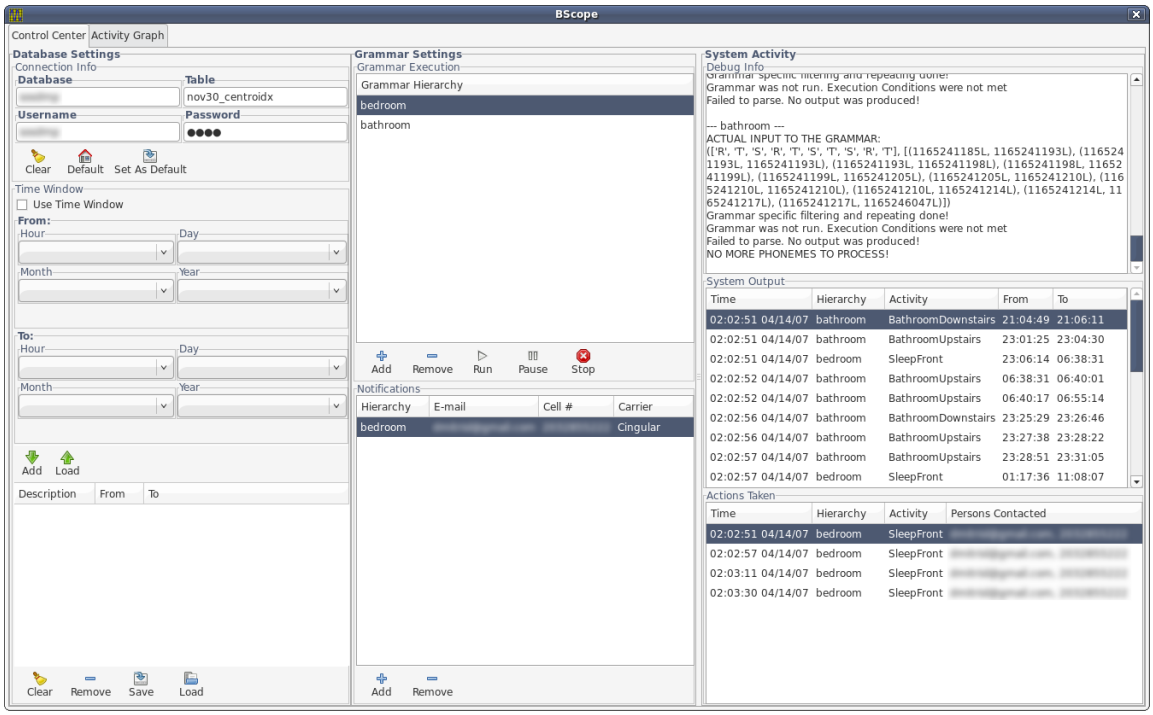


Fig. 15. The main control panel of the Bscope System. Some fields have been blurred to preserve private information.

C. Implementation

The BScope architecture has been fully implemented in Python. The goal of this implementation was to simplify and automate the interaction of the programmer with both the system itself and the sensor network data. Programmers add new grammar hierarchies through an intuitive graphical user interface that guides them step-by-step through the necessary procedure. First they are prompted to enter the configuration parameters that the BScope engine has to be aware of and then they enter the production rules of a grammar in the same format as Figure 4 shows. All the configuration files and low level details are automatically taken care of by the system. Figure 15 shows the main control panel of the BScope system. Users can choose the database where the sensor network data is stored and specify a time window that they want to focus on. In this specific example, two grammar hierarchies for monitoring bedroom and bathroom activity were selected to run. At the same time a single notification request has been inserted to the system. Every time the bedroom hierarchy produces a semantic, an e-mail as well as a cell-phone text message is sent including information about the specific semantic that was detected and its duration. Figure 15, shows the run-time output (the input phonemes were processed in an online manner) of the system on a 4 days window. On the top right, debug and state information of the Bscope Engine is displayed at run time. A list showing all the detected semantics, the grammar from which these semantics were produced and their ground truth time duration follows. At the bottom right, an entry is created every time the BScope system triggers a notification. Note that in the case of the 4 days window, four sleeping activities were detected and therefore four notifications were generated.

VIII. GRAMMAR-BASED SYSTEM CONSISTENCY CHECKING

While according to Figures 15 and 5 the network seemed to operate correctly, we had no indication of the correctness of the results even though we knew that sensing, processing or network errors could significantly impact the results. After a careful examination of the ground truth data we had collected, we were able to identify several errors that affected the correctness of our network's output. Even though it was quite difficult to identify the exact sources of these errors, it was obvious that the output of the system was altered. However, even then we were only aware of the existence of faults in the system at a specific time instant. We were still unable to answer a list of very important questions:

- Which outputs where affected by these faults?
- When were these outputs affected?
- How can we know when future outputs of the system are affected by these or even new faults introduced into the network?

To provide a structured mechanism for describing and running system validation checks we expand the proposed architecture into an automated, in-situ network verification tool. The users describe network verification models in high level scripts in exactly the same way that they would describe human activities. These descriptions use phonemes recorded on the sensor nodes to verify if the network is operating according to the design specifications. The type of phonemes used can vary among different networks and applications but, in general, it can be classified into two broad categories: *data* and *metadata*. *Data* corresponds to the actual phonemes used by the application stack for the detection of activity patterns. *Metadata* corresponds to the phonemes specifically recorded to be used for system verification purposes. In practice, users are asked to provide a syntax over a finite set of phonemes (*data*, *metadata* or both) that will determine when the behavior of the network is not consistent with its design specification. This methodology is analogous to Natural Language Processing where a grammar is defined over a set of phonemes to determine what a valid word, sentence and paragraph is. Depending on the type of phonemes used, system verification at different levels of granularity can be achieved.

As Figure 16 shows, the data provided by the sensor network is forwarded to the application stack where it is mapped to application outputs. At the same time, both *data* and *metadata* are processed by the system verification stack, running in parallel with the actual application stack, to detect system design violations. At the next level, the system design violations are mapped to the actual application outputs to identify and remove possible false positives/negatives and the outcome of this step becomes the actual system output. It is important to note here that the user-defined system verification models run in parallel with the actual application (Figure 16) and therefore operate on the same real data. This gives the ability to the network to associate time information to the detected faults and correlate them with the actual output of the system in an on-line manner and after the sensor network has been deployed. Note, that from the system perspective, there is no difference between system verification and activity recognition grammars. The only difference is the logical interpretation of these grammars. In addition, both *data* and *metadata* are nothing more than a finite set of pre-defined symbols, the phonemes of the system.

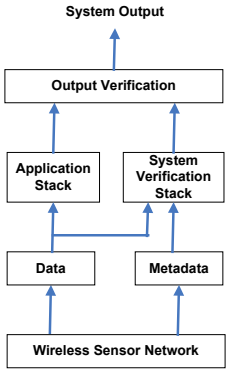


Fig. 16. System Verification Overview.

A simplified system verification grammar (only a fraction of the phonemes is used) for the deployment described in Section IV is shown in Figure 17. This grammar detects errors by checking for invalid sequences of phonemes produced by the sensor network. Initially, the different phonemes, representing different areas of interest in the house, are grouped into higher level phonemes that represent different rooms. This is done by using the $Time\_Replace$ parameter of the time abstraction layer. The actual grammar checks if the input sequence of the visited rooms is valid. For instance, given the floorplan shown in Figure 1(a), it is impossible to move from the kitchen to the second floor, where the bedroom is, without first passing through the living room. In that way, several areas in the house can be defined as "checkpoints" for monitoring the correct operation of the network. The produced errors can be easily mapped to the activity semantics produced by the system since all semantics are mapped to ground truth time. This is be shown in more detail in the next section.

IX. Activity Summaries

Driven by the assisted living application example and to demonstrate some of BScope's capabilities, we have designed a set of grammar hierarchies for automatically generating meaningful activity summaries. In particular, the following grammar hierarchies have been defined:

Bedroom Activity: A single level grammar hierarchy for detecting sleeping activity.

Bedroom Duration Activity: A two-level grammar hierarchy for categorizing the sleeping activity into short (less than 8 hours) and long (more than 8 hours) based on its duration.

Breakfast Activity: A three-level grammar hierarchy for detecting breakfast activity. At the first level, sleeping activity and meal preparation is detected. At the second level, the detected sleeping and meal activities are used to identify possible breakfast activity. At the last level, the time abstraction layer is used to filter out false positives.

Bathroom Activity: A single level grammar for monitoring bathroom usage.

Bathroom Activity: A two-level grammar for classifying bathroom usage into short (less than 15 minutes) and long (more than 15 minutes) based on its time duration.

Input: Any sequence of the phonemes:$\{Bedroom, Bath, Couch1, Couch2, Stove, KitchenTable, Refrigerator, \ldots\}$

Output: A sequence of any of the following non-terminal symbols: $\{Normal, Error\}$

1. $Time\_Rule: Bedroom > 0$	$\rightarrow$	$Upstairs$
2. $Time\_Rule: Bath > 0$	$\rightarrow$	$Upstairs$
3. $Time\_Rule: Couch1 > 0$	$\rightarrow$	$LivingRoom$
4. $Time\_Rule: Couch2 > 0$	$\rightarrow$	$LivingRoom$
5. $Time\_Rule: Stove > 0$	$\rightarrow$	$Kitchen$
6. $Time\_Rule: KitchenTable > 0$	$\rightarrow$	$Kitchen$
7. $Time\_Rule: Refrigerator > 0$	$\rightarrow$	$Kitchen$
8. $Use\_Time\_Phonemes\_Only$	$\rightarrow$	YES
9. $repeater$	$\rightarrow$	YES
10. V_N	$=$	$\{Start, Normal, Error\}$
11. V_T	$=$	$\{Bedroom, Bath, Couch1, Couch2, Stove, KitchenTable, Refrigerator, \ldots\}$
12. $Start$	$\rightarrow$	$Normal \mid Error$
13. $Normal$	$\rightarrow$	$Upstairs\ LivingRoom \mid LivingRoom\ Upstairs \mid LivingRoom\ Kitchen \mid Kitchen\ LivingRoom$
14. $Error$	$\rightarrow$	$Upstairs\ Kitchen \mid Kitchen\ Upstairs$

Fig. 17. Grammar for detecting system faults.

Floor Activity: A single level grammar hierarchy for monitoring the floor at which the person inside the house is located.

System Fault Activity: A singe level grammar for identifying faults in the network by checking for invalid sequence of phonemes.

Note that the different grammars used in this section are more context-free grammars (CFGs) in the sense that they do not take advantage of the probabilistic parsing that PCFGs offer. We use these grammars to demonstrate the ability of the system to process the incoming streams of sensing data at run-time while taking into account both their spatial and temporal characteristics. However, an elaborate example that demonstrates the probabilistic power of PCFGs on real data recorded by the same home sensor network deployment used in this paper can be found in [24].

The BScope system was configured to run the above grammars at run-time on the stream of phonemes produced by the house camera sensor network deployment described in Section IV. The network was deployed for 25 days over which approximately 1.2 million packets were transmitted from the camera nodes to the basestation. From these, over 200 thousand packets carried more than 1.3 million location measurements that were directly converted to approximately 444 thousand phonemes.

Figure 18 shows the run-time output of the first five grammar hierarchies on a 25-day dataset acquired using our pilot camera sensor network deployment(Figure 1). All the detected activities are represented as waveforms where the width of each pulse denotes the actual duration of the activity. The bedroom grammar was able to detect

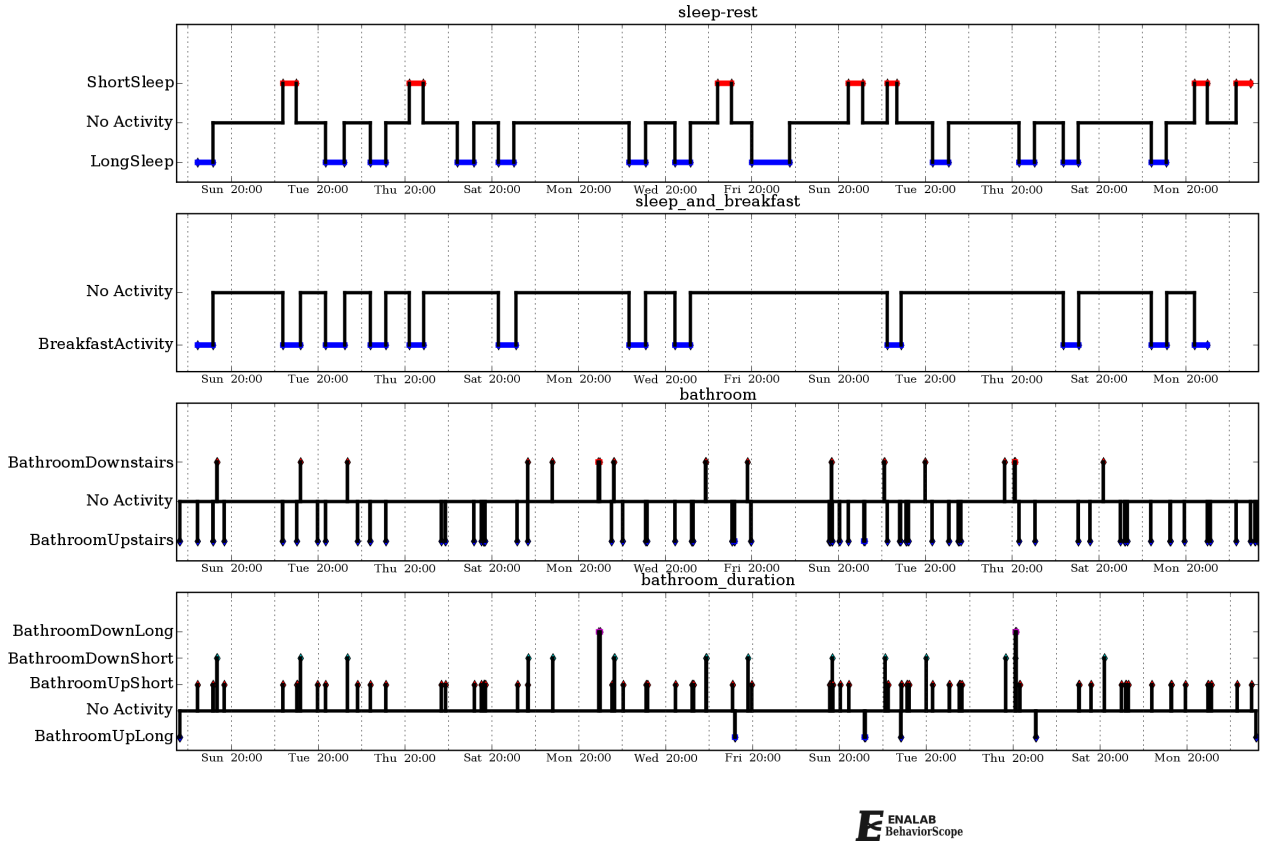


Fig. 18. Activity Summaries. Note, that the duration of breakfast activity seems to be similar to the duration of the corresponding sleeping activity. This is due to the fact that the breakfast activity is defined as a time-constrained sequence of sleeping and meal activities. As a result of this, the duration of the breakfast activity becomes equal to the sum of the duration of the sleeping and meal activities that is composed from.

20 different sleeping activities in the course of the 25 days. In 2 cases the monitored person spent his night into another house and therefore we correctly did not identify any sleeping activity. For the rest 3 nights we were unable to detect the sleeping activity due to a malfunction of the node monitoring the bedrooms. For 12 nights the person slept at least 8 hours and for the rest 8 nights he slept between 6 and 7 hours. He had breakfast only 8 times during this period and he was visiting the bathroom an average of 6 to 7 times a day (these visits include any type of bathroom activity such as toilet, shower etc.). Among these bathroom visits, there was always a visit right before and after every sleeping activity.

Figure 19 shows the results of running the floor and fault activity grammars on a smaller time window[1]. The floor grammar is consistent with the bedroom grammar in the sense that for every sleeping activity the person was at the second floor where the bedrooms are located. However, it is obvious that in some cases there is unusually dense floor activity. This dense activity is caused by false positives produced by the sensor nodes when sudden

[1]We do not provide the results for these grammars over the 25-day dataset because the plots are very dense and thus very difficult to read.

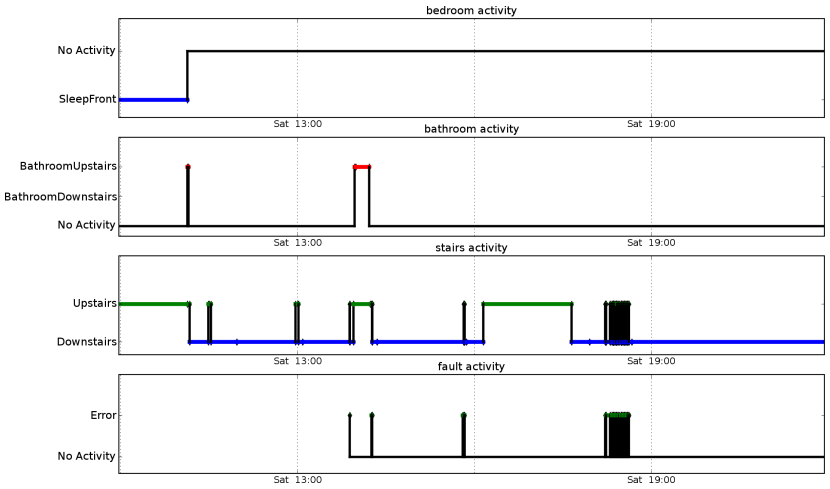


Fig. 19. Typical activity after waking up.

changes of lighting take place (i.e. lights are turned on or off). In this case phonemes are continuously recorded on both floors resulting into a large number of transitions between the two floors. This conclusion is verified by the fault detection grammar results. It is obvious that most of the dense floor activity is classified as erroneous (invalid phoneme sequences). Note that sudden bursts or node false positives that cause instantaneous floor transitions are classified as erroneous.

Overall, using seven grammar hierarchies we were able to automatically translate over 1.3 million location measurements into a few hundreds of high level activity semantics.

X. Conclusion

In this paper we have presented the BScope architecture and its ability to classify behaviors using distributed sensors. Our deployment has demonstrated its use in a practical real life application to simultaneously classify human activities while also conducting consistency checks on the collected data to verify that the system worked according to our specifications. The embedded consistency check mechanisms can now inform us about node failures, poor connectivity, and transient errors that are hard to spot in the collected datasets. The versatile programming interface and support software allows quick customization of the system into different house layouts and a diverse set of applications. The proposed timing abstraction was operated as expected, but during the process we found that grammar authoring required some expertise on the programmer side. Nonetheless, our implementation achieved its goal of shifting the programming effort from embedded systems and network programming to higher level grammar programming thus shifting the expertise requirements to each particular domain. The completed system described here is currently under deployment in multiple homes, and soon will be deployed in other applications different from assisted living. Our future research plan is to develop mechanisms for automatically mapping grammar hierarchies on the network topology as well as mechanisms with which nodes will be able to autonomously decide what phonemes they should be generating according to a generic specification.

REFERENCES

[1] T. Teixeira, D. Lymberopoulos, E. Culurciello, Y. Aloimonos, and A. Savvides, "A lightweight camera sensor network operating on symbolic information," in *Proceedings of the First Workshop on Distributed Smart Cameras, held in conjunction with ACM SenSys 2006*, April 2006.

[2] T. Teixeira and A. Savvides, "Lightweight people counting and localizing in indoor spaces using camera sensor nodes," in *ACM/IEEE International Conference on Distributed Smart Cameras*, September 2007.

[3] K. Whitehouse, J. Liu, and F. Zhao, "Semantic streams: A framework for the composable semantic interpretation of sensor data," in *Proceedings of European Workshop on Sensor Netowrks (EWSN)*, February 2006.

[4] S. S. Intille, K. Larson, and E. M. Tapia, "Designing and evaluating technology for independent aging in home," in *International Conference oon Aging, Disability and Independence*, 2003.

[5] M. Philipose, K. P. Fishkin, M. Perkowitz, D. J. Patterson, D. Fox, H. Kautz, and D. Hahnel, "Inferring activities from interactions with objects," *IEEE Pervasive Computing*, vol. 03, no. 4, pp. 50–57, 2004.

[6] E. M. Tapia, S. S. Intille, and K. Larson, "Activity recognition in the home setting using simple and ubiquitous sensors," in *IEEE Pervasice Computing*, 2004.

[7] D. J. Patterson, D. Fox, H. Kautz, and M. Philipose, "Fine-grained activity recognition by aggregating abstract object usage," in *IEEE International Symposium on Wearable Computers*, October 2005.

[8] L. Liao, D. Fox, and H. Kautz, "Location-based activity recognition using relational markov models," in *Nineteenth International Joint Conference on Artificial Intelligence*, 2005.

[9] A. Hauptmann, J. Gao, R. Yang, Y. Qi, J. Yang, and H. Wactar, "Automated analysis of nursing home observations," *IEEE Pervasive Computing 3.2: 15-21*, 2004.

[10] Y. A. Ivanov and A. F. Bobick, "Recognition of visual activities and interactions by stochastic parsing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 852–872, 2000.

[11] A. S. Ogale, A. Karapurkar, and Y. Aloimonos, "View-invariant modeling and recognition of human actions using grammars," *IEEE International Conference on Computer Vision (ICCV)*, October 2005.

[12] N. M. Oliver, B. Rosario, and A. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 831–843, 2000. [Online]. Available: citeseer.ist.psu.edu/oliver99bayesian.html

[13] N. Moenne-Loccoz, F. Bremond, and M. Thonnat, "Recurrent bayesian network for the recognition of human behaviors from video," in *Third International Conference on Computer Vision Systems*, April 2003, pp. 68–77.

[14] D. Moore and I. Essa, "Recognizing multitasked activities from video using stochastic context-free grammar," in *Proceedings of AAAI*, 2002.

[15] S. Park and J. K. Aggarwal, "Event semantics in two-person interactions," in *Proceedings of International Conference on Pattern Recognition (ICPR)*, August 2004.

[16] D. Wyatt, M. Philipose, and T. Choudhury, "Unsupervised activity recognition using automatically mined common sense," in *Proceedings of AAAI*, July 2005.

[17] S. Rao and P. Sastry, "Abnormal activity detection in video sequences using learnt probability densities," in *TENCON*, October 2003.

[18] S. Geman and M. Johnson, "Probabilistic grammars and their applications," in *International Encyclopedia of the Social & Behavioral Sciences. N.J. Smelser and P.B. Baltes, eds., Pergamon, Oxford, 12075-12082*, 2002.

[19] R. C. Gonzalez and M. G. Thomason, *Syntactic Pattern Recognition: An Introduction*. Addison-Wesley, 1978.

[20] K. S. Fu, *Syntactic Methods in Pattern Recognition*. Volume 112, Mathematics in Secience and Engineering, 1974.

[21] K. S. Fu, *Syntactic Pattern Recognition, Applications*. Springer-Verlag, 1977.

[22] D. Lymberopoulos, A. Ogale, A. Savvides, and Y. Aloimonos, "A sensory grammar for inferring behaviors in sensor networks," in *Proceedings of Information Processing in Sensor Networks (IPSN)*, April 2006.

[23] C. S. Wetherell, "Probabilistic languages: A review and some open questions," *ACM Comput. Surv.*, vol. 12, no. 4, pp. 361–379, 1980.

[24] D. Lymberopoulos, T. Teixeira, and A. Savvides, "Detecting patterns for assisted living using sensor networks," in *Proceedings of SensorComm*, October 2007.

[25] L. Nachman. Intel Corporation Research Santa Clara. CA, "New tinyos platforms panel:iMote2," in *The Second International TinyOS Technology Exchange*, Feb 2005.

[26] C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava, "A dynamic operating system for sensor nodes," in *Proceedings of the Third International Conference on Mobile Systems, Applications, And Services (Mobisys)*, 2005.

[27] D. Lymberopoulos, A. Bamis, and A. Savvides, "Extracting temporal characteristics from sensor data streams," Yale University, Tech. Rep. ENALAB 041408, 2008.

Dimitrios Lymberopoulos is a Ph.D. candidate in the Electrical Engineering Department at Yale University and a member of the Embedded Networks and Applications Lab (ENALAB). He graduated from the Computer Engineering and Informatics Department at University of Patras, Greece (2003) and he holds an M.S. in Electrical and Computer Engineering from Yale University (2005). His research is focusing on self-configuring sensor networks, particularly the ones that can be used to automatically interpret human behavior in order to provide assistive services. Dimitrios' research is supported by a Microsoft Graduate Student Research Fellowship.

Thiago Teixeira is currently a Ph.D. Candidate at Yale University. He attained B.S. degree in Electrical Engineering (2003), B.A. in Mathematics (2003) and M.S. in Engineering (2005) from the Johns Hopkins University, where he developed an address-event imaging sensor network. In 2005, he joined the Yale Embedded Networks and Applications Lab (ENALAB) where he has been since. His latest work is in lightweight algorithms for human detection and tracking in indoors camera networks.

Andreas Savvides is an Assistant Professor in the Electrical Engineering and Computer Science Departments at Yale University. He is the founder of the Embedded Networks and Applications Lab (ENALAB) that specializes in the design and implementation of distributed sensor networks and smart spaces. Dr. Savvides completed his Ph.D. in the Electrical Engineering Department at UCLA in 2003. Before this he earned his B.S in Computer Engineering from the University of California, San Diego and an M.S in Electrical and Computer Engineering from UMASS, Amherst. Dr. Savvides' research is supported by an NSF CAREER award as well as other federal grants and industrial support. His current research is focusing on self-configuring sensor networks, particularly the ones that can be used to interpret human behavior in order to provide assistive services.