# BScope: A Scalable, Run-Time Architecture for Activity Recognition Using Wireless Sensor Networks

Dimitrios Lymberopoulos, Thiago Teixeira, Andreas Savvides Embedded Networks and Applications Lab (ENALAB) Yale University New Haven, CT, USA

{dimitrios.lymberopoulos,thiago.teixeira,andreas.savvides}@yale.edu

#### Abstract

This paper presents a new system for interpreting human activity patterns using a sensor network. Our system 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 in 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. 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.

#### **1** 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 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 a run-time architecture that processes streams of timestamped sensor data along with prior context information to infer activities and generate appropriate notifications. In our system, 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 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 demostrates 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 ap-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

proach biases the required programming effort towards domain expertise rather than embedded systems, networking and database programming.

Our presentation is organized as follows: in the next section we motivate the choice of the inference engine and we provide some background information. Section 3 presents the methodology for converting a sensor network into an interpretation mechanism and shows the main idea of the proposed system architecture. Section 4 provides an overview of the related work and it highlights the main contributions of our work. In Section 5 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 6 describes in detail the architecture through illustrative examples based on an assisted living scenario. Section 7 presents the evaluation of the system on a 25-day dataset acquired from a real home deployment and Section 9 concludes the paper.

# 2 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 [13]: (1) the decisiontheoretic 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. 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 decisiontheoretic 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 2.2, 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 [13]. In practice, this means that the recognition power and training capabilities of both tools is the same [9]. However, PCFGs provide a much more expressive interface that is closer to the human way of thinking 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 capabilitites 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.

#### 2.1 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 [reference withheld]. A scheme for distributing and mapping grammar hierarchies on the network has already been devised and will be presented in a future publication.

# 2.2 Probabilistic Context-Free Grammars (PCFGs)

A probabilistic context-free grammar *G* [18, 9] 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.
- *Start*  $\in$  *V*<sub>N</sub> is the start symbol.



Figure 1. A typical bathroom layout.

- *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*<sub>*ij*</sub>} 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.  $0 < p_{ij} \le 1, \forall i, j$
  - 3. For every *i* with  $1 \le i \le |V_N|$ :  $\sum_{1 \le j \le n_i} p_{ij} = 1$ , where  $n_i$  is the number of productions with the ith 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:  $A \rightarrow a^{(0 \le p \le 1)}$ , 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  $\alpha$  derives string  $\beta$  ( $\alpha \Rightarrow \beta$ ) if there is a sequence:  $\alpha = \alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_n = \beta, n \ge 0$ , 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 probabilities that were used to generate this sentence. If the same sentence can be derived in more than one ways then the derivation with the highest probability wins. For instance, given the following PCFG:

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

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

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

<b>Input:</b> Any sequence of the phonemes: { <i>TO</i> , <i>SI</i> , <i>TW</i> }
<b>Output:</b> A sequence of any of the following non-terminal
<pre>symbols: {NormalBathVisit,IncompleteBathVisit}</pre>

<b>1.</b> <i>V</i> <sub><i>N</i></sub>	=	{Start,NormalBathVisit, IncompleteBathVisit}
<b>2.</b> <i>V</i> <sub>T</sub>	=	$\{TO, SI, TW\}$
3. Start	$\rightarrow$	NormalBathVisit IncompleteBathVisit
4. NormalBathVisit	$\rightarrow$	TO SI TW   TO SI
5. IncompleteBathVisit	$\rightarrow$	ТО

#### Figure 2. A grammar for identifying bathroom visits.

(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, cuncurrently 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 [reference withheld].

# **3** The Sensor Network As An Interpretation Engine

#### 3.1 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, this set of discrete symbols corresponds 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 1, 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 2. 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 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 dinning table, the refrigerator or the sink inside the kitchen. These phonemes could be used in many different ways to describe basic activities that take place inside a house.

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 PIR 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.* 

#### **3.2** System Process Overview

Figure 3 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



Figure 3. Data interpretation process

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.

#### **4** Related Work

As in the Semantic Streams work presented in [6] 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 [11, 8, 5, 4, 7]. 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 [14]. 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 [15], identify human gestures [2], recognize high-level narratives of multi-player games and player's strategy [16], and to classify different types of interactions(dancing, hugging etc) between two people [17]. 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 [12, 10]. 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.

# 4.1 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*. 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 trainining capabilities as HMMs or similar pattern recognition tools [9, 13].

#### 5 Considering Space and Time

According to the description provided in Section 3.1, 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 diferent 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.

#### 5.1 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 4 shows a high-level overview of the timing abstraction mechanism in the BScope system. Area phonemes



Figure 4. Overview of the time abstraction layer.

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 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 independetly of its position in the hierarchy. In that way, time information is hierarchicaly 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 2. 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 5 and Figures 6 provide two different alternatives.

In Figure 5, 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 5

<b>Input:</b> Any sequence of the phonemes:{ <i>TO</i> , <i>SI</i> , <i>TW</i> }
<b>Output:</b> A sequence of any of the following non-terminal
symbols: { <i>NormalBathVisit</i> , <i>IncompleteBathVisit</i> }

		1 1 1 1 1 1
7. IncompleteBathVisit	$\rightarrow$	TO_LONG
		TO_LONG SI
6. NormalBathVisit	$\rightarrow$	TO_LONG SI TW
		In complete Bath Visit
5. Start	$\rightarrow$	NormalBathVisit
<b>4.</b> <i>V</i> <sub><i>T</i></sub>	=	$\{TO\_LONG, SI, TW\}$
		IncompleteBathVisit}
<b>3.</b> $V_N$	=	$\{Start, Normal BathVisit,$
2. Use_Time_Phonemes_Only: YES		
<b>1.</b> <i>Time_Rule</i> : <i>TO</i> > 900	$\rightarrow$	TO_LONG

Figure 5. 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 ·			
NormalBathVis	sit > 900	$\rightarrow$	Normal
2. Time_Rule :			
IncompleteBat	hVisit > 900	$\rightarrow$	Incomplete
3. Use_Time_Phone	mes_Only :		YES
<b>4.</b> $V_N$	=	{Start	,
		Norma	lLongBathVisit,
		Incom	pleteLongBathVisit}
<b>5.</b> $V_T$	=	{Norm	ualLongBathVisit,
		Incom	pleteLongBathVisit}
6. Start	$\rightarrow$	{Norn	ualLongBathVisit
		Incom	pleteLongBathVisit}
7. NormalLongBath	Visit		
$\rightarrow$		Norma	ıl
8. IncompleteLongE	BathVisit		
	$\rightarrow$	Incom	plete

Figure 6. Using a second level grammar for identifying abnormally long bathroom visits.

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 5 will include only those toilet visits that are abnormally long.

Another way to achieve the same functionality is shown in Figure 6. In this case, an additional grammar is created that receives as input the output of the grammar shown in Figure 2. 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.

# 6 **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 hierachies and trigger notifications on the detection of semantics of interest.

This process is shown in detail in Figure 7. 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 redundand occurences of phonemes. At the same time, the initial input sequence of phonemes is broken into multiple grammar-specific phoneme sequences. The grammarspecific 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 trigerring 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.

#### 6.1 BScope Configuration

Programmers can configure the BScope interpretation engine to preprocess the input phoneme streams on a pergrammar 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 these parameters is to give the ability to the programmer to configure the BScope engine according to the needs of his grammar



Figure 7. Overview of the BScope's interpretation engine.

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 5 and the stairs activity grammar shown in Figure 9. The configuration files for both grammars can be seen in Figures 8 and 10 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 instance if we wouldn't like # 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

Figure 8. The configuration file for the bathroom monitoring grammar.

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 intial 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 grammarspecific 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. Everv "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 or more

<b>Input:</b> Any sequence of the phonemes: { <i>DOWN</i> , <i>UP</i> }
<b>Output:</b> A sequence of any of the following non-terminal
symbols: { <i>MoveUp</i> , <i>MoveDown</i> }

<b>1.</b> $V_N$	=	{Start,MoveUp,MoveDown}
2. V <sub>T</sub>	=	$\{DOWN, UP\}$
3. Start	$\rightarrow$	MoveU p MoveDown
4. MoveUp	$\rightarrow$	DOWNUP
5. MoveDown	$\rightarrow$	UPDOWN

Figure 9. 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
10 70	

Figure 10. The configuration file for the stairs grammar.

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 5, 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 respresented 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.

#### 6.2 **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 chunking process on a pergrammar basis. In particular, another parameter called "Execute\_Condition" is introduced. Every Execute\_Condition parameter consists 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 9), the input is considered to



Figure 11. Overview of the Bscope run-time architecture.

be valid when it contains at least one *DOWN* and one *UP* phonemes:

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

Figure 11 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.

#### 6.3 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 2 shows. All the configuration files and low level details are automatically taken care of by the system. Figure 12 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



Figure 13. Sensor node software stack.

cell-phone text message is sent including information about the specific semantic that was detected and its duration. Figure 12, 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.

# 7 Asssisted Living Deployment

To evaluate the BScope system in a real environment we created a pilot camera sensor network deployment in a twofloor house. Five Intel iMote2 [1] 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 14. The camera nodes are attached to the ceiling with the camera lens pointing down into the room. Each camera node (Figure 13) features an OV7649 VGA camera from Omnivision coupled to a 162 degree lens. The PXA271 processor aboard the iMote2 runs the SOS operating system from UCLA [3] in which we have implemented the corresponding camera drivers, a small image processing library and the software required by the BScope system(Figure 13). This camera node acquires images at 8 frames per second, downsamples them to a  $80 \times 60$  resolution and uses a lightweight image processing algorithm to extract the location of a person inside the house.

To ensure that the ordering of measurements from different nodes is preserved, we have also 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. 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

		BScope							X
Control Center Activity Graph									
Database Settings Connection Info	Table	Grammar Settings Grammar Execution		System Activity Debug Info	cenng and re	pearing uone:			
(and they	nov30_centroidx	bedroom		Failed to parse. No o	un. Execution output was pr	Conditions were not m oduced!	et		
Clarr Default Set As Defa	Password	bathroom	bathroom ACTUAL INPUT TO THE GRAMMAR: ((TR, T, 'S, TR, 'T, 'S, 'T, 'S', 'R', 'T), ((1165241185L, 1165241193L), (116524 (1392L, 1165241193L), (1165241193L, 1165241198L), (1165241198L), 11652						
Time Window Use Time Window Hour Month	Day			41199L), (1165241 5241210L, 116524 65241217L), (1165 Grammar specific fil Grammar was not ru Failed to parse. No M NO MORE PHONEME	199L, 11652 1210L), (116 241217L, 11 tering and re un. Execution output was pr S TO PROCES	41205L), (116524120 5241210L, 11652412 65246047L)]) peating done! Conditions were not m oduced! S!	5L, 11652+ L4L), (116 et	41210L), (11 5241214L, 1	6 1
×	×			System Output	Hierarchy	Activity	From	To	
				02:02:51 04/14/07	bathroom	BathroomDownstairs	21:04:49	21:06:11	
To: Hour Month V	Day Vear	Add Remove Run Pause Stop Notifications Hierarchy E-mail Cell # bedroom	Carrier Cingular	02:02:51 04/14/07 02:02:51 04/14/07 02:02:52 04/14/07 02:02:52 04/14/07 02:02:56 04/14/07 02:02:56 04/14/07	bathroom bedroom bathroom bathroom bathroom	BathroomUpstairs SleepFront BathroomUpstairs BathroomUpstairs BathroomDownstairs BathroomUpstairs	23:01:25 23:06:14 06:38:31 06:40:17 23:25:29 23:27:38	23:04:30 06:38:31 06:40:01 06:55:14 23:26:46 23:28:22	
Add Load				02:02:57 04/14/07 02:02:57 04/14/07	bathroom	SleepFront	23:28:51 01:17:36	23:31:05	
Description From To				Actions Taken					
				Time 02:02:51 04/14/07 02:02:57 04/14/07 02:03:11 04/14/07 02:03:30 04/14/07	Hierarchy bedroom bedroom bedroom bedroom	Activity Persons SleepFront SleepFront SleepFront SleepFront	Contacted	111/0011 111/0011 111/0011	
🏷 📼 🚵 Clear Remove Save	Ead	Add Remove							

Figure 12. The main control panel of the Bscope System. Some fields have been blurred to preserve author's anonymity.

recorded at the node. These are collected at the basestation and compared against the statistics observed at the basestation to determine where packet losses occur in the system.

# 7.1 Network Statistics

Over the course of 25 days, approximately 1.2 million packets were transmitted from the camera nodes to the base. Over 200 thousand packets carried more than 1.3 million location measurements that were directly converted to approximately 444 thousand phonemes.

From the data transmitted in the heartbeat packets, Figures 16 and 17 can be computed to show a diagnostic of the network over the duration of the experiment. Figure 16 shows the number of dropped centroid packets for each node. Drop rates increase with the distance from the base for all nodes except node 4, which showed the worst drop-rate despite being the closest to the base. It is also worth noticing that the centroid packets are dropped in sudden bursts, and that these bursts usually occur simultaneously on multiple nodes. This correlation is more obvious on nodes that are geographically close, such as nodes 4 and 10. These observations point to the conclusion that centroid packets are dropped when events span multiple nodes at the same time, causing many packet collisions. This is more intense on the nodes near the base, as expected.

Meanwhile, Figure 17 paints a different picture: heartbeat drop rates are not bursty like centroid drop rates, but rather smooth instead. Moreover, nodes 10, 12 and 15 exhibit similar behavior, dropping around 4000 packets each. Node 4 experiences a large spike in dropped heartbeats between 03-14 and 03-15, but afterward follow a pattern akin to that of node 6. There were close to 23 thousand dropped centroid packets making up 10.2% of all attempted centroid transmissions. Since heartbeats are only sent when there is no centroid activity, and since they are transmitted in relatively sparse intervals, the smaller number of dropped heartbeats (around 16.5 thousand, or 1.56% of all attempted transmis-

sions) seems to agree with the analysis that the major culprit of packet drops is packet collisions. However, even under the presence of these issues, we now show how the BScope system can automatically recognize faulty operation.

# 7.2 Grammar-Based System Consistency Checking

While according to Figures 12 and 15 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



(a)House deployment floorplan

(**b**) Nodes 4, 6 deployed

Figure 14. Pilot network deployment overview.

(c) The field of view for nodes 4(bottom) and 10(top)



Figure 15. Various phonemes over a four day period



Figure 16. Dropped centroid packets over the period of 25 days for all 5 nodes.

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 18 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 *meta*-



# Figure 17. Dropped heartbeat packets over the period of 25 days for all 5 nodes.

*data* 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 18) 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 activ-



Figure 18. System Verification Overview.

ity 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.

A simplified system verification grammar (only a fraction of the phonemes is used) for the deployment described in Section 7 is shown in Figure 19. 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 14(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.

# 8 Activity Summaries

Driven by the assisted living application example, 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*, *Couch*1, *Couch*2, *Stove*, *KitchenTable*, *Refrigerator*, ...} Output: A sequence of any of the following non-terminal symbols: {*Normal*, *Error*}

<b>1.</b> $Time_Rule : Bedroom > 0$	$\rightarrow$	Upstairs
<b>2.</b> <i>Time_Rule</i> : <i>Bath</i> > 0	$\rightarrow$	Upstairs
<b>3.</b> <i>Time_Rule</i> : $Couch1 > 0$	$\rightarrow$	LivingRoom
<b>4.</b> Time_Rule : Couch $2 > 0$	$\rightarrow$	LivingRoom
<b>5.</b> Time_Rule : Stove $> 0$	$\rightarrow$	Kitchen
<b>6.</b> <i>Time_Rule</i> : <i>KitchenTable</i> $> 0$	$\rightarrow$	Kitchen
<b>7.</b> Time_Rule : Refrigerator $> 0$	$\rightarrow$	Kitchen
8. Use_Time_Phonemes_Only: YES		
9. repeater : YES		
<b>10.</b> $\dot{V}_N$	=	{Start,Normal,Error
11. $V_T$	=	Bedroom, Bath, Couch1,
		Couch2, Stove,
		KitchenTable,
		Refrigerator,}
<b>12.</b> <i>Start</i>	$\rightarrow$	Normal Error
13. Normal	$\rightarrow$	Upstairs LivingRoom
		LivingRoom Upstairs
		LivingRoom Kitchen
		Kitchen LivingRoom
14. Error	$\rightarrow$	Upstairs Kitchen
		Kitchen Upstairs
		-

Figure 19. 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.

Figure 20 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 14). 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 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 21 shows the results of running the floor and fault activity grammars on a smaller time window<sup>1</sup>. 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 changes of lighting take place (i.e. lights are turned on or off). In this case

<sup>&</sup>lt;sup>1</sup>We do not provide the results for these grammars over the 25day dataset because the plots are very dense and thus very difficult to read.



Figure 21. Typical activity after waking up.

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.

#### **9** 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 reallife 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.

# **10** References

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

- [2] A.S. Ogale et. al. View-invariant modeling and recognition of human actions using grammars. *ICCV'05*, October 2005.
- [3] Chih-Chieh Han et. al. A dynamic operating system for sensor nodes. In *Proceedings of (Mobisys)*, 2005.
- [4] D. Patterson et. al. Fine-grained activity recognition by aggregating abstract object usage. In *IEEE International Symposium on Wearable Computers*, October 2005.
- [5] E. M. Tapia et. al. Activity recognition in the home setting using simple and ubiquitous sensors. In *PER-VASIVE 2004*, 2004.
- [6] K. Whitehouse et. al. Semantic streams: A framework for the composable semantic interpretation of sensor data. In *Proceedings of EWSN*, February 2006.
- [7] L. Liao et. al. Location-based activity recognition using relational markov models. In *Nineteenth International Joint Conference on Artificial Intelligence*, 2005.
- [8] M. Philipose et. al. Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 03(4):50–57, 2004.
- [9] S. Geman et. al. 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.
- [10] S. Rao et. al. Abnormal activity detection in video sequences using learnt probability densities. In *TEN-CON*, October 2003.
- [11] S.S. Intille et. al. Designing and evaluating technology for independent aging in home. In *International Conference oon Aging, Disability and Independence*, 2003.
- [12] W. Pentney et. al. Unsupervised activity recognition using automatically mined common sense. In *Proceed*ings of AAAI, July 2005.
- [13] R. C. Gonzalez and M. G. Thomason. Syntactic Pattern Recognition: An Introduction. Addison-Wesley, 1978.
- [14] 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.
- [15] Y. A. Ivanov and A. F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):852– 872, 2000.
- [16] D. Moore and I. Essa. Recognizing multitasked activities from video using stochastic context-free grammar. pages 770–776, Menlo Park, CA, USA, 2002. AAI.
- [17] S. Park and J. K. Aggarwal. Event semantics in twoperson interactions. August 2004.
- [18] C. S. Wetherell. Probabilistic languages: A review and some open questions. ACM Comput. Surv., 12(4):361– 379, 1980.



Figure 20. 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.