An Easy-to-Program Sensor System for Parsing Out Human Activities

Dimitrios Lymberopoulos, Andrew Barton-Sweeney, Thiago Teixeira and Andreas Savvides

Embedded Networks and Applications Lab, ENALAB
Yale Univerisity, New Haven, CT 06520, USA

Abstract. We present a new system for interpreting human activity patterns using a sensor network. The paper takes an illustrative approach that presents our methodology using an example, cooking activity detection, in an assisted living application. The activity is detected using a hierarchical probabilistic grammar formulation that is able to detect the activity across multiple instances by reasoning on location information and the kitchen layout map. In our system node level code authoring is automated by a middleware infrastructure that provides a layer of abstraction between the programmer and the sensor nodes. Short descriptions of grammars and rules in the form of scripts are converted by the middleware to elaborate node-level code. In our cooking detection implementation, 18 lines of grammar definition are automatically transformed into 2600 lines of node level C code that morphs the sensor node into a cooking sensor with binary outputs.

1 Introduction

Understanding and acting on the data at the sensor node level would be of immense value to numerous aspects of everyday life. If a small sensor network deployed at home could understand the inhabitant's behaviors it could provide numerous services. It would be able to guard against unsafe situations, provide warnings, alarms and manage actuation in ways that would greatly improve the quality of life of elders living alone around the globe.

To investigate this direction in this paper we present a new methodology for understanding behaviors from low level sensor data using a hierarchy of probabilistic grammars. Our system uses a new imager sensor network to track a person inside a home without tagging the person with a sensor. The collected locations are then correlated with the building map to derive a string of symbols that is processed by a hierarchy of probabilistic grammars to *parse out* specific behaviors. We illustrate this process with the detection of cooking activity with a single sensor and argue that similar automated classification of activity will enable a new generation of applications in assisted living and aging in place. Our system design is structured to act as an extensible rule-based system classify daily activities using a set of rules and grammar definitions. Using this system

one can easily describe various activities and unsafe behaviors that can be used to provide warnings alarms and other services.

Our work makes two main contributions. First, it demonstrates the application of a grammar-based formulation to detect complex activities (e.g cooking) from simple sensor measurements (locations). Our experimental results presented here demonstrate that this *sensory grammar* driven approach can invariably detect activity across many possible instances and actors. We illustrate this by detecting cooking activity across different dishes and cooks. Our second contribution is the design and implementation of a lightweight, privacy preserving imager sensor network and its supporting middleware infrastructure. The middleware provides a powerful programming abstraction that greatly simplifies network programming and re-tasking. Through this abstraction, network operator can task the nodes by writing short descriptions of rules and behaviors in the form of simple scripts. The scripts define the grammar and rule specifications that are automatically translated to node-level code generated by the middleware. Code is generated in the form of modules that can be dynamically re-wired at runtime. This approach simplifies the programming effort by significantly reducing programming and debugging time. We demonstrate the power of the proposed system by recognizing a complex activity, cooking, with a single sensor node.

2 Related Work

As in the Semantic Streams work presented in [7, 16] the system presented here tries to generate 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 strings that is not application specific. 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, 14, 15, 13, 6]. This approach requires a person to wear an RFID reader and extensive tagging of objects with RFID tags. Other researchers, attempted similar monitoring with video cameras [3]. The approach we propose poses several differences to previous systems. When compared to RFID approaches, our solution does not require object tagging, only a few sensor nodes (1-2 per room). The grammar hierarchy however, could also be applied to deployments that have RFIDs. The power for the grammar hierarchy arises from its ability make high level inferences from low-level sensor measurements, that is, simple sensor measurements that cannot provide direct information about an activity (as opposed to high level sensing that involves the tagging of every single item, i.e subject is touching the tea pot and tea cups, therefore subject is making tea). Unlike vision-based approaches, our approach does not rely on vision alone. Instead, relies on the detection of patterns using multimodal sensing and grammars, and only borrows very specific information from vision that would be useful to the task. This approach will allow us to bypass hard and computationally demanding vision problems

with a lightweight sensor network. Probabilistic grammars have been previously successfully applied by computer vision researchers to detect persons that are picked-up or dropped-off in a parking lot [5], identify human gestures [11], recognize high-level narratives of multi-player games and player's strategy [9],[8], and to classify different types of interactions(dancing, hugging etc) between two people [12]. What has not been done before is the applications of such grammars on a distributed multimodal sensor network, and the creation of a hierarchy to produce training independent reasoning.

3 Recognizing Activities

Understanding human activity with a sensor network is a challenging task that involves the interpretation of large amounts of data into meaningful actions. To do so, our approach uses building maps to encode a person's location into a symbol that implies high degree of correlation to a specific activity. A set of such symbols provides an approximate string containing information about the activity. Our goal is to parse that string using a set of parsers defined in probabilistic grammars. The goal of these grammars is to interpret the low-level sensor data into meaningful, higher level semantics.

3.1 Initial Grammar Formulation

Our description of the "act of cooking" (we will refer to this as "cooking activity" from now on) is based on the kitchen floor plan shown in Figure 1(a). To simplify our discussion we first abstract out the sensing modality by assuming that there is a sensor that can reliably detect if a person is in areas D, R, P, S, and ST in Figure 1(a). These areas denote where the subject will be located when using the dining table, refrigerator, pantry, sink, and stove respectively. The symbol E is also used to denote the exit area of the kitchen. The whole kitchen was monitored by a single $iMote2$ sensor node from Intel equipped with a camera module we designed for this application (Figure 1(c)). The module uses an OV7649 camera module from Omnivision coupled to a 162 degree lens. This camera node acquires images at 8 frames per second, downsamples it to a 128×128 resolution and uses an image processing algorithm to extract the location of a person inside the kitchen. All the processing is done on the PXA271 processor on the node, and the node transmits a binary decision if cooking is detected.

To specify a sensory grammar that recognizes cooking, we must decompose the cooking activity into a sequence of basic actions. On the one hand, these actions should not be too abstract or too general because the difficulty of robustly detecting these actions increases significantly. On the other hand, these actions should be general enough to capture multiple instances of the activity. According to these considerations, we decompose the food preparation process into 4 main components, each of which requires a set of smaller actions:

1. **Get ingredients** from the refrigerator and the pantry.
2. **Prepare the dish** by spending time at the sink.

(a) (b) (c)

Fig. 1. a) Kitchen layout, b) Ceiling camera view of the kitchen, c)$iMote2$ node with camera module.

3. **Cook the food** by spending time at the stove.
4. **Serve dish** at the dining table.

Using this decomposition of the food preparation process, one could describe cooking as the ordered sequence of actions 1, 2, 3 and 4. However, this simple description of the cooking process is not adequate to capture all the different instances of a real cooking activity. Humans tend to forget and/or repeat actions without any obvious reason. For instance, people often do not get all the ingredients at once. Usually, they get a portion of them, they prepare it, then they get more ingredients of them and so on. Also, even in a specific activity, such as cooking, people tend to multi-task. For instance, while the food is on the stove, appetizers can be prepared at the sink or the initial preparation of the table might take place (put the dishes at the table, get sodas and drinks from the refrigerator, etc.). It becomes apparent from these observations that there is a huge number of different sequences of actions that describe a realistic cooking activity. A robust grammar definition therefore, should be able to recognize as many of these instances as possible and at the same time differentiate them from other similar activities that might take place in the monitored area.

3.2 Detailed Grammar Specification

Figure 2 shows the structure of a 2-Level grammar hierarchy for recognizing cooking activity based on the formulation presented in the previous section. At the lowest level, a sensor correlates a subject's location with areas and provides a string of symbols, where each symbol corresponds to an area in the kitchen (e.g. R, S, etc.). This string of symbols is then fed as input to the first level grammar which translates it and summarizes it to a new string of higher level semantics. These semantics may be related to the detection of the cooking activity (e.g $AccessFood$, $CookFood$, etc. as it will be shown later) or they may represent $Exceptions$ (e.g. $CommunicationAction$). $Exceptions$ are not different from other semantics produced by the grammar but their logical interpretation is different. For instance, a typical exception in the case of the cooking grammar

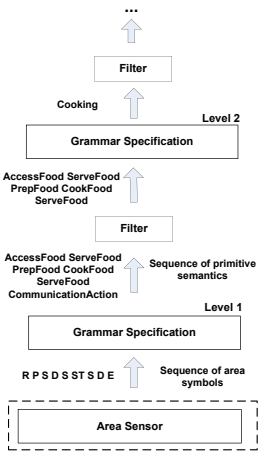


Fig. 2. 2-Level grammar hierarchy for the detection of cooking activity.

could be the presence of a person at the stove for less than a predefined minimum amount of time. This would mean that the person is passing by the stove without actually using it. Having defined *Exceptions*, we can see the output of a grammar as a sequence of symbols that may contain exceptions. As a result of this, the *Filter* component is introduced along with the notion of *Exceptions* and *Exception Handling*. The *Filter* component deals with the *Exceptions* by removing them from the output sequence of a grammar. The removed *Exceptions* are fed to an *Exception Handler* that is responsible for triggering the necessary actions assigned to each type of *Exception* (a detailed description of this process is given in Section 4). The exception-free sequence of symbols generated by the *Filter* component is then fed as input to the next grammar in the hierarchy. The second-level grammar uses the high-level semantics identified at the immediate previous level to describe and identify a typical cooking activity. The output of the second level grammar in the hierarchy is then passed through a *Filter* so that possible exceptions that were detected are handled properly. In the same way the output of the second-level grammar can be fed to any other higher level grammar for the detection of even higher level semantics.

The detailed implementation of the proposed grammar hierarchy is shown in Table 1. The grammar at Level 1 identifies the four cooking activity components ($FoodAction$) and the communication triggering component $CoommunicationAction$ by assuming that the underlying sensing modality will provide a sequence of activity regions; the *phonemes* of this language. Lines 1 and 2 specify the non-terminal and terminal symbols of this language. The terminal symbols are fed as input to the grammar and represent the different activity regions. Therefore, an input to the Level 1 grammar consists of a string of the predefined activity regions $R, P, S, ST, DandE$. The non-terminal symbols include the four cooking

Table 1. Cooking Grammar

Level 1 Grammar
Input: A sequence of any of the terminal symbols: $\{R, P, S, ST, D, E\}$
Output: A sequence of any of the following non-terminal symbols:
$\{AccessFood, PrepFood, CookFood, ServeFood, CommunicationAction\}$

1. V_N	$= \{Start, M, Action, CommunicationAction, FoodAction, AccessFood, PrepFood, CookFood, ServeFood\}$
2. V_T	$= \{R, P, S, ST, D\}$
3. $Start$	$\rightarrow M^{(1.0)}$
4. M	$\rightarrow M\ Action^{(0.5)} \mid Action^{(0.5)}$
5. $Action$	$\rightarrow FoodAction^{(0.5)} \mid CommunicationAction^{(0.5)}$
6. $CoommunicationAction$	$\rightarrow CommunicationAction\ E^{(0.5)} \mid E^{(0.5)}$
7. $FoodAction$	$\rightarrow AccessFood^{(0.25)} \mid PrepFood^{(0.25)} \mid CookFood^{(0.25)} \mid ServeFood^{(0.25)}$
8. $AccessFood$	$\rightarrow R\ AccessFood^{(0.16)} \mid P\ AccessFood^{(0.16)} \mid R\ S^{(0.16)} \mid P\ S^{(0.16)} \mid R\ ST^{(0.16)} \mid P\ ST^{(0.16)}$
9. $PrepFood$	$\rightarrow S\ PrepFood^{(0.5)} \mid S^{(0.5)}$
10. $CookFood$	$\rightarrow ST\ CookFood^{(0.5)} \mid ST^{(0.5)}$
11. $ServeFood$	$\rightarrow ServeFood\ S\ D^{(0.1)} \mid ServeFood\ R\ D^{(0.1)} \mid ServeFood\ ST\ D^{(0.1)} \mid ServeFood\ P\ D^{(0.1)} \mid ServeFood\ D^{(0.1)} \mid S\ D^{(0.1)} \mid R\ D^{(0.1)} \mid ST\ D^{(0.1)} \mid P\ D^{(0.1)} \mid D^{(0.1)}$

Level 2 Grammar
Input: A sequence of any of the terminal symbols:
$\{AccessFood, PrepFood, CookFood, ServeFood\}$
Output: A sequence of any of the following non-terminal symbols:
$\{Cooking\}$

1. V_N	$= \{Start, M, Cooking, Process, Prepare\}$
2. V_T	$= \{AccessFood, PrepFood, CookFood, ServeFood\}$
3. $Start$	$\rightarrow M^{(1.0)}$
4. M	$\rightarrow M\ Cooking^{(0.5)} \mid Cooking^{(0.5)}$
5. $Cooking$	$\rightarrow Process\ Cooking^{(0.2)} \mid CookFood\ Cooking^{(0.2)} \mid Prepare\ Cooking^{(0.2)} \mid Process\ CookFood\ Process^{(0.2)} \mid Prepare\ CookFood\ Process^{(0.2)}$
6. $Prepare$	$\rightarrow AccessFood\ Prepare^{(0.25)} \mid AccessFood^{(0.25)} \mid PrepFood\ Prepare^{(0.25)} \mid PrepFood^{(0.25)}$
7. $Process$	$\rightarrow ServeFood\ Process^{(0.25)} \mid Prepare\ Process^{(0.25)} \mid ServeFood\ Prepare^{(0.25)} \mid ServeFood^{(0.25)}$

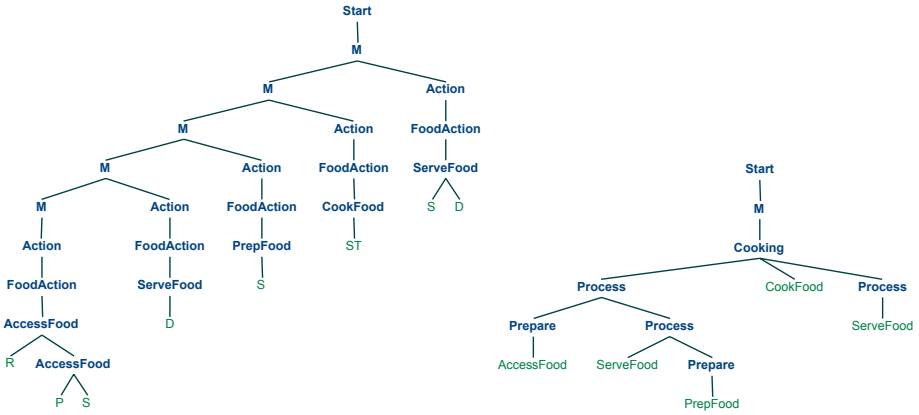
components and a set of standard symbols including the *Start* and M symbols[1]. The non-terminal symbols in a grammar represent the semantics to which the input of the grammar is mapped. Note also that there is an explicit hierarchy of the semantics in the grammar. For instance, the *Action* semantic is composed by the $FoodAction$ and/or $CommunicationAction$ semantics. This hierarchy can express the output of the grammar at different levels of granularity. For instance, the output of the first level grammar could be either a sequence of $FoodAction$ and $CommunicationAction$ semantics or a sequence of $AccesFood$, $PrepFood$, $CookFood$, $ServeFood$ and $CommunicationAction$ semantics. In this case, the output of the first-level grammar is the latter one.

The rest of the lines in Table 1 describe the production rules of the first-level grammar. Lines 4 and 5 describe how to recursively generate an arbitrary sequence of $FoodAction$ and $CommunicationAction$ semantics. Line 6 provides the definition of the $CommunicationAction$ semantic as any sequence of consecutive E terminal symbols. The $CommunicationAction$ semantic is nothing more than a specific instance of an *Exception*. This exception semantic embeds the actual communication protocol in the inference framework. Its purpose is to actually trigger communication in the sensor network when the person leaves the kitchen. Every time a person is moving out of the kitchen, the sensor node in the kitchen will parse its observations and it will transmit the output to the sensor network.

Line 7 describes the $FoodAction$ semantic as any of the $AccessFood$, $PrepFood$, $CookFood$ or $ServeFood$ semantics. Each one of these semantics is defined as a sequence of terminal symbols in Lines 8-11. Line 8 defines the $AccessFood$ semantic as any trip between the refrigerator R and the pantry P, that ends at the sink S or the stove ST. Lines 9 and 10 define the $PrepFood$ and $CookFood$ semantics as being at the sink S and the stove ST respectively. Line 11 describes $ServeFood$ as any sequence of trips between any of the possible areas R, P, S, and ST and the dinning table D. Note that the number of appearances of each of the terminal symbols or their order of appearance is not explicitly defined in Lines 8 and 11. However, the recursive nature of the production rules allows the unified description of numerous different expressions for the $AccessFood$ and $ServeFood$ semantics. This shows the great generative power of grammars where very simple rules similar to the one in the human language can be used to describe numerous instances of the same complex activity.

The grammar at Level 2 takes as input the activity components identified at Level 1 to describe a typical cooking activity. As it can be seen by Line 2, the vocabulary of the second level grammar is composed by the output semantics of the first level grammar. The output of this level is a sequence of $Cooking$ semantics. Lines 3 and 4 use recursion to allow multiple appearances of the cooking activity. The $Cooking$ semantic is described in Line 5 as any sequence of the $CookFood$, $Process$ and $ServeFood$ semantics that starts with the $Process$ or $Prepapre$ semantics, ends with the $Process$ semantic and contains at least

[1] The *Start* symbol is a standard symbol used in grammar descriptions to represent the starting point of the grammar. We use the M symbol for recursion.



(a) Level-1 parse tree (b) Level-2 parse tree

Fig. 3. Example parse trees for the 2-Level cooking grammar hierarchy.

one $CookFood$ semantic. Line 6 describes the $Prepare$ semantic as any sequence of the terminal symbols excluding the $CookFood$ symbol. Line 7 defines the $Process$ semantic as any sequence of the $Prepare$ and $ServeFood$ semantics that contains at least once the $ServeFood$ semantic. Note that because of the recursive nature of their definition, each of the production rules in Lines 5 and 6 can correspond to a huge number of different instances of the cooking activity. However, this large number of different instances are described in 6 lines of production rules for the second level grammar and 11 lines of production rules for the first level grammar.

Since our grammar is probabilistic, each production rule is associated with a probability denoted as a superscript inside parentheses at the end of each production rule. Note that the sum of the production probabilities for each non-terminal sums up to one. In the grammars shown in Table 1, we assume that there is a uniform probability distribution for the production rules. However, in some particular scenarios these probabilities could be learned from ground truth data. This could be done by applying this grammar on real data and keeping track of how often each production rule is used. The more often a production rule is used the higher its probability.

The grammar parser makes use of these probabilities, to calculate the most probable string of non-terminal symbols for a given input string of terminal symbols. Level 1 of the grammar translates a sequence of object areas (such as R, D etc.) into a new sequence of basic cooking components ($FoodAction$) in a probabilistic way. The probabilistic nature of this translation implies that the same input sequence might correspond to different sequences of the basic cooking components according to the grammar definition. For each of these possible different output sequences a probability is computed based on the individual probabilities of the production rules used to derive each output sequence. The output sequence with the highest probability is chosen as the final output. This output is then fed into a Level 2 grammar which in a similar way translates a

sequence of basic cooking actions to a sequence of cooking actions. For instance, Figure 3 shows the most probable parse trees for both levels and for a given input sequence of object areas. As it can be easily verified, each edge in the tree corresponds to a production rule of the corresponding grammar. The probability assigned to the parse tree is computed by multiplying the probabilities at each branch from the root to the leaves and then summing the probabilities of all the branches in the tree. For instance, in Figure 3(a) there are 5 branches with probabilities (p_1 corresponds to the leftmost branch and p_5 to the rightmost branch):

$$p_1 = (0.5)^4 \times 0.5 \times 0.5 \times 0.25 \times (0.166)^2 = 0.0001075$$
$$p_2 = (0.5)^4 \times 0.5 \times 0.25 \times 0.1 = 0.000781$$
$$p_3 = (0.5)^3 \times 0.5 \times 0.25 \times 0.5 = 0.0078125$$
$$p_4 = (0.5)^2 \times 0.5 \times 0.25 \times 0.5 = 0.015625$$
$$p_5 = (0.5) \times 0.5 \times 0.25 \times 0.1 = 0.00625$$

The probability of the tree is equal to $\sum_{i=1}^{5} p_i = 0.0305$. In exactly the same way, a probability for the tree shown in Figure 3(b) can be computed using again the probabilities assigned to the production rules shown in Table 1.

4 Middleware Implementation

Our supporting middleware architecture, creates a node-level abstraction that isolates the grammar developer from low-level embedded systems programming by automating node-level code development. For this we have developed a desktop tool that automates code development for grammar hierarchies. Using this tool, the grammar developer can develop the entire grammar hierarchy by entering high-level grammar definitions much like the ones shown in Table 1. The output of the tool is a set of binary modules for the SOS operating system [2], that is automatically installed and configured on the target sensor nodes. Our design makes heavy use of dynamic modules and messaging of the SOS operating system.

The code generation process is illustrated in Figure 4. On the PC side, a module composer tool takes grammar definitions as inputs and generates a parser in C for each grammar using a parser generator. The resulting parser code is then passed to a module generator that encapsules the parser into a SOS binary grammar module. The well-defined structure of the grammar hierarchy allows us to design modules with well-defined inputs and outputs, so that they can be stacked into a hierarchy inside a sensor node processor. Each generated module has a standard message handler (H), an input token buffer (IB), a parser (P), an output token buffer (OB) and a destination table (DT). The inputs for each grammar module are passed as messages of tokens and are buffered in the input buffer of the destination module. When the module determines that enough token have been collected, it invokes the parser that parses the contents of the token buffer and forwards all the output tokens to all the subscribed modules listed in

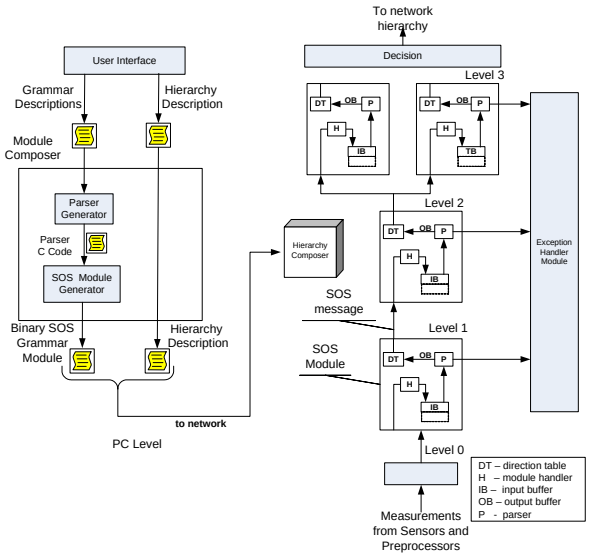


Fig. 4. Grammar to SOS module conversion

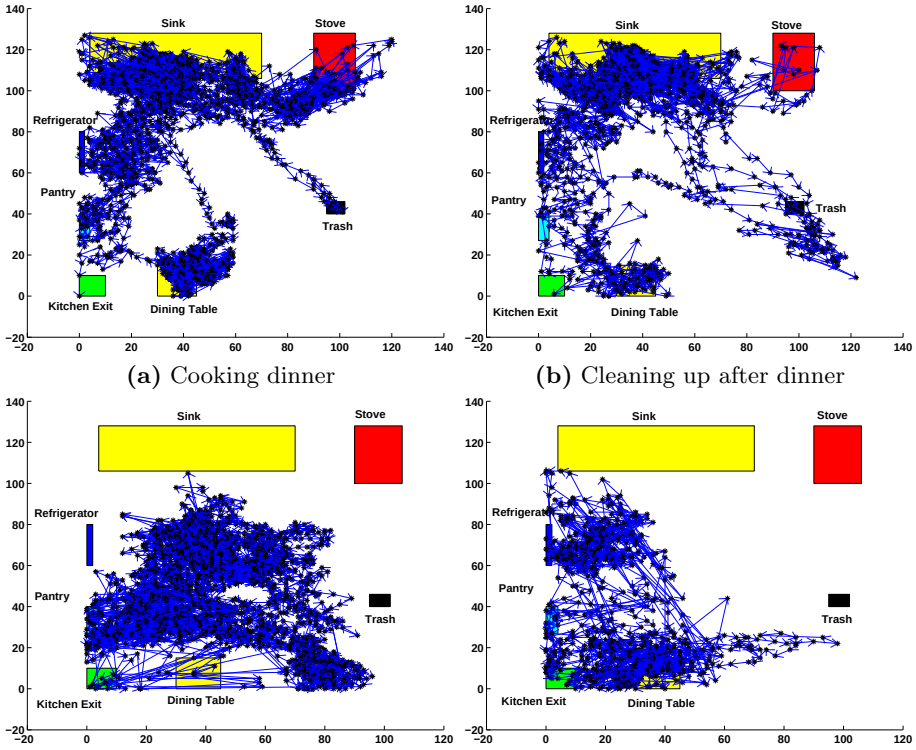
the destination table of the module. Our design tries to exploit the underlying OS features to minimize book-keeping inside the module. Any inputs received while the parser is executing are automatically buffered by the SOS messaging services and are processed on the next run of the parser. The dynamic module support functions of the operating system allow the developer tools on the PC side to install the modules in the node's memory at runtime. These modules can be dynamically re-wired by updating the destination table (DT) entries for each of the involved modules with the module IDs of the destination modules.

All the interaction between the grammar hierarchy on each node and the developer's PC takes place through a hierarchy composer module residing at the node level. The hierarchy composer handles hierarchy descriptions and hierarchy modification requests from the PC toolchain. It also maintains all the state information of the hierarchy configuration and can return it to the PC toolchain when queried. All re-wiring of the grammar modules is done by the composer via the OS messaging mechanism.

The uniform structure of the grammar modules facilitates debugging. The grammar developer can easily examine the outputs of each module by registering a debug module on the PC to receive the outputs from all levels of a grammar hierarchy. In our implementation this is done through a scripting command interface that also allows a developer to integrate hand-written modules in the same operating system that runs the automatically generated grammar modules.

5 Cooking Recognition Experiments

Our scheme was evaluated on data acquired in a series of experiments in the kitchen deployment described earlier. In every experiment the person in the kitchen was preparing either breakfast or dinner. The data collection started



(a) Cooking dinner **(b)** Cleaning up after dinner

(c) Cleaning the floor **(d)** Sorting the groceries

Fig. 5. Area definitions and example data sets.

when the person was entering the kitchen or while the person was already in the kitchen. It was stopped when the person started eating breakfast or dinner at the dining table. The person in each experiment was not aware of what he would have to cook until a couple of minutes before the recording of the data. This prevented the person from using pre-meditated moves. The person cooking was also unaware of the actual grammar hierarchy definition. In total, 10 cooking traces were collected lasting from approximately 10 minutes (breakfast) to 50 minutes (dinner) each.

In order to challenge the capabilities of the proposed scheme, we also recorded a set of activities other than cooking in the same kitchen area. In total, 5 different traces were recorded on different days. These activities included cleaning the kitchen after having dinner, cleaning the floor of the kitchen and sorting the groceries after returning from the super-market. Especially when cleaning up the kitchen after having dinner, the areas visited are almost the same as when cooking. This can be seen in Figure 5(a) and Figure 5(b). The recorded traces of image locations are very similar. However, the grammar hierarchy should only recognize the cooking activity trace.

Table 2. Recognition performance of the proposed cooking grammar hierarchy

Kitchen Activity	Number of Traces	Correctly Classified (Ground Truth)	Correctly Classified (Filtered)
Cooking	10	10	10
Cleaning	5	4	5
Other	1	1	1

Table 3. The effect of imperfect sensing in the observed sequences of areas.

Kitchen Activity	Number of Areas(Ground Truth)	Number of Areas(After Filtering)
Dinner	116	109
Breakfast	15	19
Cleaning 1	12	9

For each recorded activity trace the ground truth area information activity was also recorded. This was done manually by a person that examined a recorded video for each recorded trace. The ground truth area information was used to investigate the false negatives and false positives of the area sensor.

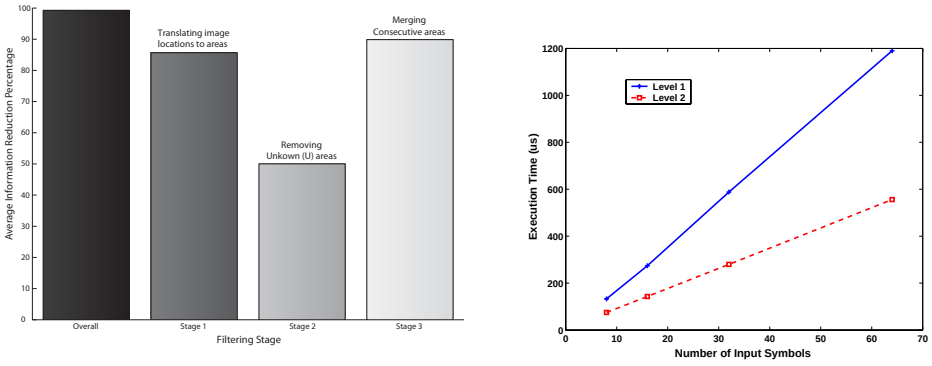
Table 2 shows the recognition results of the proposed grammar hierarchy for all the recorded activities and for both the ground truth data and the actual data provided by the area sensor. In both cases, all the cooking activities are correctly classified. What is even more interesting is the fact that the proposed scheme can differentiate between very similar activities such as cooking and cleaning. This demonstrates that the grammar definition is general enough to capture various instances of cooking activities, but at the same time it is specific enough to robustly differentiate cooking from other similar activities. This is a result of the second level grammar definition shown in Table 1. There, the cooking activity must always include visits to the stove and it should always include a transition to the dining table after the last visit to the stove. However, when people are cleaning, they either do not visit the stove area or any other area (for instance cleaning the floor or sorting the groceries) or they do not move to the dining table after cleaning everything (including the stove). These restrictions in the description of the cooking activity allow the system to differentiate between cooking and cleaning.

However, as for example shown in Table 2, the proposed system fails to correctly classify the cleaning activity shown in Figure 5(b) when the ground truth data is used. This is due to the successful calibration of the area sensor. The table area was defined by using real image locations acquired when a person was sitting or standing up at the table. This data gave us a very precise definition of the table area. While cleaning the table, for instance picking up the plates and putting them into the sink, people do not sit at the dining table and therefore the area sensor would rarely detect the dining table area in such a case. However, this table area information is recorded in the ground truth data. Based on the grammar definition this consists of a typical cooking activity and the specific trace is incorrectly classified as cooking.

The experimental data provides insight on how to better calibrate the area sensor. Table 3 shows the number of area symbols generated by the area sensor

Table 4. The effect of raw sensing data filtering.

Kitchen Activity	Number of Centroids	Number of Areas After Filtering		
		Stage 1	Stage 2	Stage 3 (grammar input)
Dinner	6648	1456	728	109
Breakfast	2924	446	223	12
Cleaning	2583	421	211	9



(a) (b)

Fig. 6. a) The effect of filtering on the number of area symbols fed as input to the grammar hierarchy. b) Execution time of the cooking grammar on the $iMote2$ sensor node as a function of the number of input symbols.

versus the ground truth number of area symbols for three of the collected traces. It is clear that the area sensor gives both false positives and false negatives. The false positives are caused by the fact that the area of the kitchen used in our experiments was small. As a result, the areas of the refrigerator and the pantry (Figure 1) are very close and when a person tries to use the pantry it is possible that the refrigerator area will also be recognized. The false negatives are mainly caused by small movements of the person in the kitchen that cannot be robustly captured at the 128×128 resolution. For instance, in many cases the person was able to reach the sink by simply stretching but without moving out of the stove area. In this case, the sink would appear in the ground truth data but not in the output of the area sensor.

Figure 6(a) shows the effect of a 3-stage sensor calibration mechanism used to reduce the size of the input to the grammar hierarchy for all the collected experimental data. The first stage converts the time series of image locations to a time series of visited areas. The $Undefined$ area symbol can be produced when the monitored person is moving in any place in the kitchen that is not one of the predefined areas. The second filtering stage, removes all the $Undefined$ area symbols because they are not used by our grammar and they increase the size of the input to the grammar hierarchy. After removing the $Undefined$ area symbols, consecutive appearances of the same area symbol might appear. The third filtering stage merges all these consecutive appearances to a single area symbol.

Table 5. Generated Module Code per Grammar Definition

	Number of Lines		
	Grammar Definition	**Generated Module**	**Generated Parser**
Level 1	11	395	917
Level 2	7	395	893
Total	18	2600	

Table 6. Cooking Grammar Module Memory Usage

	*iMote*2		**Micaz**	
	Level 1	**Level 2**	**Level 1**	**Level 2**
Flash(Bytes)	7224	7072	2594	2540
RAM(Bytes)	143	143	143	143
Stack(Bytes)	96	96	96	96
SOS Kernel + MAC Size				
Flash(Bytes)	135616		54602	
RAM(Bytes)	5392		3073	

The average percentage of information reduction at each stage of the calibration and the total one are shown. The overall average information reduction is approximately 99%. This is the percentage of reduction in the number of symbols that are given as input to the grammar hierarchy with respect to the number of image locations initially recorded. The average percentage of information reduction for each one of the three stages of filtering are illustrated by the other three bars in Figure 6(a). At stage 1 there is a reduction of 85% from translating the raw image locations to areas (including the $Undefined$ area). At stage 2, we get a reduction of approximately 50% by simply removing all the $Undefined$ areas. At stage 3, the number of symbols is reduced by 90% and the final area sequence is fed to grammar hierarchy. Due to the sensor calibration the number of symbols eventually fed as input to the grammar hierarchy (approximately 10 to 100) are orders of magnitude less than the initial number of image centroids recorded (2583 to 6648), as shown in Table 4. These numbers demonstrate the feasibility of such a system running in real time on a sensor network. An input of 10 to 20 symbols is relatively small and can be parsed in a very short period of time even on an a sensor node as will be made clear in the next section. In addition, the fact that activities lasting as much as 50 minutes can be reduced down to a sequence of only 100 symbols shows that modeling human activity as a sequence of actions could meet the real time requirements and limitations of sensor networks.

Table 4 provides some more insight into the effect of sensor calibration in the size of input fed to the inference framework. In the case of dinner preparation, 6648 image locations were acquired that were finally reduced down to only 109 area symbols. Similarly, in the cases of preparing breakfast and cleaning, 2924 and 2583 image locations were reduced down to 12 and 9 area symbols respectively. These numbers demonstrate the feasibility of such a system running in real-time on a sensor network. An input of 10 to 20 symbols is relatively small and can be parsed in a very short period of time even on a sensor node as will

be made clear in the next section. In addition, the fact that activities lasting as much as 50 minutes can be reduced down to a sequence of only 100 symbols shows that modeling human activity as a sequence of actions could meet the real time requirements and limitations of sensor networks.

5.1 Middleware Performance Evaluation

The middleware configuration was tested using $iMote2$ nodes [10] with camera modules and Micaz motes. The final integrated design of grammars, middleware tools and deployment were tested using $iMote2$ nodes.

Our middleware evaluation aims to determine the effectiveness of our automatic code generator using the cooking grammar definitions from Table 1. Table 5 demonstrates the effectiveness of our node level abstraction from the developers standpoint. With 18 lines of grammar definition, our system automatically generates 2600 lines of SOS code. The memory footprint of each generated grammar is shown in Table 6. This compares favorably with the OS size and the memory resources available on the sensor nodes available in today's technology. On Micaz the two levels of the cooking grammar consume about 5KB of Flash and 239 bytes of RAM. The stack size reported refers to a pre-set parser stack and RAM refers to the memory required by the grammar modules. This shows that our architecture could be implemented on existing small platforms such as the Micaz node and UCLA's Cyclops camera module.

The runtime of each grammar is of order $O(n|R|)$ where n is the length of the input token string and $|R|$ is the number of lines in the grammar [1]. In terms of the cooking grammar running on an $iMote2$, the runtime as a function of the input tokens is shown in Figure 6b. This suggests that in terms of runtime it is better to have hierarchical grammar modules instead of implementing a monolithic grammar for the entire behavior. These numbers also suggest that small nodes can accommodate multiple hierarchies at a time that help to reduce the sensed information over a large window to a few bits: the results of the activity recognition.

6 Conclusion

The results obtained in this study are very encouraging. Our probabilistic grammar formulation can robustly detect cooking activity across different meals and cooks. The middleware implementation provides a programmer abstraction that allows developers to focus on the specification of behaviors and other safety rules, significantly reducing development and debugging time. One drawback of the current system is that the imager node can only operate with one person in the field of view. Multiple targets and power management recognition is handled by other parts of our research [references witheld]. Nonetheless, our system provides a powerful tool for rapidly developing descriptions for new behaviors. With it, a small collection of other behaviors including a system for supervising traffic rules in the roads on a model street map has already been completed. For aging in place, a network of 8 nodes has already been deployed into a house to

collect an extensive set of data that will be used for the development of a more extensive library of activities. In our future work, we also plan to examine the development of distributed grammars, grammar induction techniques for learning behavior patterns from data and using grammars to stitch together data from multiple sensing modalities.

References

1. 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.
2. Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, and Mani Srivastava. A dynamic operating system for sensor nodes. In *Proceedings of the Third International Conference on Mobile Systems, Applications, And Services (Mobisys)*, 2005.
3. 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.
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. Yuri A. Ivanov and Aaron F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):852–872, 2000.
6. Lin Liao, Dieter Fox, and Henry Kautz. Location-based activity recognition using relational markov models. In *Nineteenth International Joint Conference on Artificial Intelligence*, 2005.
7. J. Liu and F. Zhao. Towards semantic services for sensor-rich information systems. In *Proceedings of IEEE BaseNets 2005*, October 2005.
8. D. Minnen, I. Essa, and Thad Starner. Expectation grammars: Leveraging high-level expectations for activity recognition. volume 2, page 626, 2003.
9. Darnell Moore and Irfan Essa. Recognizing multitasked activities from video using stochastic context-free grammar. pages 770–776, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.
10. L. Nachman. Imote2, http://www.tinyos.net/ttx-02-2005/platforms/ttx05-imote2.ppt, 2006.
11. A.S. Ogale, A. Karapurkar, and Y. Aloimonos. View-invariant modeling and recognition of human actions using grammars. *Workshop on Dynamical Vision at ICCV'05*, October 2005.
12. S. Park and J. K. Aggarwal. Event semantics in two-person interactions. August 2004.
13. D. Patterson and M. Philipose D. Fox, H. Kautz. Fine-grained activity recognition by aggregating abstract object usage. In *IEEE International Symposium on Wearable Computers*, October 2005.
14. 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*, 03(4):50–57, 2004.
15. E. Munguia Tapia, S. S. Intille, and K. Larson. Activity recognition in the home setting using simple and ubiquitous sensors. In *PERVASIVE 2004*, 2004.

16. 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 Wireless Sensor Networks (EWSN)*, February 2006.