KNOWLEDGE ASSISTED HUMAN ACTIVITY RECOGNITION FOR IMPROVED ACCURACY AND PROGRAMMABILITY

By

EUNJU KIM

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2013

To my Parents, Sisters, Brothers, and Sagnik

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Sumi Helal, for his huge support and encouragement in pursuing topics that are of interest to me and for being accessible for discussion at all times. This work would not have been possible without his insightful advises and supports. Also, I would like to show my deep gratitude and respect to my role model, Dr. Sumi Helal for his guidance and mentorship. No amount of words can express my gratitude to him.

I am indebted to Dr. Douglas D. Dankel, Dr. Ye Xia, Dr. Alin Dobra, Dr. Pradeep Kumar, Dr. Paul Fishwick, and Dr. Hyojin Kim. They kindly agreed to be part of the supervisory committee; their suggestions and advice greatly enriched this research effort.

I would like to thank to Dr. Eunman Choi, Dr. Kihyun Um, and Dr. Kyungeun Cho in Korea. They encouraged me to study in US.

I extend my deep gratitude to Dr. Christopher Nugent at the University of Ulster, Ireland for his suggestions and advice. Also, some experiments for activity recognition validation had been performed at his smart lab. I thank Dr. Marko Turpeinen at Aalto University, Finland for the opportunity to experience cutting-edge multidisciplinary research in the area of pervasive computing.

I would like to appreciate Mr. John Bowers and Mrs. Joan Crisman for their helpful advice and guide.

My life at graduate school has been an enriching experience, thanks to my colleagues and visiting researchers at the Mobile and Pervasive Computing Lab. and friends in CISE department— Dr. Jungmin Shin, Dr. Raja Bose, Dr. Choonhwa Lee, Dr.

Hyeyoung Lee, Jungwook Park, Dr. Eunsun Cho, Dr. Younsik Sung, Dr. Chao Chen, Shangtonu Hossain, Prithvi Raj, Duckee Lee, Jaewoong Lee, Yi Xu, and Sirui Chen.

Last but foremost, I thank my family—Father, Mother, Youngju, Sunae, Hunam, Daehan, Younghan, and my boyfriend, Sagnik Pal, for their unconditional love and support. Throughout my life, they have always stood by me.

TABLE OF CONTENTS

page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AR	Activity Recognition
GAF	Generic Activity Framework
SGAF	Semantic Generic Activity Framework
TAM	Traditional Activity Model
GAM	Generic activity framework based Activity Model
SGAM	Semantic and Generic activity framework based Activity Model
MLNNK	Multi Layer Neural Network
FL	Fuzzy Logic
HMM	Hidden Markov Model
CRF	Conditional random Field Model
AOS	Activity Observation System
CF	Certainty Factor
MB	Measure of Belief
MD	Measure of Disbelief
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor
of Philosophy

KNOWLEDGE ASSISTED HUMAN ACTIVITY RECOGNITION FOR IMPROVED ACCURACY AND PROGRAMMABILITY

By

Eunju Kim

May 2013

Chair: Abdelsalam (Sumi) Helal
Major: Computer Engineering

Activity recognition (AR) is a key technology for developing human-centric applications in smart environments. However, state-of-the-art AR technology cannot be used for addressing real world problems due to insufficient accuracy and lack of a usable activity recognition programming model. To address these issues, a new AR approach is developed in this dissertation.

AR performance is strongly dependent on the accuracy of the underlying activity model. Therefore, it is essential to examine and develop an activity model that can capture and represent the complex nature of human activities more precisely. To address this issue, we introduce generic activity framework (GAF) and activity semantics. The GAF is a refined hierarchical composition structure of the traditional activity theory. Activity semantics are highly evidential knowledge that can identify activities more accurately in ambiguous situations. We compare our activity model with traditional activity model in terms of attainable recognition certainty.

Two new AR algorithms— Multilayer Neural Network (MLNNK) based algorithm and fuzzy logic (FL) based algorithm—have been developed in this dissertation. These

algorithms utilize and work in tandem with the developed activity framework and model. The MLNNK based AR algorithm illustrates the high recognition accuracy of generic activity framework modeling approach. FL based AR algorithm utilizes both activity semantics and generic activity framework.

For achieving high accuracy, it is important to identify and mitigate the most debilitating sources of uncertainty. The efficacy of AR systems is usually quantified based on the recognition accuracy at the final step of activity recognition process. This method does not reveal the uncertainty sources that affect overall performance significantly. Therefore, it is necessary to quantify every possible uncertainty source through all activity recognition procedures. To address this issue, metrics and measurement methods for each uncertainty source are developed.

AR technology should provide programmable interface to developers to support AR system design change according to new application requirements or AR environment changes. This dissertation classifies developers into three categories: smart space developer, activity model and algorithm developers, and application developers. The hierarchical aspects of our generic activity framework decouple the observation subsystem from the rest of the activity model. We demonstrate the value of this decoupling by experimentally comparing the level of effort needed in making sensor changes.

CHAPTER 1
INTRODUCTION

This dissertation focuses on developing practical activity recognition (AR) technology that can be widely applied to many human-centric applications such as healthcare and elder care [1,2,6]. Understanding human desires and intentions through activity recognition is a key prerequisite to determine the services that a pervasive space should provide to a user in a variety of contexts [10]. Therefore, activity recognition has been actively investigated in past decades [7,10,12,62,65,69]. These approaches represent significant and promising contributions to pervasive computing and its applications. Nevertheless, despite significant progress, state-of-the-art activity recognition (AR) technology is not practically acceptable because of several limitations such as low recognition accuracy and limited system programmability.

To be a more practical technology, activity recognition should guarantee enough accuracy because AR technology is often utilized by applications that are directly related to human wellbeing. However, recognizing human activities accurately is very challenging due to many factors including the diverse characteristics of human activities.

In addition to being accurate, AR technology should offer ease of implement ability. It should provide usable programmable interfaces to smart space developers and application programmers for easy development or upgrade. Nevertheless, activity recognition programmability has not received much attention from people in this domain. Therefore, to address these issues, this dissertation proposes a new approach that can improve both accuracy and programmability of AR technology.

The next section provides an overview of sensor based human activity recognition technology and Section 1.2 describes research motivations of activity recognition. Section 1.3 enumerates the current challenges in activity recognition technology. The objectives of the current research are shown in Section 1.4. Section 1.5 details the organization of the chapters in this dissertation.

1.1 Overview of Sensor based Activity Recognition

Sensor based activity recognition (AR) systems recognize human activities through interpretation of sensor data. For recognizing activities, a smart space is instrumented with sensors and devices. Figure 1-1 (b) shows an example of sensor setup in a dining room for sensing “eating” activity [8,10].

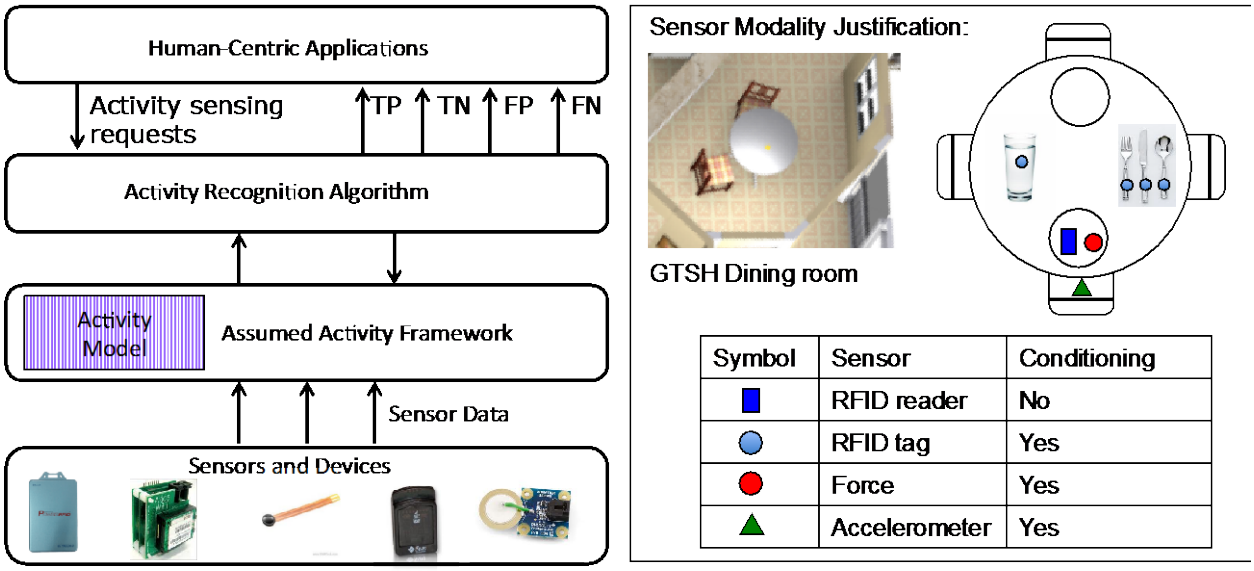


Figure 1-1. Sensor based activity recognition system. (a) Structure of sensor based activity recognition system (b) An example of sensor installation in a smart space

When sensor data is sent to activity recognition system, activity recognition algorithm interprets the sensor data based on activity model and activity knowledge.

Activity recognition system in this dissertation is composed of a domain specific activity model, an AR algorithm and activity knowledge. All three subsystems are based on a "generic activity framework" shown in Figure 1-1 (a).

1.2 Research Motivations

Activity recognition technology can significantly empower many human centric applications in a variety of areas including healthcare and care of the elderly [1,2]. These applications are very critical for enhancing the quality of human life. For example, many people including the elderly, infirm or patients need a caregiver's help. However, human caregivers are prone to errors and inconsistencies. Intelligent activity based computing applications that can complement, supplement or replace human caregivers are a possible solution to the aforementioned problems. The consistent service provided by an intelligent care system is especially useful for people who need long-term care. Computing systems can assist human caregivers by making up for their limitations. For instance, computing systems can continuously observe patients and monitor their vital signs or food and medicine intake. Additionally, the system can store observation data, analyze the data quickly, and create meaningful information for human caregivers and doctors. Activity recognition technology is key to improving the quality of life and it has received a lot of attention from researchers in the past decade.

However, the accuracy of current activity recognition is not acceptable for real world applications. A lot of AR research is focused on finding AR algorithms with improved accuracy using several techniques such as machine learning, artificial intelligence or statistics. Nevertheless, it is difficult to find a one-size-fits-all activity recognition algorithm because human activities reflect high human intelligence; therefore the activities are more diverse and complex than what a single computing

algorithm can process. Therefore, a new AR approach that integrates several features of previously reported systems needs to be explored.

Another area that requires attention is activity recognition programming model. Activity recognition programming model is very critical for activity recognition (AR) technology to be practical.

Activity recognition (AR) technology should provide programmable interface to application developers so that the developers can easily utilize AR technology in their application without much knowledge of activity recognition. Moreover, activity recognition technology is continuously evolving and it is important to devise techniques for incorporating these improvements into existing systems. To illustrate, sensor technology and nanotechnology continue to provide new devices with improved performance. Also new activity modeling techniques and algorithms are being continually developed. It should be possible to seamlessly replace and upgrade sensors, models, and algorithms. Therefore, programmable activity recognition is important. However, currently, activity recognition programming model research is in its infancy and further research is required.

For improving recognition accuracy and activity recognition programmability, we propose knowledge assisted activity recognition approach, which is analogous to activity recognition by human beings. Humans can recognize other humans' activities better than a computing algorithm because humans utilize their accumulated knowledge. On the contrary, even though accumulated knowledge is a good source of information, current activity recognition systems utilize limited knowledge only.

Therefore there is no systematic method for achieving knowledge assisted activity recognition.

As a solution, this dissertation seeks to utilize commonsense knowledge and experts' knowledge for activity recognition. Commonsense knowledge of activities comprises of facts about activities that are widely known to most people. For example, most people know that tools such as spoon, fork, or knife are used for having food. Therefore, this information comes under the purview of common sense knowledge. Domain experts have specialized knowledge about activities. For instance, doctors know the relationship between "eating" and glucose level. We then develop methods to organize and utilize the knowledge to improve activity recognition accuracy and programmability. Finally, we validate and compare our developed approach with other approaches.

1.3 Current Challenges

As discussed in Section 1.2, activity recognition accuracy and programmability are crucial for developing a practical activity recognition technology. However, enhancing accuracy and programmability is very challenging due to several reasons. In this section, we describe detailed challenges to improve accuracy and programmability in activity recognition.

1.3.1 Accuracy

Human activities have several characteristics and intricacies that reflect humans' high intelligence. To increase activity recognition accuracy, activity recognition should address these complexities as enumerated below:

Concurrent activities. People may be involved in actions corresponding to several activities at the same time. For example, people watch TV while talking to

friends. These behaviors are not sequential; therefore, an activity model needs to represent these characteristics of activities [7,11].

Interleaved activities. In a real world scenario, some activities may be interrupted by other activities before completion whereas some are not. For instance, if a friend calls while cooking, the resident may pause cooking and talk to the friend for a while. After talking, he/she continues to cook [7,11].

Ambiguity. Even though sensor detects user activities well, the exact interpretation may be difficult. For example, we cannot guarantee that a person really takes a medicine even though a sensor detects opening the medicine bottle because the sensor could be reporting on other related activities such as checking whether bottle is empty or cleaning bottle but not taking the medicine.

Variety. Humans can perform activities in a variety of ways. For example, there are multiple ways to eating such as having a meal or having a snack. Typical scenario based activity recognition is not enough to handle this variety [7].

Multiple subjects. More than one person could occupy the same space and perform activities together sometimes. An activity model needs to be capable of associating the detected activities with the resident who actually executed them [7,11].

1.3.2 Programmability

To improve the programmability of activity recognition, several challenges need to be addressed. In particular, the new activity recognition approach should incorporate the following features to enhance programmability:

Scalable activity observation system. It is difficult to find an effective sensor set for recognition of target activities at the first trial. Therefore, it may need test several sensors with addition or deletion of new sensors. This sensor addition or deletion should

not be too difficult. And it should not affect other parts of AR system such as activity model or activity recognition algorithm too much.

Flexible activity model or algorithm. Activity modeling technique and activity recognition algorithm should be replaced with better techniques. And it will be upgraded if target activities are changed. These activity model or AR algorithm change should be encapsulated to the applications that utilize the activity recognition to minimize the effect of the upgrade.

Programmable Interface. Activity recognition technology should provide programmable interface to application developers. The requirement of interface may vary depending on developers and application domain. For example, some developers want real time activity recognition whereas other developers require assurance mechanism for recognized activities. Practical activity recognition interface should satisfy the various requirements.

1.4 Research Objectives

1.4.1 Accuracy Enhancement of Activity Recognition

The objectives of accurate activity recognition have been enumerated below:

- Human activity framework will be extended after observing human activities and their components.
- Activity semantic knowledge will be collected and applied to activity modeling.
- A new activity recognition algorithm that can cooperate with the new activity modeling technique will be developed.

1.4.2 Uncertainty Analysis and Management

The main objectives of uncertainty analysis and management study have been enumerated below:

- Uncertainty sources will be analyzed thoroughly.

- Quantifiable indices for uncertainty due to each identified source will be developed.
- Uncertainty from all sources will be measured and compared to find the most detrimental uncertainty source.

1.4.3 Development of Activity Recognition Programming Model

The major objectives of activity recognition programming model have been enumerated below:

- A new method to utilize knowledge for activity recognition programmability will be developed.
- Programmable interface for application developers will be designed and implemented.
- The developed activity-programming model will be tested.

1.5 Dissertation Organization

This dissertation is organized as follows. Chapter 2 presents literature reviews. Overview of the problems in current activity recognition is provided in Chapter 3 using the hierarchical structure of problems. Then, the detailed discussion of the problem in each layer and developed solution of the problem are explained from Chapter 4 to Chapter 8 respectively. Chapter 4 describes a new activity modeling technique and activity semantics. Two different activity recognition algorithms are introduced in Chapter 5 and Chapter 6 respectively. Chapter 7 outlines uncertainty sources, their analysis methods, and management techniques. Chapter 8 deals with activity recognition programming model.

CHAPTER 2
LITERATURE REVIEW

This chapter summarizes prior research in activity recognition, uncertainty analysis, and activity recognition programming model.

2.1 Activity Theory - Origin of Activity Modeling

L. S. Vygotsk who was a psychologist during 1920s and 1930s found activity theory. Later, A. N. Leontjev and A. R. Lurija further developed the activity theory [19,39,41,64]. Activity theory was first applied to human-computer interaction (HCI) in the early 1980s [39]. Recently, it is applied implicitly or explicitly in a lot of activity recognition model.

The definition of activity theory is described in [39] as: “Activity Theory is a philosophical and cross-disciplinary framework for studying different forms of human practices as development processes, both individual and social levels interlinked at the same time.” The activity theory has four components such as subject, tool, objective, and outcome [64]. A subject is a person who performs an activity. An objective is a plan or common idea that can be shared for manipulation and transformation by the participants of the activity. Tool is an artifact a subject uses to fulfill an objective. Outcome is another artifact or activity that are result of the activity. Transforming the objective into an outcome motivates the performing of an activity. For instance, having one’s own house is an objective and the purchased house is the outcome. Transforming an object into an outcome requires various tools including documents, equipment, devices, etc. These relationships among components are presented with lines in Figure 2-1.

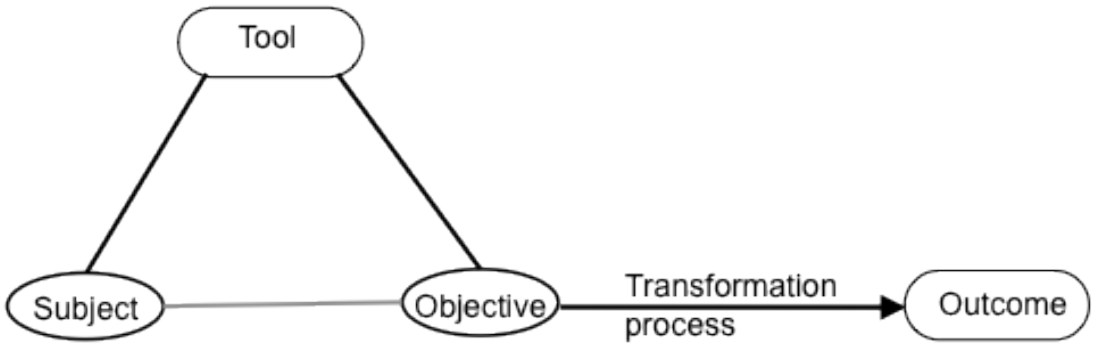


Figure 2-1. Structure of activity theory

Gray line between subject and objective represents a mediated relationship whereas bold line indicates direct relationship between components. Subject and tool have a direct relationship because a subject uses a tool in person. A tool mediates the relationship between an objective and subject, which is represented in gray line, because subjects achieve their objective using tools.

As shown in Table 2-1, activity theory has a three-layered hierarchical structure and activity is composed of actions and an action is composed of operations [39].

Table 2-1. Hierarchical layers of activity, action, and operation

Levels	Related purpose	Example of purpose
Activity	Motive	Completing a software project
Action	Goal	Programming a module
Operation	Conditions	Using an operating system

Activities are composed of cooperative actions or chains of actions. In Table1, these actions are all related to the motive of an activity. Each action has a goal and consists of operations to reach the goal. Operation is a unit component and it depends on the faced condition where the operation performs. The detailed description of each level can be found in [9,39]. Even though activity theory is well known and is often used in activity recognition research, it has some limitations.

Firstly, the border between hierarchical layers is blurred. As described in [39], an activity can lose its motive and become an action and an action can become an operation when the goal changes [39]. This unclear border makes automated activity recognition difficult because the change of motive of activity and goal of action are not easy to detect. Hence, it is necessary to find clearer ways to determine each layer.

Secondly, tool and object are not distinguish in activity theory even though it is necessary to distinguish them because the same artifact item may be used as tool in an activity, but object in other activities. In this case, the item has different meaning for each activity. For example, when a dish is used as a tool for cooking, it implies it contains food. On the other hand, if it is an object for washing dish activity, it means that it is an empty dish.

In third, some activities are too diverse to be represented by a single activity name. For instance, eating has several similar activities such as having a meal or having breakfast, lunch or dinner. Because the top layer is activity in activity theory, the layer includes everything. This makes activity recognition system design cumbersome and difficult to conceptualize. This difference in component granularity is not conducive to sharing or modularizing activity recognition systems.

Lastly, it is difficult to represent a temporal relationship between activities in activity theory due to the fact that activity theory focuses on the relationship between components such as subject, activity objective, tool, and outcome rather than the relationship between activities.

2.2 Probabilistic Activity Models

In probabilistic activity recognition, it is assumes that human activities are continuously performed and each activity is a sequential composition of activity

components such as motions, operations or actions according to a temporal sequence [7]. According to this idea, several probabilistic models including Hidden Markov Model or the Conditional Random Field Model have been used to build an activity model because they are suitable for handling temporal sensor data.

2.2.1 Hidden Markov Model (HMM)

HMM is a probabilistic function of Markov chains based on the first order Markov assumption of transition [43]. The basic idea of Markov chain of order m is that the future state depends on the past m numbers of states. Therefore, for HMM that is based on the first order Markov assumption, the future state depends only on the current state, not on past states [43]. Also HMM is a model that is used for generating hidden states from observable data. HMM determines the hidden state sequence (y_1, y_2, ..., y_t) that corresponds to the observed sequence (x_1, x_2, ..., x_t) [7]. In activity recognition, hidden state is human activities; therefore, HMM recognizes activities from both sensor observation and recognized activities in previous time according to the first order Markov chain. Moreover, HMM is also a generative and directed graph model [7]. Generative model means that observation data is randomly generated. In other words, it should enumerate all possible random cases in the model. Directed graph is used to capture orders between states. Therefore, a generative and directed graph model in activity recognition implies it should find all possible sequences of sensor observations of an activity.

However, many activities may have non-deterministic natures in practice, where some steps of the activities may be performed in any order. In practice, because many activities are concurrent or interleaved with other activities, HMM has difficulty in representing multiple interacting activities (concurrent or interleaved) [7]. Also HMM is

incapable of capturing long-range or transitive dependencies of the observations due to its very strict independence assumptions on the observations. Therefore, enumerating all possible observation cases and orders is difficult for a practical activity recognition system. Furthermore, missing an observation or an order will cause the HMM to produce errors in the model.

2.2.2 Conditional Random Field (CRF)

CRF is a more flexible alternative to the HMM, which relaxes the strict assumption of HMM [24]. In other words, CRF solves the issues of HMM by neglecting the order constraint. Like HMM, CRF is also used to determine a hidden state transition from randomly generated observation sequences. However, CRF is a discriminative model, which does not generate all possible cases from the joint distribution of x and y. Therefore, CRF does not include arbitrarily complicated features of the observed variables into the model. Also, CRF is an undirected acyclic graph, flexibly capturing any relation between an observation variable and a hidden state [24]. Because CRF does not consider order, it considers only relationships such as state feature function (relationship between observations over a period of time and activities) and transition feature function (relationship between past activities and future activities). Even though CRF removes order constraint from an activity model, CRF could outperform HMM [33].

2.3 Knowledge based Activity Recognition

Some researchers have tried to utilize activity knowledge to achieve the accuracy required for recognizing dynamic human activities in real world. Different individuals may perform the same activity differently. Furthermore, a particular individual may perform the same activity differently at different times. To support this dynamic characteristic, activity model and activity recognition algorithm should be adjusted

dynamically. To address this issue, knowledge-based activity recognition approaches are developed [11,40] in which activity recognition system stores lots of activity knowledge including common sense, expertise knowledge, rules, or sensor information. Activity recognition systems use the knowledge to build an activity model. When new knowledge is found and added to the knowledgebase, the activity model is automatically modified and adjusted. Based on the activity model, activity recognition system recognizes an activity.

In [40], a conceptual model of daily living activities are built using ontology, which is described by a number of properties that specify relationships between activities and other entities such as action, object, or tool. The advantage of ontological activity model is that ontology language (OWL) is theoretically based on Description Logic (DL), which is a formal representation language of knowledge. Hence, the ontological activity model enables representation using DL. Activity recognition algorithm in [9] is developed based on DL reasoning, which supports several reasoning tasks such as subsumption, equivalence, disjointness, and consistency.

2.4 Performance Metrics and Measurement

Activity recognition performance needs to be measured and evaluated using adequate metrics. In this section, the metrics and measurements in literature will be reviewed. A measurement is a quantifiable attribute of an object such as program, product, or location. It is the raw data, which are associated with various elements of an object [55]. For example, people measure attributes (e.g. temperature or humidity) of an object (e.g. location). Metrics (or indicators) are computed from measures. They are quantifiable indices used to compare software products, processes, or projects or to predict their outcomes. Metrics are selected in measurement process [55]. In this

dissertation, we review several metrics to measure performance of activity model and activity recognition algorithm.

2.4.1 Certainty Factor

To compare the accuracy of the activity models, we measured the uncertainty incurred by each model under the same activity scenario. Certainty factor is very effective evaluative analysis used in several areas such as diagnostics or medicine [34,52]. We briefly define Certainty Factor below.

CF(H, E): CF is a certainty factor from hypothesis H influenced by evidence E [34]. The value of certainty factor ranges from -1(very uncertain) to +1(very certain) through zero (neutral).

$$CF(H, E)=MB(H, E) - MD(H, E) \tag{2-1}$$

MB(H, E): MB is the measure of increased belief in hypothesis H influenced by evidence E [34]. p(H) and 1-p(H) are the probabilities of that hypothesis being true or false respectively. p(H|E) is a probability of hypothesis given E. If the evidence, E, is very strong, p(H|E) will equal to 1 and p(H|E) - p(H) will be also close to 1 - p(H) and MB will be close to 1 and certainty factor will increase. On the other hand, if the evidence is very weak, then p(H|E) - p(H) is almost zero and the uncertainty remains about the same with MD (H, E). The function max is used to normalize the MB value positive (between 0 and 1).

$$MB(H, E) = \begin{cases} 1 & \text{if } p(H)=1 \\ \dfrac{\max\left(p(H|E),\ p(H)\right) - p(H)}{1-p(H)} & \text{otherwise} \end{cases} \tag{2-2}$$

MD(H, E): measure of increased disbelief on hypothesis H influenced by evidence E [34]. If the evidence, E, is very strong, p(H) – min(p(H|E), p(H)) will equal 0 and MD

will be 0. On the other hand, if the evidence is very weak, then p(H) – p(H|E) is almost p(H), and the uncertainty will be close to 1. The purpose of function min is to make the MD value positive.

$$MD(H, E)=\begin{cases} 1 & \text{if } p(H)=0 \\ \dfrac{p(H)-\max\left(p(H|E), p(H)\right)}{1-p(H)} & \text{otherwise} \end{cases} \quad (2\text{-}3)$$

Certainty Factors are combined according to the sign of their value as shown in Equation (2-4) [52].

$$CF(CF_1, CF_2)=\begin{cases} CF_1+CF_2\left(1-CF_1\right) & \text{if } \left(CF_1>0 \wedge CF_2>0\right) \\ CF_1+CF_2\left(1+CF_1\right) & \text{if } \left(CF_1<0 \wedge CF_2<0\right) \\ CF_1+\dfrac{CF_2}{\left(1-\min\left(\left|CF_1\right|,\left|CF_2\right|\right)\right)} & \text{if } \left(CF_1<0 \vee CF_2<0\right) \end{cases} \quad (2\text{-}4)$$

2.4.2 Metrics for Activity Recognition Performance

Metrics are used to compare the performance of AR approach/system. The metrics are accuracy, precision, recall, sensitivity, specificity, accuracy, time slice accuracy, and class accuracy. To compute metrics, it is measured the number of cases such as true positive, true negative, false positive, and false negative [10,62].

- True positive (TP): the number of correctly recognized activities.
- True negative (TN): the number of gaps between activities, which had no activities.
- False positive (FP): the number of activities that are not actually performed but recognized.
- False negative (FN): the number of activities that are not recognized even though they were actually performed.

Statistical metrics such as accuracy, recall, precision, and specificity are defined in Equation (2-5), (2-6), (2-7), and (2-8).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2-5}$$

$$\text{Recall (Sensitivity)} = \frac{TP}{TP + FN} \tag{2-6}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2-7}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{2-8}$$

We also used two statistical activity recognition accuracy measures defined in [8]: time slice accuracy and class accuracy.

$$\text{Timeslice} = \frac{\sum_{n=1}^{N} \left[\text{inferred}(n) = \text{true}(n) \right]}{N} \tag{2-9}$$

$$\text{Class} = \frac{1}{C} \sum_{c=1}^{N_c} \left\{ \frac{\sum_{n=1}^{N_c} \left[\text{inferred}_c(n) = \text{true}_c(n) \right]}{N_c} \right\} \tag{2-10}$$

In Equation (2-9) and (2-10), [a=b] is a binary indicator yielding 1 when true, 0 otherwise. N is the total number of time slices, C is the number of different activity types (classes) and Nc is the total number of time slices for class c [62]. Timeslice accuracy counts true positives within a time slice. It offers a more meaningful measure of recognition performance than global accuracy especially when some activity classes are more frequent than others [62]. Class accuracy averages time slice accuracy over all activity types, which is bound to report lower numbers than time slice but is more representative of actual model performance. HMM and CRF performance has been

compared in Table IV using the Timeslice and Class accuracy over the Amsterdam dataset.

2.5 Uncertainty Analysis

A measurement is an observation and measured value results in information. Uncertainty is related to the quality of the information. Since there is always a margin of doubt about any measurement, it is necessary to quantify uncertainty [82]. According to GUM (Guide to the Expression of Uncertainty in Measurement), which is published by ISO (the International Organization for Standardization), uncertainty is defined such that uncertainty of a measurement reflects the lack of exact knowledge of the value of the measurand. Uncertainty arises from random effects and from imperfect correction of the result for systematic effects. Even after correction for recognized systematic effects, the result of a measurement is only an estimate of the value of the measurand because it is not possible to remove uncertainty completely [82].

There are two types of uncertainty estimates: type A and type B [82][57]. The differences between them may be attributed to different uncertainty evaluation methods and different possibilities of reducing the uncertainties. Type A uncertainty estimation involves statistics. The source of Type A uncertainty is the nature of randomness of measurement. In other words, measurement values are different even though they are measured in the same environment. Some examples of type A uncertainty are temperature or humidity. Temperature or humidity value keeps changing according to time or location. This inherent randomness cannot be reduced by collecting more data. Therefore, type A uncertainty is also referred to as aleatory, stochastic, variability, or irreducible uncertainty [42].

Type A uncertainties are usually modeled with probability distributions. In Type A evaluations of measurement uncertainty, if statistically significant number of measurement data for X is collected, the distribution of X is close to a Gaussian distribution according to central limit theorem (CLT) [82]. X then has expectation equal to the average indication value and standard deviation equal to the standard deviation of the average.

Type B uncertainty is from systematically different results of measurement. Unlike Type A, which measures values repeatedly to find expected value, the measurement in Type B utilizes knowledge such as manufacturers' knowledge or experts' knowledge to determine whether the measured value is correct or erroneous. And the knowledge is represented using fixed values. For example, when waking up in the morning, if glucose level is [70 to 99 mg/dL], then the glucose level is considered normal according to diabetes experts' knowledge [58]. This knowledge is based on the fixed values of 70 and 99. However, even experts do not know the fixed value exactly sometimes. This lack of knowledge can be a source of Type B uncertainty. In addition to the lack of knowledge of measurement, there are several other causes of Type B uncertainty such as incomplete definition of the measurand, the limitation of measurement equipment, rounding of measured values, incorrect parameters for analyzing data [82]. It is possible to reduce these kinds of reasons by improving measurement system or knowledge. Therefore, Type B uncertainty is sometimes referred to as state of knowledge uncertainty, subjective uncertainty, epistemic uncertainty, or reducible uncertainty, meaning that the uncertainty can be reduced through increased understanding (research), or increased and more relevant data [42]. However, it is not possible to

remove Type B uncertainty completely because the knowledge for measurement cannot be known perfectly.

For an evaluation of Type B uncertainty, often the only available information is that X lies in a specified interval [a, b] with fixed value a and b. In such cases, knowledge of the quantity can be characterized by a rectangular probability distribution with limits a and b [82]. Type B uncertainty may or may not be modeled probabilistically. There are many ways of representing Type B uncertainty, including probability theory, fuzzy sets, possibility theory, Dempster-Shafer evidence theory, or imprecise probability [57].

It is important not to confuse the terms “error” and “uncertainty” [57]. Error is the difference between the measured value and the “true value” of the quantity being measured. Uncertainty is a quantification of the doubt about the measurement result. And uncertainty occurs while performing a measurement. When we recognize some errors, we may try to correct them by applying correction methods like calibration certificates. But some errors cannot be clearly recognized. An unrecognized error whose value is not known is a source of uncertainty. Like uncertainty, there are two types of errors: Type I error and Type II error. Type I error is false positive error, which rejects correct result. Type II error is false negative error, which accepts false result. We can remove error completely; this feature distinguishes it from uncertainty. However, uncertainties will always exist because it is not possible to remove them completely.

There has been a lot of research about uncertainty measurement in several areas. The measurement in healthcare needs high certainty due to its critical characteristic. In [37], it applies GUM method to medicine area and introduces a standard approach to measure uncertainties for scientists and engineers in medicine. It shows an example

how to make a model of measurement, identify and characterizes uncertainty components for radiation dose. Both type A and B uncertainty estimation approaches are used to measure uncertainties in medicine. Type A uncertainty is applied to evaluate uncertainties in observed data. Type B uncertainty evaluation approach is applied for uncertainties from “rounding” operation of values.

In [42], the author raised an issue that Type B uncertainty in engineering application is often treated incorrectly probabilistically as Type A uncertainty. With real example, it demonstrated three evaluation methods of Type B uncertainty such as interval analysis, Dempster-Shafer evidence theory, and second-order probability methods. The experiment results show how Type B uncertainty due to lack of knowledge affects to the measurement of the example.

In pervasive computing, the context of a smart space is recognized using sensor data. If the sensor data is uncertain, the context derived from the sensor data is also uncertain. In [58], an uncertainty model of context is introduced. It shows how to evaluate uncertainty of context in high level using uncertainty measurement result of sensor data in low level.

2.6 Activity Recognition Programming Model

Activity recognition programming model research is in its infancy. Currently, many research groups treat an activity as a component of context or situation. In [33], context driven programming model is employed to resolve the conflict between two contexts. For example, two actuator services “increase temperature” and “turn on the air conditioner” may be in conflict with each other and may not be initiated at the same time [33]. To solve this issue, Context knowledge about the physical world including human activity is utilized. In their approach, the physical world is represented explicitly using a

description logic language. Their programmable pervasive space is composed of four components—user, sensor observation data of a space, programming statements, and set of services.

Paper [27] presents a programming framework for robust context-aware applications. The framework is based on a forward recovery model, where exceptions are propagated to other sub modules that can deal with the exception better. The model consists of mechanisms for synchronous exception handling and asynchronous event handling. Exceptions and events are synchronous because exceptions occur when a particular undesired event takes place. Events are asynchronous because they occur whenever context satisfies particular condition and events do not wait for other events for synchronization. These two mechanisms enable programming recovery actions for handling different kinds of failure conditions such as service discovery failures, service binding failures, exceptions raised by a service, and context invalidations.

In [44], a programming model is developed for tasking applications on smart phones, which should run continuously, process data from sensors on the phone, determine user's current context such as location or activity using the sensor data and trigger certain services for the user according to the context. Due to the limitation of mobile devices, accuracy and energy efficiency are the main goals of the programming model described in [41]. To achieve the goals, the CITA (Code In The Air) framework was developed. The framework is composed of activity hierarchy, activity composition rules, and tasking framework. Activity composition rules support multi-phone activity prediction, i.e., a user's context may be determined the user's phone as well as phones belonging to other people.

2.7 Summary

This chapter provides a literature review on activity recognition, uncertainty, and activity recognition programming model. Firstly, it reviews well-known approaches for modeling and recognizing human activities. Secondly, uncertainty analysis research is described. Lastly, programming models based on contexts or situations are reviewed.

CHAPTER 3
STATEMENT OF PROBLEMS IN ACTIVITY RECOGNITION

As mentioned in the literature review in the previous chapter, state-of-the-art activity recognition (AR) technology suffers from several limitations. In this chapter, we will look at the problems in current activity recognition research. In activity recognition technology, there are several sub-research areas that are related to each other. Figure 3-1 presents the research areas and their relationships using layered structure.

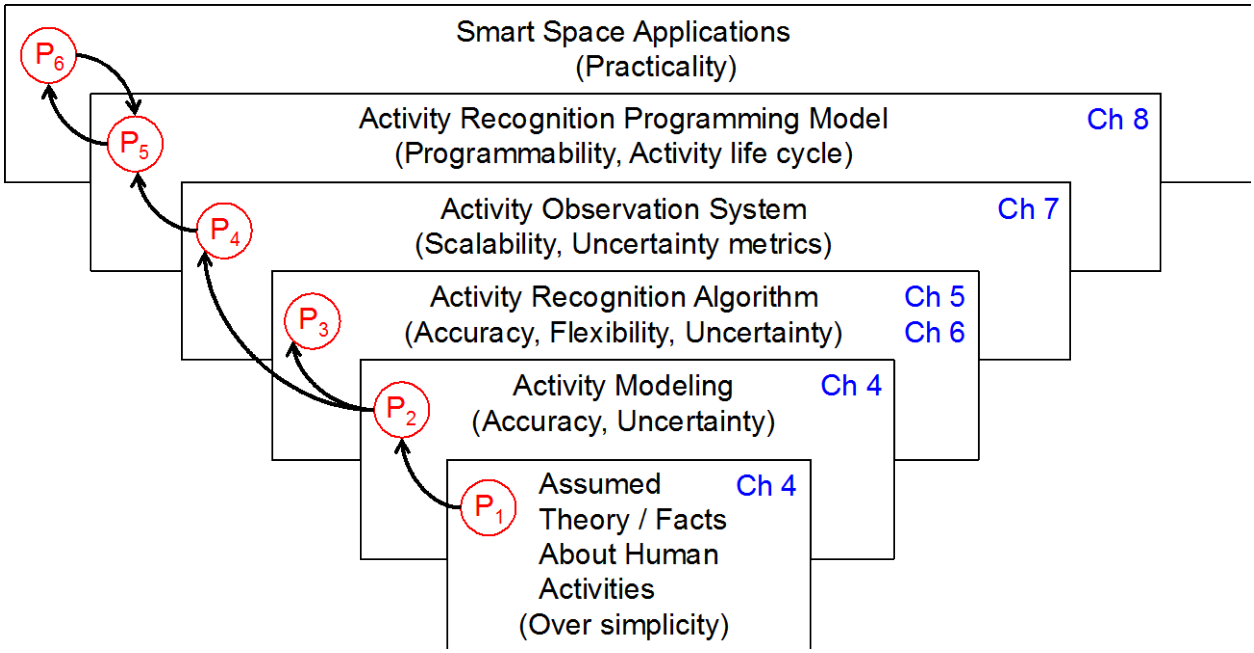


Figure 3-1. Problems (P_i) in activity recognition techniques and related chapters

A higher-level research area builds upon a lower level area; a problem (P_i) in the lower layer module propagates to the higher layer. For example, activity modeling is based on assumed theory and facts about human activities. Therefore, the limitations of the assumed theory affect activity modeling. To address a problem, it is necessary to address it in the lowest layer where it originates.

This dissertation discusses the problems in each layer in detail and proposes solutions in the following chapters as indicated in Figure 3-1.

Section 3.1 describes the limitations of assumed theory or facts about human activities. Section 3.2 discusses the problems in activity modeling, which is based on the assumed theory and facts about human activities. In Section 3.3, the issues with the relationship between activity recognition algorithm and activity model are discussed. Section 3.4 shows the challenges in activity sensing/observation system. Section 3.5 details the problems in activity recognition programming model.

3.1 Assumed Theory and Facts about Human Activities

Understanding the nature or characteristics of human activities is critical for activity recognition research. Therefore, many activity recognition researchers try to conceptualize human activities. Currently, human activity is popularly assumed based on activity theory or temporal sequence.

3.1.1 Problem of Activity Theory Based Assumption

Activity theory assumes that people use tools to fulfill their objectives [39]. This assumption is too simple to represent variety of human activities; therefore, activity theory does not provide an exact representation of real world human activities. Activity theory was developed by psychologists to understand human activities. Application of activity theory techniques for understanding human activities relied in the cognitive ability of psychologists [39,64]. Modern activity recognition techniques are however being employed to understand human activities automatically by intelligent computing systems. Since cognition by a computing system is very different from cognition by a human, it is necessary to revise activity theory, or at least develop a new activity framework to extend the theory, to better support computer-based recognition systems [10]. The new activity framework should capture human activities with the high precision that is adequate for automated activity recognition (AR). Such a framework will enable

the development of activity models that are more accurate than those based on traditional activity theory [10].

3.1.2 Problem of Probabilistic Assumption of Activity

In probabilistic activity recognition, it is assumed that human activities are continuously performed and each activity is composed of temporally sequential components such as motions, operations or actions [7]. However, this assumption is not practical because, unlike procedural instructions, humans may perform the components of an activity in a variety of ways. It is difficult for a computer-based system to foresee all possible sequences. Additionally, a human may perform two or more activity components in a concurrent or interleaved fashion. Probabilistic AR does not account for these complexities.

As presented in Figure 3-1, creating a comprehensive activity theory is critical because it is the groundwork for activity recognition. Therefore, a new activity framework is introduced in this dissertation and described in Chapter 4.

3.2 Activity Modeling

Activity model is an abstract and simplified conceptual representation of real world human activities. Currently, activity theory based activity models or probability based graphical activity models are commonly used [28,60,62,65]. However, both modeling techniques are not adequate for modeling real activities.

3.2.1 Problem of Activity Theory Based Model

Activity theory based model makes too simple assumptions and probability based activity model is too complicated. As discussed below, activity theory is “too naïve” and probability based modeling is “too strict” for real world activities.

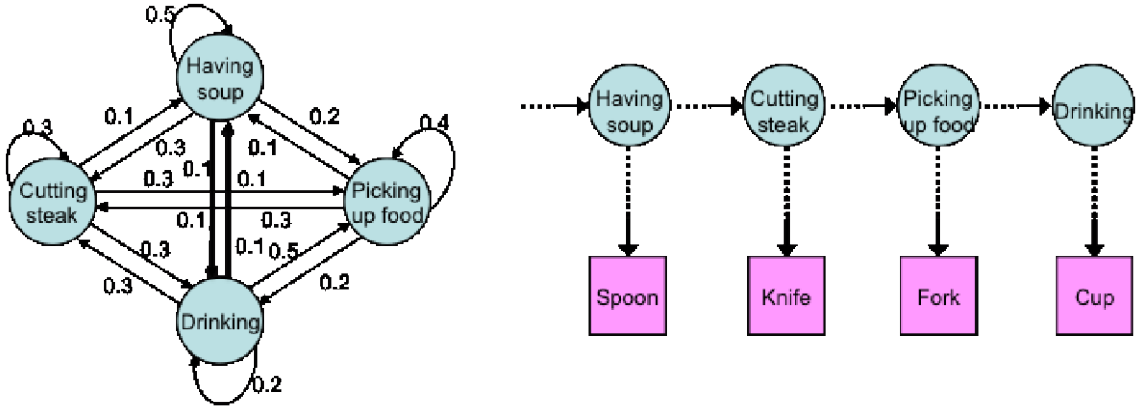
Activity theory based activity model inherits the problems of activity theory. Activity theory defines four components (subject, tool, objective, and outcome) [39,64]. However, components such as time, location, or object, which are also important for characterizing activities, are not incorporated by activity theory. Therefore, an activity theory based model may miss some important components, and is therefore considered a naïve model. For instance, an activity theory based model does not distinguish between tool and object because there is no “object” in activity theory. These two parameters, however, need to be distinguished given the same item may be used as a tool or object. For example, when a dish is used as a tool, it implies it contains food for “serving”, “eating” or “cooking” activities. On the other hand, a dish may also be an object for “dishwashing” activity.

3.2.2 Problem of Probability Based Model

Probability based activity model can be applied to real world problems because it utilizes a database of all possible cases of performing activities. Hidden Markov Model (HMM) and the Conditional Random Field (CRF) are amongst the most popular modeling techniques as mentioned in Chapter 2.

Simple activities can be modeled accurately using HMM. However, complex activities are difficult to create model. A HMM determines the hidden state sequence (human activities) that corresponds to the observed sequence (sensor observation) and previously determined hidden state. HMM recognizes activities from both sensor observation and the previous activity in accordance with the first order Markov chain. The HMM is a generative, directed graph model [36] that should find all possible sequences of observations. Many activities may be performed in any order. Therefore, enumerating all possible observation cases and orders is difficult for a practical system.

Furthermore, missing an observation or an order will cause the HMM to produce errors in the model. Figure 3-2 shows an example of HMM model for eating activity. In the figure, the hidden states are presented with circles and rectangles are sensor observations.



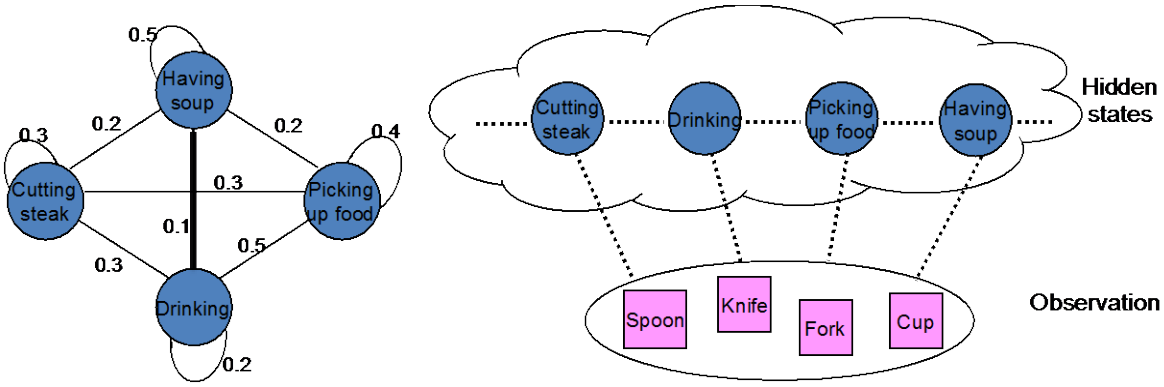
(a) An HMM of eating (b) Observation sequence of a eating activity

Figure 3-2. Example of HMM for eating activity

Conditional Random Field (CRF) is more flexible than HMM because it relaxes the strict order assumptions of HMM [24]. Similar to HMM, CRF also determines a hidden state transition from observation sequences as shown in Figure 3-3. However, a CRF is a discriminative and an undirected acyclic graph, flexibly capturing any relation between an observation variable and a hidden state [24]. Given that CRF does not consider order, it considers only relationships such as state feature function (e.g., relationship between observations over a period of time and activities) and transition feature functions (e.g., relationship between past activities and future activities). Therefore, lines in Figure 3-3 represent the relationship between activities.

Even though CRF removes order constraint from an activity model, it has been shown to be capable for outperforming HMM [62]. However, CRF requires finding all possible relationships between activities. This is not practical because human activities

may not always be in accordance with the typical relationships assumed by CRF. For example, “washing dishes” usually follows “eating", but sometimes activities such as “watching TV”, “talking”, “going to gym”, etc. may be performed right after “eating”. If an activity model does not enumerate all possible activities that can come before or after “eating”, the missing relationships will cause errors in activity recognition.



(a) CRF states of eating (b) State transition and observation of eating

Figure 3-3. Example of CRF for eating activity

The limitations of the algorithms mentioned above necessitate the development of a new activity modeling technique. The new activity modeling technique should satisfy two conditions. Firstly, it should represent characteristics of human activity accurately. Secondly, it should be feasible to utilize in computing based activity recognition. To address these requirements, a new Generic Activity Framework (GAF), based on activity Semantic knowledge (SGAF), is introduced in Chapter 4.

3.3 Activity Recognition Algorithm

Activity Recognition (AR) algorithm interprets sensor data based on the activity model; therefore, it is closely related to the activity model. If an activity model is changed, AR algorithm should also be modified accordingly. Therefore, as shown in Figure 3-1, AR algorithm inherits the problems in activity model. As mentioned in Section 3.2, a new activity modeling technique that is based on generic activity frame

and activity semantic knowledge is introduced. Therefore, our AR algorithms are based on this novel modeling technique.

In this dissertation, two AR algorithms are developed. In Chapter 5, Multi Layer Neural Network (MLNNK) based AR algorithm is introduced. MLNNK based AR algorithm supports generic activity framework. The generic activity framework has multi layer structure and represents composition relationship between activity and activity components. Due to the problems mentioned above, GAF excludes order and activity relationship. Also GAF does include order or activity relationship in it because it consider only compositional structure, which shows the relationship between activity and activity components. Therefore, MLNNK can support GAF better than HMM and CRF.

Fuzzy logic based AR algorithm based on activity semantic knowledge is introduced in Chapter 6. This approach accounts for the ambiguous nature of activities and overcomes several limitations of current technology. Presently, AR algorithms assume that an activity is either performed completely or not performed at all; they do not account for incompletely performed activities. However, some activities in real world may be incompletely or ambiguously performed. For example, a person may only have a small amount (e.g., 20% of his dietary requirement) of food for breakfast. Fuzzy logic can be used to represent the fact that only 20% food was consumed. Since, probability based methods can represent 20% as well, a clarification is in order. In probability-based methods, 20% implies that the person had 20 breakfasts in 100 days, and skipped breakfast in the remaining 80 days [9]. Probability based activity recognition algorithm is unable to account for incomplete meals.

Activity knowledge may also be fuzzy. For example, the activity knowledge, “sleeping is an exclusive activity because people can’t do anything while they are deeply sleeping”, usually holds true. However, “deeply” is a linguistic word that has fuzzy interpretation.

Moreover, activity observation system produces fuzzy activity data. Ideally, sensors should work properly and provide clear indications about performed activities. However, because of several reasons such as sensor failure, weak sensor signal or power etc., sensor data is sometimes vague and it affects activity recognition performance as shown in Figure 3-1.

Therefore, in the fuzzy situation mentioned above, it is necessary to account for uncertainties. One possible way is utilizing inference rules, which can make up for the limitations of insufficient evidential data. In fuzzy logic based methods, it is possible to use such inference rules.

3.4 Activity Observation System

Activity Observation System (AOS) utilizes several sensors and smart devices to detect human activities. Building an environment that can sense human activities in real world requires significant effort and cost. Problems in an activity observation system can be traced back to two sources—activity models and sensor technology.

Firstly, an AOS system inherits problems from the activity model because AOS system is designed based on the activity model, and sensors are selected in accordance with the activity model. For example, if “eating” is composed of “eating motions” and “eating tools” in an activity model, sensors will be chosen to observe the motions and tools. If “eating” is detected using “eating sound”, the sensor set will include acoustic sensor devices. Therefore, the limitations of activity modeling affect

AOS. Secondly, activity observation system produces uncertain observation data for activities. Even for a well-designed sensor system, some activity data may be missing or erroneous because of several reasons such as sensor failure or limited sensor lifetime. Therefore, it is necessary to develop a method that can overcome these problems.

In Chapter 6, a fuzzy logic based AR algorithm is introduced which tolerate the uncertainties in activity data. Uncertainty in activity observation system is analyzed in Chapter 7.

3.5 Activity Recognition Programming Model

AR programming model is important for developing a practical AR technology because it provides an interface for developers to utilize AR technology in their application. As shown in Figure 3-1, AR programming model inherits problems from several areas including activity modeling, activity observation system, or application requirement.

Firstly, activity recognition system classifies activities from sensor data according to an activity model and an activity recognition algorithm. However, arriving at a good model is an iterative process. Programmers may be provided a mechanism that allows them to easily change the model and activity knowledge independent of other AR sub systems. Existing approaches do not allow for such isolated changes to be made to the model. The slightest change usually requires careful examination of the entire model and many laborious changes and updates.

Secondly, an activity observation system may frequently need addition or replacement of sensors during system upgrade or modification (e.g., due to availability of a better or more accurate sensor). Observation system programmers therefore need

scalability to incorporate such new sensors. Also, sensors have a finite lifetime. Hence, programmers of activity observation systems should be able to replace worn out or damaged sensors. Such sensor changes are very costly in most activity recognition systems because the sensing sub-system is incorporated into and an integral part of the activity model itself.

Moreover, AR programming model should provide a programmable interface to application developers so that they can easily utilize activity recognition technology without spending much effort towards understanding the details of the technology. Some application developers may require specific programmable interface for their application. For example, if an application requires a high assurance mechanism for recognized activities, AR programming model should support these demands.

In Chapter 8, AR programming model for these three cases is discussed. In the developed approach, we utilize activity knowledge and sensor or device knowledge to achieve high programmability.

CHAPTER 4
ACTIVITY SEMANTICS AND GENERIC ACTIVITY FRAMEWORK

Creating activity frameworks is a prerequisite for enabling AR algorithms and systems. To illustrate, if activities are recognized through artifacts that a person uses, an artifact-usage based activity framework must first be constructed so that the AR system can recognize activities based on their relationships with the artifacts. In this Chapter, a new framework for more accurate activity modeling is introduced.

Next section explains the goal of the generic activity framework and activity semantics. Section 4.2 introduces activity generic activity framework. Section 4.3 describes activity semantics. Section 4.4 shows a case study of the SGAF based activity modeling technique. A comparison and analysis of the introduced activity model with other models in terms of attainable recognition certainty are presented in Section 4.5.

4.1 Goal: Increased Accuracy in Activity Model

Enhancing accuracy of activity model is important because activity recognition (AR) performance is significantly dependent on the accuracy of the underlying activity model. Activity framework should capture human activities in the real world with the high precision. Then, activity models based on the new framework will be more accurate because they inherit the advantages of the new activity framework. To achieve this goal, a new activity framework, which is a refined hierarchical composition structure, is developed.

In addition to the compositional structure in activity framework, it is essential to examine and develop an activity model that can capture and represent the dynamic nature of human activities precisely. To address this issue, a knowledge-assisted

activity modeling technique, which is comprised of activity Semantics and a Generic Activity Framework (SGAF) is developed. Activity semantics are highly evidential knowledge that can identify an activity more accurately in ambiguous situations.

4.2 Generic Activity Framework (GAF)

The generic activity framework has a hierarchical structure. The hierarchical structure has several advantages. Firstly, it makes the activity recognition system more tolerant to sensor environment change. For instance, even if more sensors are inserted in the AR system, the upper layers in the hierarchy will not be seriously influenced. In other words, the additional sensors will cause changes to the motion, tool, and operation layers. But, the operation layer will not be affected directly from the sensor change. Secondly, activity recognition using hierarchical structure is analogous to the way people recognize, so it is easier to design more natural and intuitive AR algorithm. To illustrate, when people observe an event, they accumulate it, and compose the unit observations until they find what it is.

Each layer of the structure consists of activity components. In total, there are eight components in the generic activity framework. We chose the eight components according to 5W1H framework and we also found the eight is the most influential. 5W1H is well known method to represent knowledge for many applications including web or newspaper because of its capacity to describe knowledge. It was created by Rudyard Kipling, the Nobel Laureate of Literature in 1906 [17]. 5W1H framework deals with six keywords (“who”, “what”, “where”, “when”, “why”, and “how”). We split “how” into two components (tool and order) for more detailed information. We also add context because it is important to understand activities. It is not necessary that every activity contains all eight components as long as the activity is recognized clearly. For example,

the walking activity does not require any object. Also some components such as motive are difficult to know. The detailed description of the eight components is given below [9]:

Subject. A subject is an actor of the activity. Subject has an important role as an activity classifier especially when there are multiple people. In other words, a different subject means a different activity.

Time. This is the time when an activity is performed. It consists of start time and end time. We can also calculate the duration of an activity with time. Time and duration are useful to recognize more detailed activities. For example, if an eating activity is performed around noon; it is having lunch instead of just eating. Also depending on the duration of the eating, we can classify the eating to having a meal or having a snack.

Location. Location is the place where an activity is performed. If an activity is performed in several places, location will have multiple values. Location is also a crucial classifier of activities. In other words, since location has a particular function such as sleeping or cooking, it gives important information for recognizing activities.

Motive. Motive is the reason why a subject performs a specific activity. Motive is the objective in activity theory as shown in Figure 2-1. To determine motive, some artificial intelligent reasoning technique may be required.

Tool. Tool is an artifact that a subject uses to perform an activity. Tool provides essential information to classify activities. For example, a spoon or a fork is a tool for eating or cooking. Therefore, an AR system can expect those activities when it detects that a user uses a spoon or fork.

Object (Target). An object can also be any artifact like tool. However, object is the target of an activity while as a tool is used by a subject. Distinction between tool and object is important for accurate activity recognition because some artifacts are both tool and object depending on an activity.

Context. Context is information, which is used to determine the situation where an activity is performed. Installed sensors directly sense some contexts such as temperature or humidity. On the other hand, some contexts like motive of activity need some artificial intelligent techniques such as reasoning or pattern recognition to elicit them. Figure 4-1 shows a composition diagram of the generic activity framework [9].

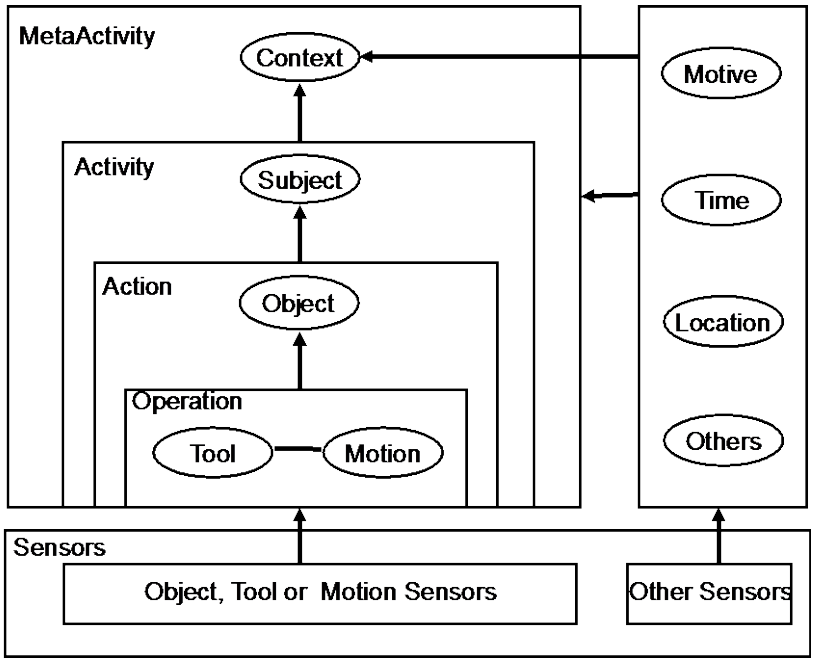


Figure 4-1. Composition diagram of a generic activity framework

In Figure 4-1, rectangles are layers and ellipses present components. According to the composition of components, the activity framework has a hierarchical structure that is similar to activity theory. Activity theory has three layers—operation, action, and activity. In our generic activity framework, meta activity and classified meta activity

layers are added. Also it clearly defines the components of each layer. The detailed description for each layer is given below [9,11]:

Sensors. Sensors are installed in the pervasive space (e.g. a smart home) to collect event information of the space. Based on the source of sensor data, sensor is classified into four types: motion, tool, object, and context sensor. A motion sensor is about peoples' movements such as raising an arm or turning body. Nowadays, gyro sensors or accelerometer sensors are used to sense human motion. Tool sensor data is from sensors attached to equipment, which is used by people. Object sensor data is from sensors installed on passive objects such as groceries or frozen food packets [1]. RFID readers and tags are representative sensors to recognize tools and objects. Sound sensors, vibration sensors or pressure sensors are also used to recognize activities.

Operation. Operation is a composition of tool and motion. The user operates tools with specific motion. For example, if computer is a tool, some hand or finger motion will be performed for typing a keyboard.

Action. Action is determined by combination of operation and object. For example, if a user types a command to open a file, typing on the keyboard is an operation and the file is an object and this combination is open file action. Object is important to recognize actions. However, some actions do not have an object. For instance, in sleeping activity, the action is lying down on the bed. A bed is a tool for sleeping, but no object is involved in sleeping. Moreover, some activities like walking do not require tools. Therefore, although tool and object are important, they are not mandatory components.

Activity. Activity is a collection of actions. Activity may involve multiple actions that occur in a certain order. For example, for laundry activity, we should put clothes into the washer first. Otherwise, it is difficult to say it is laundry activity even though the washer runs. But for many activities, the order of actions varies a lot according to the user. For example, there are several actions such as scooping, picking, and cutting food for eating. The order of these actions is totally up to a person. Therefore, we consider the relationship between activity and action unless the order is critical for recognizing activity.

Meta activity. A meta activity is a collection of activities. When an activity is complicated, it is composed of several simple activities. For example, to prepare a meal, people obtain food material from refrigerator and wash, cut, or chop the materials. Sometimes, microwave is used, and people may also wash dishes to serve food. In this example, preparing a meal is a meta activity. Other activities such as preparing food material, washing, cooking, and microwaving are simple activities.

4.3 Semantics – Enriched GAF (SGAF)

Even though a generic activity framework in Section 4.2.1 describes the composition hierarchy of activity components, it is a general framework, which does not contain detailed activity semantic knowledge such as role of an activity component, constraint or relationship with other components [11]. The activity semantics should be represented in an activity model because they are important for classifying an activity. For example, eating is composed of three actions such as picking food, chewing food, and swallowing food. In this case, if only picking food and chewing food are detected, then it is not clear whether we consider eating is really performed or not because we are not sure the person completes the activity through chews and swallows the food or not.

Activity semantics reduce these kinds of ambiguity. There are three activity semantics: dominance semantics, mutuality semantics and order semantics [11].

4.3.1 Dominance Semantics

This is semantic information of vertical relationship between components in upper layer and lower layer in Figure 4-1 (e.g. meta activity and activity, activity and action, or action and operation). In other words, components in upper layer (e.g. activity) are composed of the components in lower layer (e.g. action). In this hierarchical composition structure, the contribution of each action is different. Even though some actions are components of the same activity, some actions are dominantly essential component of the activity whereas some are not. According to the dominance, we classify them as key or optional components [11].

Key component. Key component is a mandatory component for identifying an activity. If an activity has multiple key components, all of them are required to agree with the activity. Otherwise, the activity is not considered performed. To illustrate, swallowing is a key action for eating because if people don't swallow food, it is not regarded eating is completely performed even though there are many other actions such as picking food, scooping food or chewing food.

Optional component. If a component is neither a key nor unique component, it is an optional component. It is possible to omit an optional component because it does not always affect activity classification. For example, cutting food is an action of eating but it may be omitted depending on the food. However, if optional component is detected, it increases the certainty of the recognition of an activity.

4.3.2 Mutuality Semantics

This is semantic information of horizontal relationship between components at the same layer (e.g. meta activity and meta activity, activity and activity, action and action or etc). This semantic knowledge is used to determine whether multiple activities can be concurrently performed or not.

Concurrent component. If two or more components are performed together, they are in concurrent relationship. For example, laundry or watching TV is concurrent because while the washer is running, the user can watch TV at the same time.

Exclusive component. If an activity cannot be performed simultaneously with another activity, it is an exclusive activity. For example, sleeping is an exclusive activity because people cannot perform anything when they sleep.

Ordinary component. Ordinary components are partially exclusive and concurrent. If an activity is performed with a part of the body (e.g. human limb), the activity is both concurrent and exclusive. For example, when people eat food, they cannot sing a song at the same time. In this case, they are exclusive. But if the people take a walk, they can sing a song concurrently. Therefore, sing a song is both partially exclusive and concurrent.

4.3.3 Order Semantics

Some activities like an instruction should follow a procedural sequence. However, many activities have flexible order or do not have any order. Therefore, the role of order among activity components should be considered depending on the activity.

No order. There is no specific order required between activity components. For example, actions for eating such as cutting, picking, and scooping food does not have any order restriction.

Strong order. Some activity requires that activity components (e.g. actions) should be performed in a specific order always. For instance, in case of sleeping and waking-up, waking-up comes immediately after sleeping because people perform another activity before waking up.

Weak order. For many activities, their action components are performed according to a flexible order, which is not mandatory or strict. For example, usually eating is performed after cooking, but there are exceptions to this order depending on several situations.

Skip chain order. When an activity is interleaved, other activity components may be performed between two ordered activity components. To illustrate, eating is usually performed immediately after cooking, but sometimes we can do other activities between them.

4.3.4 Effect Semantics

Effect is caused by an activity. For example, the eating activity will have effects such as “increasing glucose level” or “increasing body temperature”. Activity effect takes time to be shown, and this temporal gap between activity and effect varies depending on activity. For instance, sound effect of an activity happens almost immediately when the activity is performed whereas some effect like temperature or glucose level appears after a period of time. According to the temporal gap, T, between an activity and the effects of the activity, effect recognition has different algorithms.

T = 0 or T ≈ 0. if there is no gap, the effect should be ready for detection. For example, a chewing sound is an effect of the eating activity. It happens immediately after a person starts to chew food.

T > 0. If there is a gap between activity and effect, when the activity is recognized, effect recognition is triggered and related situations are also collected.

4.3.5 Activity Life Cycle

A human activity has a finite life cycle, which spans from its initiation to its termination. An activity is usually initiated in response to a requirement or a stimulus. The termination of an activity is usually close to or coincides with fulfilling a certain goal. Activity life cycle (ALC) information can improve activity recognition accuracy and enhance the utility of the recognition results. As shown in Figure 4-2, ALC is composed of four stages: “starting”, “growing”, “declining”, and “finishing”.

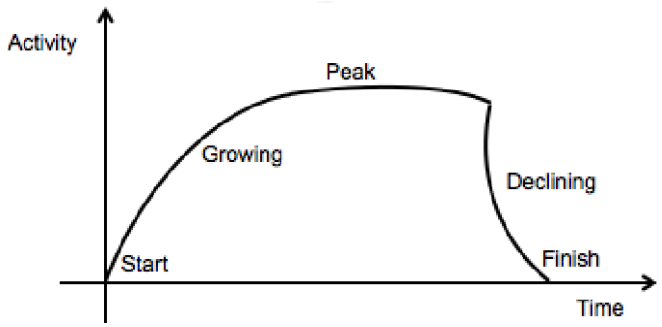


Figure 4-2. Activity life cycle stages

In Figure 4-2, recognition results may be tentative at the early stage of an activity’s execution. When an activity starts, there may not be sufficient sensor data for making decisive conclusions. If the collected sensor data is related to multiple activities, it may be difficult to clearly identify the performed activity. For example, if it is observed that a person holds a cup, it is not clear why the person holds the cup even though it is obvious the person may be performing an activity. This ambiguous start is called tentative start and recognition decision is postponed until more sensor data is collected.

Figure 4-3 shows all possible state transitions of ALC based on fuzzy values and other system parameters.

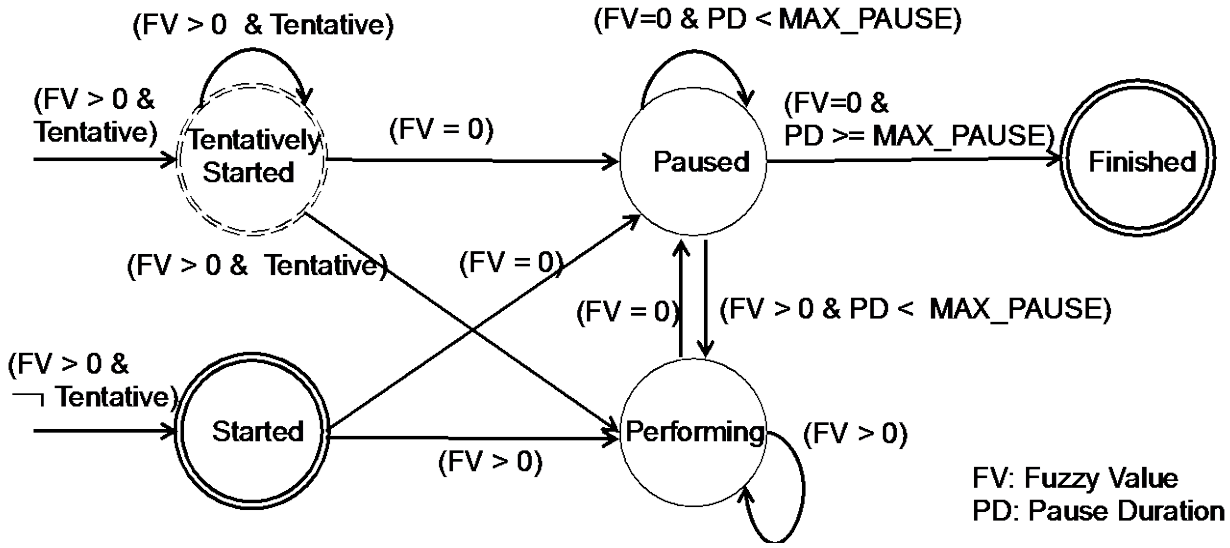


Figure 4-3. Activity life cycle state transition diagram

ALC stages can be captured by a finite set of five states as explained below:

Started. An activity is considered "Started" if the Fuzzy Value (FV) of the activity becomes greater than zero (0→ positive) and it is the only activity recognized from a sensor set.

Tentatively Started. When multiple activities show their fuzzy value change from zero to positive from a sensor set, it is difficult to determine which activity is actually performed. This ambiguous start is referred to as "Tentatively Started". Activity life cycle will stay in this state until the system can recognize which activity is really being performed.

Paused. While performing an activity, sometimes the human actor pauses the activity. For example, while cooking, if a friend calls, then the human actor may stop cooking and resumes the activity after finishing talking with the friend. In this case, the cooking activity should be considered single activity instead of two cooking activities.

Performing. After an activity starts it will stay in "Performing" stage if the fuzzy value of the activity is positive.

Finished. After the computed fuzzy value of an activity returns to zero for a certain period of time, the activity is considered to be “Finished”.

4.4 Case Study of Activity Modeling based on SGAF

In this section, an example of modeling activities using generic activity framework and activity semantics is given. First, modeling notations of activity components and activity semantics are created. Thereafter, an activity model using the notations according to the generic activity framework is designed. Figure 4-4 shows the modeling notations of each semantic. Dominance semantics and mutuality semantics are represented as nodes whereas order semantics are represented as edges. A component can have multiple semantics. For example, if a component is unique and exclusive, it is represented with both bold dotted line and filled circle. Also we can find that there are multiple elements like “x,y,z” in some circles whereas there is only “x” in other circles in Figure 4-4. The circle that contains multiple elements is a compound component and the circle with an element is an elementary component.

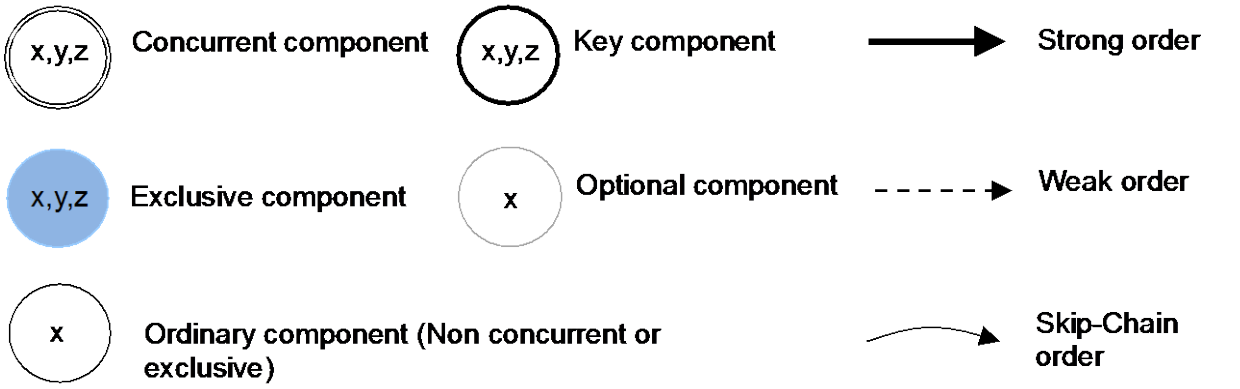


Figure 4-4. Notations of semantic components.

Figure 4-5 is an example of semantic activity modeling of daily living activities. The daily living activities are identified fro m a daily living scenario described in [6].

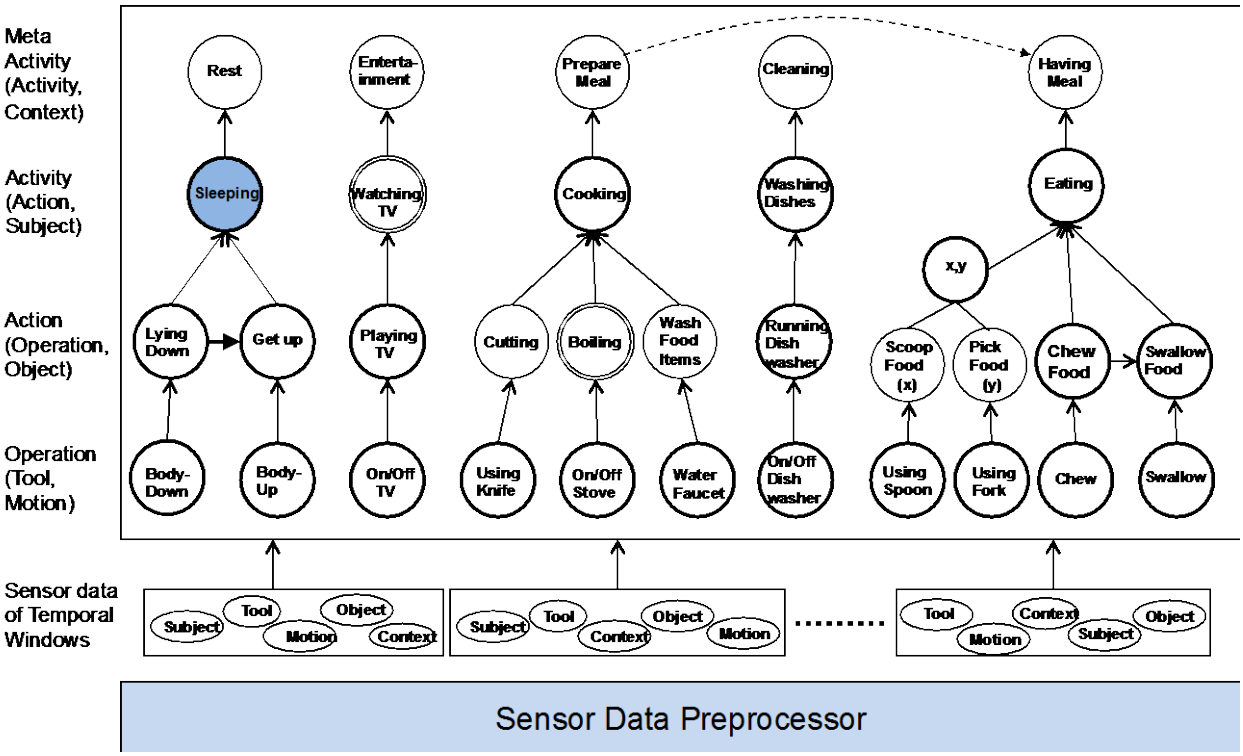


Figure 4-5. An example of semantic activity modeling of daily living activities.

In this modeling example, sleeping is a unique activity and it also exclusive activity where as watching TV is a concurrent activity. Scooping and picking are compound key components of eating. There is a skip chain relationship between preparing meal and having a meal because it can be interleaved by other activities.

4.5 Validation of SGAF Activity Modeling Approaches

In this section, we compare the developed Semantics and Generic Activity Framework based Model (SGAM) with a Traditional Activity Model based on activity theory (TAM) and the Generic Activity Framework based Activity Model without semantics (GAM).

4.5.1 Validation Scenario

A daily living activity scenario is used for a validation. To establish an activity scenario for the comparison, we used the eldercare scenarios of daily living described in [6] instantiated with a real activity dataset provided by University of Amsterdam [62].

The eldercare scenarios in [6] describe 33 different daily living activities of the elderly. The activity data set in [62] is records of activities of daily living performed by a man living in a three-bedroom apartment for 28 days. We named this dataset the Amsterdam dataset. We chose seven activities, which are common in both [6] and [62]. In terms of sensors, we assume the same sensor environment with the Amsterdam dataset are used to see how SGAM performs in real situations. We add a Bed sensor according to the scenario in [62]. Table 4-1 lists the seven activities and their components. It shows meta activities, activities, actions, operation tools, objects and related semantics for each activity [11].

Table 4-1. Activity list collected from the Amsterdam dataset.

Meta Activity	Activities (Location)	Action Operation	Tool	Object	Semantics
Rest	Sleeping (Bedroom)	-Going to the bedroom -Lying down	 Bed	Bedroom door	Mutuality: Exclusive activity
Hygiene	Taking a bath (Bathroom)	-Taking a shower -Washing face		Bathroom door Restroom door	Mutuality: Exclusive activity
	Using the toilet (Restroom)	-Opening a restroom door -Pressing a toilet flush		-Restroom door -Toilet flush	Dominance: -Key: Toilet flush -Unique: Restroom door
Preparing a meal -Breakfast -Lunch -Dinner	Cooking (Kitchen)	-Preparing food items -Heating food	 -Microwave -Pan	-Groceries -Refrigerator -Freezer	Weak order: -Taking food items -> Heating food
Drinking	Drinking	Taking drink	Cup	Refrigerator	Dominance: Key: Cup
Cleaning	Washing dishes		Dishwasher	-Pans -Cup -Dishes	Mutuality: Concurrent activity
Going out	Leaving the house	Opening front door		Front door	Dominance: -Key: Front door Mutuality: -Exclusive activity

Our activity scenario consists of six meta activities, seven activities, eleven actions, five tools, and eleven objects. It also shows the activity semantics of each activity. In Table 4-1, we can see that many artifacts are used as tools or object in activities. Especially when an artifact is used in several activities, TAM is difficult to recognize activities accurately because it regards artifacts as tools only.

For example, in Table 4-1, an artifact pan is a tool for cooking and it is also an object for washing dishes. Since TAM does not distinguish tool and object, sensing pan is not sufficient to determine which activity is performed. In contrast, GAM and SGAM consider the usage of artifacts as both tool and object. Especially, SGAM can recognize activities more accurately because it classifies activities using activity semantics. For example, sensing tools such as pan or microwave and objects like groceries usually mean that cooking is performed in GAM model. However, food items should be prepared before turning the microwave on if it is a cooking. Otherwise, it is unlikely the microwave is for cooking. GAM does not check this order semantic that is necessary for accurate activity recognition.

4.5.2 Comparison and Analysis

To find MB(H, E) and MD(H, E), the probabilities of hypothesis p(H) and the conditional probability p(H|E) need to be determined. For calculating the probabilities, we enumerate 77 possible cases based on Table 4-1. For example, to find the probabilities for sleeping, we found 6 possible cases with 2 components (bedroom door and bed) and one semantic (if the activity is exclusively performed or not). Then the number of all possible cases is three (detecting bedroom door, bed, and both bedroom door and bed) for each semantic case. In TAM, bedroom door and bed are equally treated as artifacts. In GAM, bedroom is an object and bed is a tool for lying down.

SGAM adds semantic information in the GAM model. We counted activities for each of the evidences. Table 4-2 shows an example of Sleeping [11]. Sum of probability is calculated using the addition law of probability that is the probability of A or B is the sum of the probabilities of A and B, minus the probability of both A and B. The probabilities of other activities are calculated similarly.

Table 4-2. The probabilities of sleeping activity according to the models.

	Evidence (E)	p(H and E)	p(E)	p(H|E)	Sum of p(H|E)
TAM	Artifacts	2	6	0.33	0.33
GAM	Tool	1	4	0.25	0.25
	Object	1	4	0.25	0.44
SGAM	Tool	1	4	0.25	0.25
	Object	1	4	0.25	0.44
	Semantics	2	7	0.29	0.63

Table 4-3 represents the conditional probability of hypothesis H given E for every activity [11]. We can observe some semantics are highly evidential whereas some are not according to how much the semantic contribute for identifying activities. However, SGAM has higher probability overall because it is based on GAM.

Table 4-3. The probabilities of hypothesis H given evidence E for each model.

Activity	TAM	GAM (Tool)	GAM (Object)	GAM (Tool & Object)	Semantics	SGAM (GAM & Semantics)
Sleeping	0.33	0.25	0.25	0.44	0.29	0.60
Taking a bath	0.38	0.00	0.38	0.38	0.43	0.64
Using the toilet	0.29	0.00	0.40	0.40	1.00	1.00
Cooking	0.35	0.40	0.40	0.64	0.72	0.90
Drinking	0.06	0.25	0.04	0.28	0.33	0.52
Washing dishes	0.16	0.88	0.19	0.90	0.21	0.92
Leaving the house	0.5	0.00	0.50	0.50	0.14	0.57

Using the estimated probabilities, we computed the certainty factor. Figure 4-6 shows the certainty factor for each activity. Figure 4-6 shows that SGAM has higher certainty for all activities and GAM model has higher or comparable certainty to that of TAM [11]. This is an obvious result because SGAM and GAM models provide more evidence than TAM and GAM models respectively. If no tool is used for an activity like

using the toilet or leaving the house, GAF and TAM show comparable certainty. We also can see that cooking, drinking, and washing dishes in TAM have low certainty compared to other activities because their tools or objects have low evidential certainty. The low evidential certainty may be attributed to artifacts such as a pan or a cup being used in multiple activities.

Also, we can observe that the certainty of SGAM for using the toilet and cooking have significant difference with other models. It is because the semantic information for using the toilet and cooking are more activity centric compared to other activities. For example, mutuality semantic is applied for several activities such as sleeping, taking a bath or leaving the house. On the other hand, the order semantic for cooking is only for the cooking activity and it is not applied for another activity of the scenario in Table 4-1. Therefore, the semantic is highly evidential.

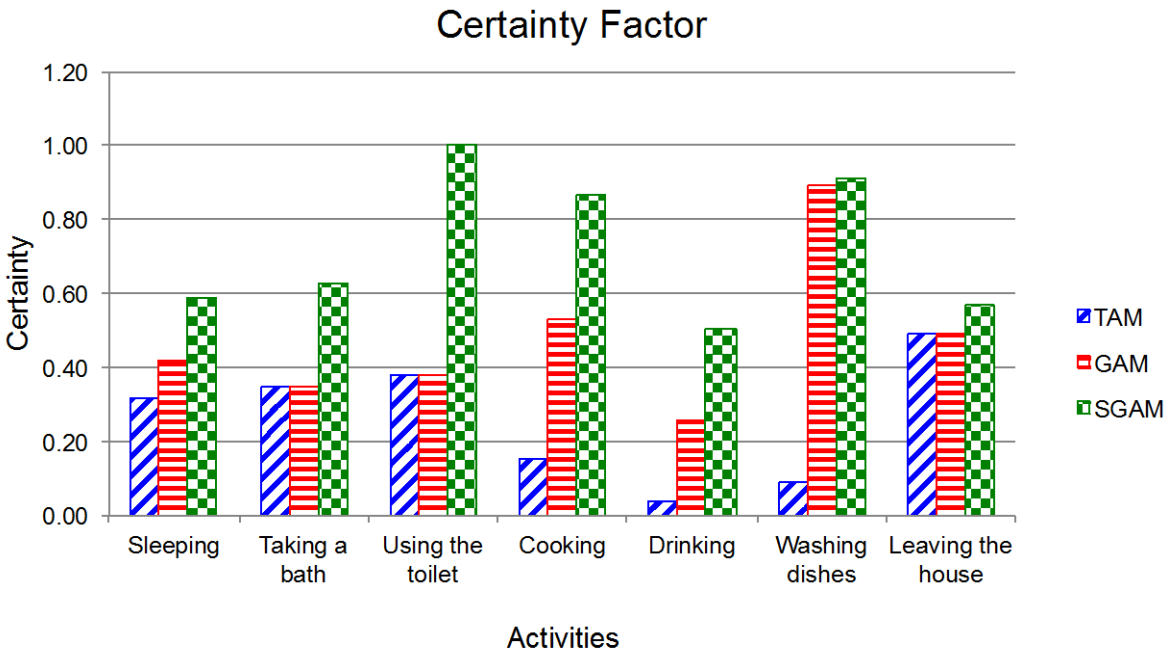


Figure 4-6. Uncertainty according to activities. It compares uncertainties of TAM, GAM, and SGAM for each activity.

4.6 Summary and Discussion

Activity recognition (AR) accuracy will be improved by enhancing the accuracy of activity model. However, accurate activity modeling is very challenging due to the diverse and dynamic nature of human activities. To address the challenges, a new activity modeling technique, which is based on generic activity framework and activity semantic knowledge is introduced. The generic activity framework is a refinement of the classical activity theory. The developed approach adds meaningful semantic knowledge to the generic activity framework for representing activities more accurately. A major advantage of the approach is that it can represent real world activities accurately by using the eight components of the generic activity framework along with the activity semantics introduced in this dissertation. Validation result shows that the developed modeling technique is higher certainty than other activity models, which are based on activity theory (TAM) or the Generic Activity Framework without semantics (GAM).

CHAPTER 5
A MILTI-LAYER NEURAL NETWORK ALGORITHM FOR GAF

In spite of the obvious importance of activity recognition technology for human centric applications, state-of-the-art activity recognition technology is not practical enough for real world deployments because of the insufficient accuracy. Among the many sources of inaccuracies, we found that the accuracy of the activity model affects the performance of other AR subsystems significantly. This is because the activity model is designed in an early stage of the AR system development process and other subsystems such as activity observation subsystem or activity recognition algorithm are developed based on the activity model. Currently, there are two popular activity-modeling techniques: activity theory based technique and probability-based technique. However, both techniques do not provide the high accuracy required by real-world applications. Therefore a generic activity framework is developed to address these issues in Chapter 4. The generic activity framework requires a new activity recognition algorithm that can interface with and leverage the benefits of the framework. To substantiate the advantages of the generic activity framework, a new algorithm that utilizes a Multi Layer Neural Network (MLNNK) is developed. We chose MLNNK because it possesses a hierarchical structure that maps very well with the generic activity framework and its algorithm structure.

Next section describes the goal of MLNNK based activity recognition algorithm that is described in Section 5.2. Section 5.3, and Section 5.4 present activity recognition algorithms that utilize multi-layer neural network and fuzzy logic respectively. Validation metrics and measures are provided in Section 5.5.

5.1 Goal: Increased Activity Recognition Accuracy

The goal of Multi Layer Neural Network (MLNNK) based Activity Recognition (AR) algorithm is to quantify the improvement in accuracy obtained by incorporating an activity model based on the generic activity framework. Activity theory based AR algorithm is not accurate because of the simplifying assumptions inherent in activity theory. The accuracy of probabilistic model based AR systems such as Hidden Markov and Conditional Random Field (CRF) is not sufficient for practical applications. HMM and CRF based activity models are too rigid and lack the agility to represent dynamic nature of human activities. In other words, HMM and CRF based activity model should represent temporal relationship between human activities. Due to the complex nature of human intelligence, almost innumerable combinations of human activities are possible, and there are many possible combinations of temporal relationships between human activities. If some of activity relationships are missed in the model, these missed activity relationships can cause a recognition error. HMM and CRF may be suitable for recognizing gestures and operations, but not complex human activities. Generic activity framework aims to remove the constraints imposed by HMM and CRF and MLNNK works in tandem with activity models based on the framework.

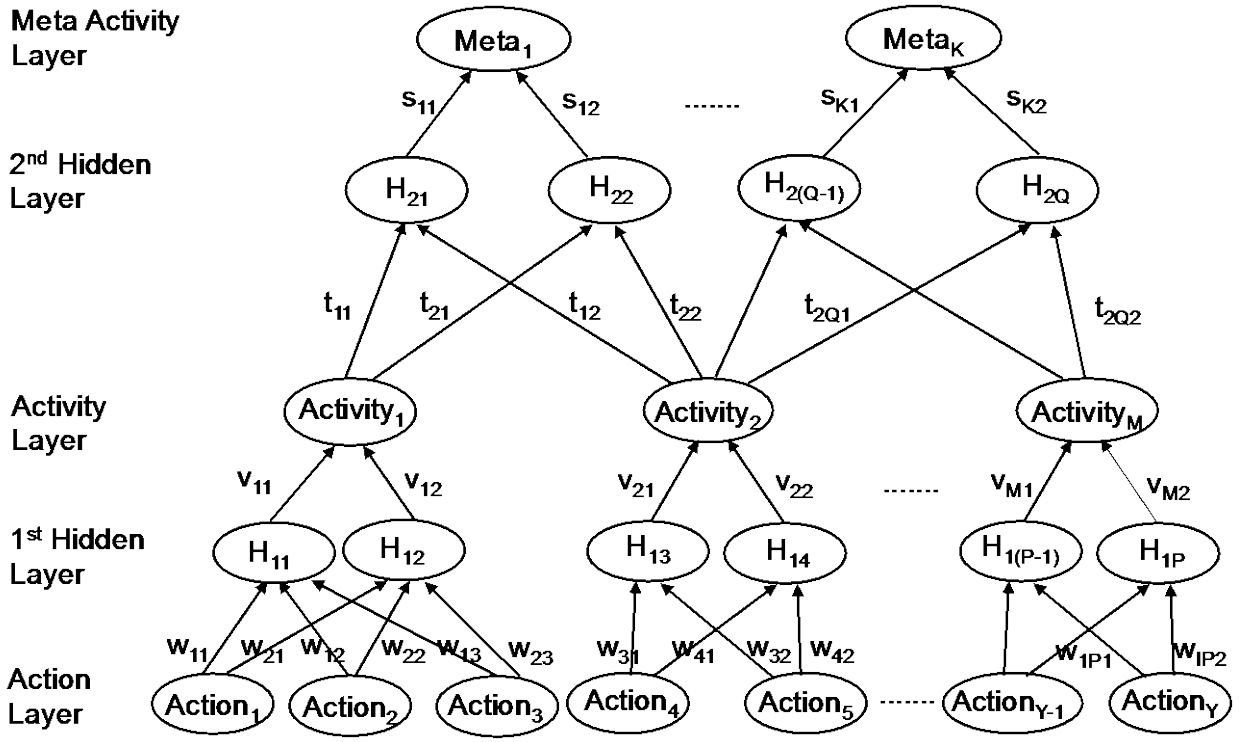
5.2 Multi-Layer Neural Network based Algorithm

This dissertation reports new activity recognition (AR) algorithm that utilizes a Multi Layer Neural Network (MLNNK) [10].

5.2.1 Multi-Layer Neural Network for GAF

We utilize MLNNK because it has a hierarchical structure that maps very well with the generic activity framework and its algorithm structure. Figure 5-1 illustrates clearly how layers of our generic activity framework are directly mapped to the layers of the

MLNNK network. Also, MLNNK in this algorithm is a localized neural network, which means it has less training burden than a unified neural network [10]. A unified (single) neural network that captures all activities is not feasible because of the high computational cost and time involved.



(a)

$$NET_{H_{11}} = \left[Action_1, Action_2, Action_3\right] \begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \end{bmatrix}, \; H_{11} = f\left(NET_{H_{11}}\right) = \left(1 + \exp^{-NET_{H_{11}}}\right)^{-1}$$

$$NET_{H_{12}} = \left[Action_1, Action_2, Action_3\right] \begin{bmatrix} w_{21} \\ w_{22} \\ w_{23} \end{bmatrix}, \; H_{12} = f\left(NET_{H_{12}}\right) = \left(1 + \exp^{-NET_{H_{12}}}\right)^{-1}$$

$$NET_{Activity_1} = \left[H_{11}, H_{12}\right] \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix}, \; Activity_1 = f\left(NET_{Activity_1}\right) = \left(1 + \exp^{-NET_{Activity_1}}\right)^{-1}$$

(b)

Figure 5-1. (a) Localized neural network for activity recognition. It is composed of several sub-neural networks, which are localized according to the relationship between inputs and outputs [46] (b) Activation function of neural networks.

For example, when two unrelated activities such as cooking and laundry are performed, their inputs are also unrelated. However, in a unified neural network, these inputs are computed together. Therefore, cooking activity should compute laundry sensor data and vice versa. Moreover, especially if there are hundreds of target activities of an AR system, even one sensor change can overwhelm the system because the system should find the relationship between the new sensor and all target activities. This explains why neural networks are not commonly used in activity recognition.

To adopt neural networks while eliminating the unnecessary waste of computational resources, we design our AR system in a way that allows unrelated activities and meta activities to have their separate neural networks so that these networks focus on more relevant relationships. We shall refer to these networks as localized neural networks. Figure 5-1 (a) shows an example of a two layer neural network [10]. In this example, there are three localized neural networks (or sub-neural network) in activity layer and two localized neural networks in meta activity layer. Activity layer is the output of the first hidden layer and the input for the second hidden layer. A neural network computes an output according to an activation function such as hyperbolic tangent function or sigmoid function. Figure 5-1 (b) shows an example of sigmoid function.

One major advantage of MLNNK is that it considers only effective relationships for activity recognition so that it can reduce error [10]. Both HMM and CRF are time sequential graphical models, which are popularly used for activity recognition. HMM requires finding all possible orders between input actions. Due to the complex nature of

human activities, finding all possible orders is cumbersome for a practical system. Furthermore, missing orders will cause the HMM to produce errors. CRF solves this problem by neglecting the order constraint. CRF does not consider order and it considers only state and transition relationships. The former is the relationship between input observation (actions or activity in Figure 5-1) and output (activity or meta activity in Figure 5-1). The latter is the relationship between output activities in the same layer (relationship between previous activities at time slice t and next activities at time slice t+1). CRF could outperform HMM even though it removes orders from an activity model [62]. Similarly, we can remove transition relationships of CRF because it is not only difficult but also meaningless to enumerate every possible relationship unless target recognition scope is small and its relationships are clear. For example, in highly abstract recognition domains such as daily living activities, after getting up in the morning, we can do a lot of things such as eating, cooking, taking a bath, running, etc. Enumerating all possible relationships is difficult and if there is a missing relationship, it will increase the error rate. MLNNK ignores the transition relationship between activities and considers only state relationships. Therefore, MLNNK is expected to outperform CRF by removing the unnecessary constraints.

5.2.2 MLNNK Training

When a new sensor is installed, the neural network needs to be retrained with new training data. Generating a new training data requires a lot of human effort as we mentioned earlier. By utilizing the knowledge, our AR system is able to generate training data automatically, using information about the types of sensors used in the target smart space. In other words, we store the sensor's information such as the range of the

values of the sensor data, feature function, and possible install locations. Using this knowledge, we can generate possible combinations of data set that sensor set produce.

I: Input data, H1: 1st hidden layer, A: Activity, H2: 2nd hidden layer, M: Meta_activity
s, t, v, w: weight vectors of each layer, (see Figure 5-1 (a))
d1, d2: desired activity, meta_activity of the input
p; the number of training data pattern pairs
E1, E2: Output error, initially zero
m1, m2: the number of hidden units in each hidden layer

(1) Assign initial weight (s, t, v, w) with random value.
(2) Set learning rate α (>0) and maximum error (Emax).
(3) For each training pattern pair (x, d1, d2),
Do following step (4) ~ (8) until k = p. (x: training pattern).
(4) Compute output H1, A, H2, M (see Figure 5-1 (b)).
(5) Compute output errors E1, E2.

$$E_1 = \frac{1}{2}\left(d_1 - A\right)^2 + E_1,\ E_2 = \frac{1}{2}\left(d_2 - A\right)^2 + E_2$$

(6) Calculate the error signal.

$$\delta_{Activity} = \left(d_1 - A\right)A\left(1 - A\right)$$

$$\delta_{MetaActivity} = \left(d_2 - M\right)M\left(1 - M\right)$$

$$\delta_{Hidden_1} = H_1\left(1 - H_1\right)\sum_{i=1}^{m_1}\delta_{Activity}v$$

$$\delta_{Hidden_2} = H_2\left(1 - H_2\right)\sum_{i=1}^{m_2}\delta_{MetaActivity}s$$

(7) Update weights s, t, v, w.

$$s^{k+1} = s^k + \Delta s^k = s^k + \alpha\delta_{MetaActivity}H_2^k$$

$$t^{k+1} = t^k + \Delta t^k = t^k + \alpha\delta_{Hidden_2}A^k$$

$$v^{k+1} = v^k + \Delta v^k = v^k + \alpha\delta_{Activity}H_1^k$$

$$w^{k+1} = w^k + \Delta w^k = w^k + \alpha\delta_{Hidden_1}I^k$$

(8) Increase a counter and go to Step 4.
k = k+1
(9) Test stop condition.
If (E1 < Emax and E2< Emax) stop else, E1=0, E2 = 0

Figure 5-2. Error back propagation algorithm

For example, if we install a new snoring sensor for recognizing a sleep apnea activity, a possible location of the sensor is at the bedside in the bedroom. Also, we can

find the features of the snoring sensor including value range from vendor datasheet (e.g. If the sensor value is 0 for no snoring and 1 for yes snoring). We store this information into knowledge storage. Later when an AR system finds the snoring sensor installed in bedroom, it queries the knowledge storage, and retrieves all related information such as sensor information and model information like related actions and activities. Then, we can generate training data automatically with this information. Figure 5-2 shows the training algorithm, which is based on the error back propagation algorithm [10,46].

5.3 Validation of MLNNK Based Activity Recognition Algorithm

In this section, we validate multi-layer neural network based activity recognition (AR) algorithm through experimentation. Our evaluation goal is to answer the following question:

- How accurate is MLNNK based AR algorithm?
- How does its accuracy performance compare to other algorithms/approaches?

5.3.1 Experiments

To perform this evaluation, we used a real world data set, which is provided by University of Amsterdam (we will refer to it in this dissertation as the Amsterdam dataset) [62]. This data set records activities of daily living performed by a 26-year-old man living in a three-bedroom apartment for 28 days [62].

Table 5-1. Meta activities and activities

Meta Activity	Activity
Cleaning	WashingDishes
GoingOut	LeavingHouse
Hygiene	Drinking,GoingToBed, TakingShower, Toileting
Laundry	WashingClothes
PreparingBreakfast	Cooking at Morning
PreparingLunch	Cooking at Lunch
PreparingDinner	Cooking at Dinner

Sensors are installed in several places in the apartment including doors, cup-boards, refrigerator, and toilet flush. Activities (such as “Leaving”, “Toileting”, “Showering”, “Sleeping”, “Drink”, “Breakfast”, and “Dinner”) are annotated by the subject himself using a Bluetooth headset [62] and used to compare the performance of activity recognition system. We identified activities and meta activities as shown in Table 5-1 [10].

We then implemented an activity recognition system using the MLNNK algorithm. The AR system collects sensor data periodically according to a time slice (1 minute) and sends the data to the neural network, which recognizes activities from the data. The recognized activities are compared with the activities annotated by a subject to measure accuracy of activity recognition algorithm.

5.3.2 Comparison and Analysis of Experiment Results

Table 5-2 shows the recognition performance of our approach. False positive is found to occur more frequently than false negative [10]. We can see that meta activity recognition performance is not much different from activity recognition performance. This is not surprising because activity is input for recognizing meta activity, and meta activity recognition performance is highly depending on activity recognition accuracy.

Table 5-2. Activity recognition performance

	Meta Activity	Activity
False positive	8.6 %	9.0 %
False Negative	2.1 %	2.1 %
Accuracy	89.9 %	88.7 %
Recall	96.7 %	96.6 %
Precision	87.7 %	87.4 %
Specificity	81.0 %	75.3 %

Figure 5-3 shows the activity recognition performance such as accuracy, recall, precision, and specificity of MLNNK based activity recognition algorithm.

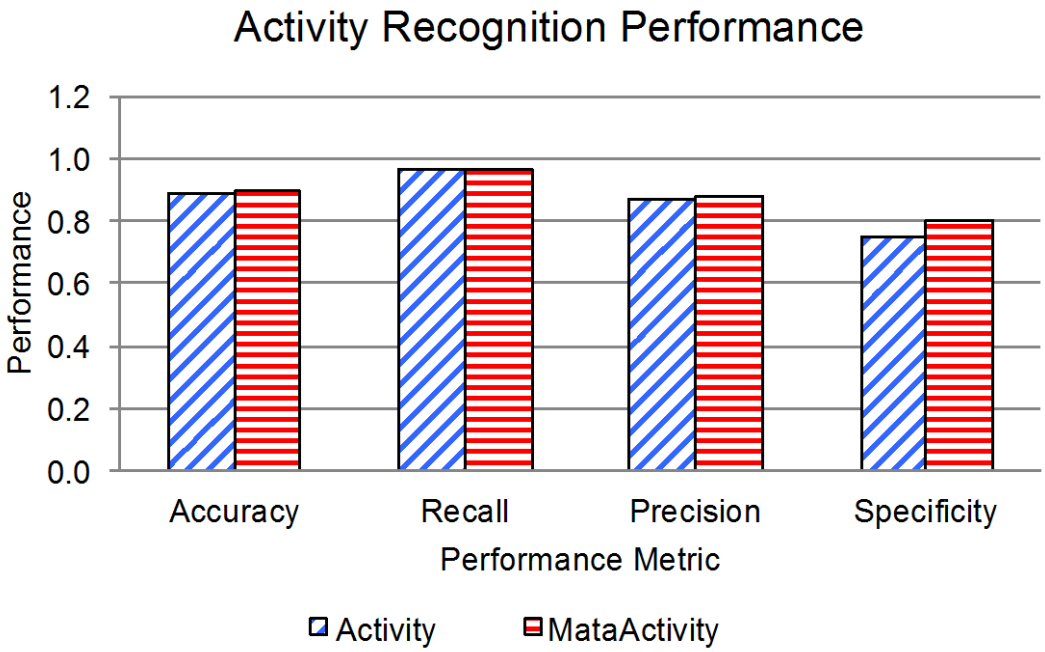


Figure 5-3. Activity recognition performance of MLNNK based AR algorithm

The introduced approach is compared with HMM and CRF model based approaches. Table 5-3 and Figure 5-4 show the performance comparison among the three approaches. MLNNK based approach shows higher accuracy than HMM and CRF for both Timeslice and Class. The Class accuracy is over 23% higher than CRF and about 14.6% higher than HMM. Another observation is that MLNNK shows similar performance for Timeslice and Class whereas Timeslice performance is higher than Class for HMM and CRF. There are a couple of reasons for this performance difference. First, the performance between Timeslice and Class in CRF and HMM are different because their training data are collected in a biased situation. In reality, some activities will be performed frequently whereas other activities will be rare. Then, the rare activity classes take more time to be trained because they need to wait until their training data is collected [62]. However, Timeslice does not have this difference. Therefore, the overall performance of Timeslice is higher than that of Class.

Table 5-3. Comparison of activity recognition performance

Algorithm	Time Slice	Class
Hidden Markov Model [62]	94.5%	79.4%
Conditional Random Field Model [62]	95.6%	70.8%
Multi-Layer Neural Network (MLNNK)	96.9%	94.0%

However, MLNNK shows similar performance for both Timeslice and Class. It is because MLNNK training is different from other approaches. MLNNK does not wait until training data is collected. In other words, since our AR system can produce initial training data using sensor and activity model information in knowledgebase, every class is trained almost equally.

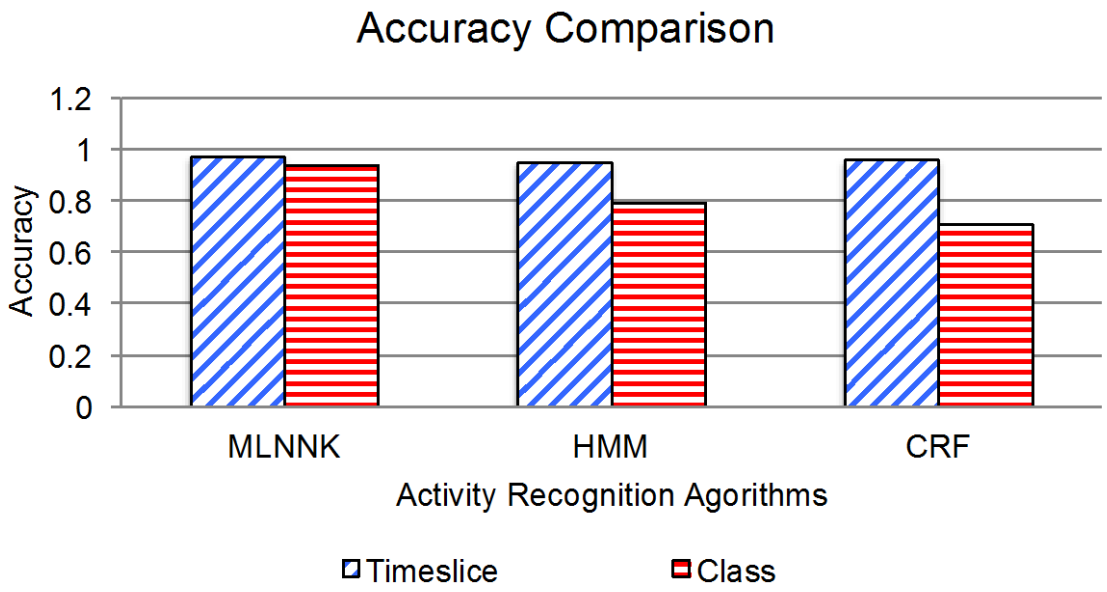


Figure 5-4. The comparison of activity recognition accuracy

Also, because MLNNK is a localized network, the training data is not huge unlike a unified neural network. Therefore, MLNNK can start training every class from the beginning and shows similar performance.

Table 5-4 dissects the accuracy for each activity class. “PreparingBreakfast” has the worst accuracy while “LeavingHouse”, “GoingToBed”, and “PreparingDinner” have the best accuracy. This result is similar to previous experiments reported in [62].

Table 5-4. Detailed class accuracy result

Class	Accuracy
LeavingHouse	1.00
Toileting	0.94
TakingShower	0.95
GoingToBed	1.00
PreparingBreakfast	0.76
PreparingDinner	1.00
Drinking	0.91

5.4 Summary and Discussion

Activity recognition (AR) accuracy will be improved by enhancing the accuracy of activity model and activity recognition algorithm. However, the activity model requires activity recognition algorithm that can support the activity model. Multi-layer neural network (MLNNK) based AR algorithm reflects hierarchical activity model and has betteractivity recognition performance than non-hierarchical HMM and CRF.

CHAPTER 6
FUZZY LOGIC BASED ACTIVITY RECOGNITION ALGORITHM FOR SGAF

Achieving human activity recognition (AR) with high accuracy is challenging because the recognition process can produce ambiguous results. Sometimes, people initiate an activity but do not complete it. These partially performed activities result in recognition ambiguity. For example, people may ingest a small amount of food but not complete their meal. In this case, it is not clear whether this partially performed activity should be considered as an “eating” activity or not. Secondly, when an activity initiates, there may not be enough information to determine the activity with a high confidence level. It takes a finite time interval to determine the performed activity. To increase AR accuracy in these ambiguous situations, a new activity modeling technique based on generic activity framework and activity semantic knowledge (SGAF) is introduced in Chapter 4. To take advantage of the SGAF, it is necessary to develop an activity recognition algorithm that can maximally utilize and cooperate with the SGAF.

However, many activity recognition techniques do not cooperate with activity semantics and many of them are based on supervised or semi-supervised machine learning algorithms that require training. In such technologies, activity recognition accuracy is highly dependent upon the training data and training process. Due to the variety of human activities, it is almost impossible to collect a comprehensive dataset for training a system to achieve high accuracy. Additionally, collecting training data incurs substantial cost and effort.

To address both issues, we have developed a new training free activity recognition approach based on a fuzzy logic algorithm that utilizes activity model and associated activity semantic knowledge in lieu of training.

Next section describes the goal of applying fuzzy logic to activity recognition. In Section 6.2, semantic fuzzy logic operators are suggested and Section 6.3 shows how to apply fuzzy logic to activity recognition. The implementation of this algorithm is described in Section 6.4. In Section 6.5, experiment results for validation are discussed.

6.1 Goal: Increased Activity Recognition Accuracy and Tolerance to Uncertainty

A major goal of Fuzzy Logic (FL) based Activity recognition (AR) algorithms is that they enhance AR accuracy even without training. Additionally, because it may not be possible to develop a perfect algorithm that completely accounts for all uncertainties, a practical activity recognition (AR) system should be able to tolerate a certain amount of uncertainty.

FL based AR algorithms are well suited for achieving the above goals because fuzzy operation and inference are able to derive highly accurate results from uncertain information using activity semantics. FL based AR algorithm can also deal with ambiguity in activities or activity semantic knowledge using linguistic words. Successful activity recognition requires the understanding of linguistic variables, which tend to be vague. Linguistic variables are used in ordinary daily activities [65]. For example, an activity such as “sleeping tightly” has a linguistic variable valued as “tightly”. Therefore, fuzzy logic is more suitable for handling intrinsic uncertainties in a subject’s activity.

6.2 Fuzzy Logic for Activity Recognition

Activity semantics should be applied in activity recognition process. In fuzzy logic based algorithm, activity semantics are combined with fuzzy logic by extending fuzzy logic to activity semantic fuzzy logic (or semantic fuzzy logic for simplicity).

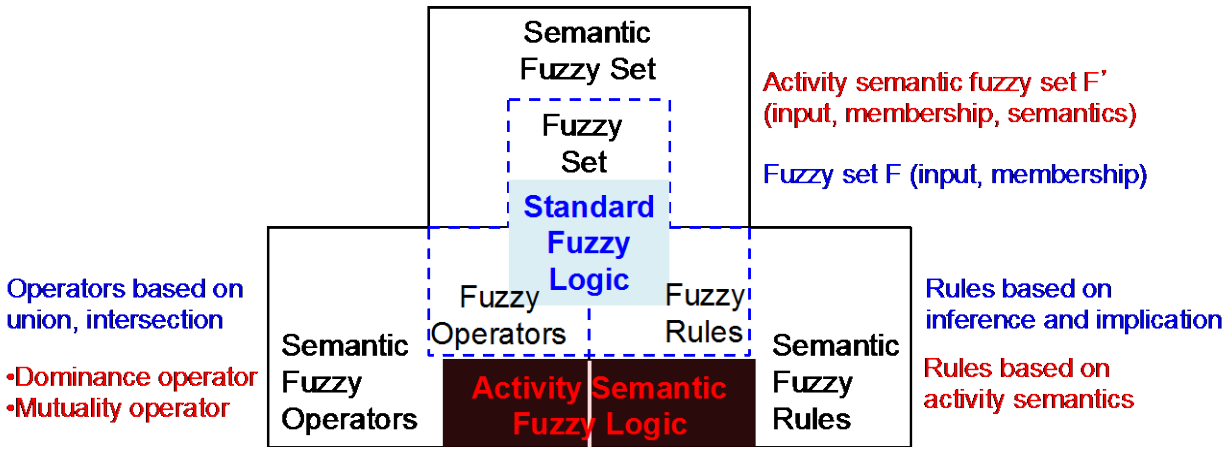


Figure 6-1. Extension of fuzzy logic to activity semantic fuzzy logic

There are three extensions: Extension of fuzzy set, Extension of fuzzy operators, and Extension of fuzzy rules as shown in Figure 6-1.

6.2.1 Extension of Fuzzy Set to Activity Semantic Fuzzy Set

In fuzzy set F, x is an input element of a real set X. An element of semantics set S is denoted by s. A fuzzy membership value of x on fuzzy membership function uA is denoted as uA(x) or u(x). A fuzzy set F is extended to an activity semantic fuzzy set F' that contains activity semantics. Fuzzy set F (input value, fuzzy membership value) is extended to Semantic fuzzy set F' (input value, fuzzy membership value, activity semantic) that includes an additional "semantic" variable. For example, F={(300, 0.3), (500, 0.5), … , (10, 0.01)} is extended to F'={(300, 0.3, Optional), (500, 0.5, Concurrent), … , (10, 0.01, Key)}.

Table 6-1. Fuzzy set and semantic fuzzy set

Set	Set Expression
Fuzzy set	$F = \{(x, u_F(x)) \mid x \in X\}$
Extended fuzzy set	$F' = \{(x, u_F(x), s) \mid x \in X, s \in S\}$

6.2.2 Extension of Fuzzy Operator to Activity Semantic Fuzzy Operator

Existing fuzzy operator (or fuzzy set operator) is extended to fuzzy dominance operator and fuzzy mutuality operator. Table 3-2 shows frequently used T-norm and S-norm fuzzy operators.

Table 6-2. Frequently used fuzzy set operators

T-norm operators	S-norm operators
(1) Minimum: $T_{min}(x, y)=\min(x, y)=x \wedge y$	(1) Maximum: $S_{max}(x, y)=\max(x, y)=x \vee y$
(2) Algebraic Product: $T_{ap}(x, y)=xy$	(2) Algebraic Sum: $S_{as}(x, y)=x + y - xy$
(3) BoundedProduct: $T_{bp}(x, y)=0 \vee (x + y - 1)$	(3) BoundedSum: $S_{bs}(x, y)=1 \vee (x + y)$
(4) DrasticProduct: $T_{dp}(x, y)=\begin{cases} x, \text{ if } y=1 \\ y, \text{ if } x=1 \\ 0, \text{ if } x, y < 1 \end{cases}$	(4) DrasticSum: $S_{ds}(x, y)=\begin{cases} x, \text{ if } y=0 \\ y, \text{ if } x=0 \\ 1, \text{ if } x, y > 0 \end{cases}$
(5) Weak: $T_W(x, y)=\begin{cases} \min(x, y) & \text{if } \max(x, y)=1 \\ 0 & \text{otherwise} \end{cases}$	(5) Strong: $S_S(x, y)=\begin{cases} \max(x, y) & \text{if } \min(x, y)=0 \\ 1 & \text{otherwise} \end{cases}$

Fuzzy Dominance Operator. This operator computes activity dominance semantics. Fuzzy dominance operator is defined that “The values of every key component should be greater than 0 for any recognized activity”. According to the defined operation, fuzzy dominance operator is defined in Table 6-3. In the table, x and y are operands. Their fuzzy values are u(x) and u(y). Their dominance semantic is either “Key” or “Optional”. We assign a fuzzy value to dominance semantics. In other words, we assign 1 to “Key” and 0 to “Optional” according o their dominance degree. If either x or y is a key component of an activity, their dominance operation result is also a key. The fuzzy value of dominance operation is minimum fuzzy value of all key

operands. If any key component has fuzzy value 0, then fuzzy dominance operation will also return 0.

Table 6-3. Fuzzy dominance operator (Operator symbol: @)

x	y	Operations (x@y)	Examples (Semantic, Fuzzy membership)
Key (=1)	Key	(Key, (min(u(x), u(y))	(Key, 0.3)@(Key, 0.5)) = (Key, 0.3)
Key	Optional	(Key, u(x))	(Key, 0.5)@(Optional, 0.7) = (Key, 0.5)
Optional (= 0)	Optional	(Optional, 1)	(Optional, 0.9)@(Optional, 0.6) = (Optional, 1)

Fuzzy Mutuality Operator. The fuzzy mutuality operator checks concurrency or mutuality semantics between two input data. The value of concurrency is between 0 and 1, depending on concurrency. We assign 1 to “Concurrent” and 0 to “Exclusive” semantic. For “Ordinary”, a fuzzy value between 0 and 1 is assigned (e.g. 0.5). Table 3 shows mutuality operation. If either x or y is exclusive, mutuality operation returns the operand that has a greater fuzzy value. Otherwise, it returns both operands.

Table 6-4. Fuzzy mutuality operator (Operator symbol: #)

x	y	Operations	Examples (Semantic, Fuzzy membership)
Concurrent	Concurrent	{(1, x), (1, y)}	(1, watchingTV(0.1)) # (1, laundry(0.2)) ={(1, watchingTV(0.1), (1, laundry(0.2)}
Concurrent	Ordinary	{(1, x), (0.5, y)}	(1, watchingTV (0.1)) # (0.5, eating(0.6)) ={(1, watchingTV(0.1), (0.5, eating(0.6))}
Exclusive	Concurrent, Exclusive, Ordinary (= 0.5)	{(0, x)} if (u(x) > u(y)) {(0.5, y)} otherwise	(0,sleeping(0.3)) # (0.5,eating(0.6)) ={(0.5, eating(0.6))}
Ordinary	Ordinary	{(0.5, x). (0.5, y)}	(0.5, talking(0.2)) # (0.5, eating(0.6)) ={(0.5, talking(0.2)), (0.5, (eating(0.6))}

6.2.2 Extension of Fuzzy Rule to Activity Semantic Fuzzy Rule

Fuzzy rule is based on classical implication and inference rules. The difference between classical reasoning rule and fuzzy rule is that x and y values denote fuzzy values in fuzzy rules whereas they represent Boolean value in classical reasoning rules as shown in Table 6-5. In fuzzy logic, these rules are used for approximate reasoning; the resulting output is linguistic words. For example, if an implication rule (if x then y, $x \rightarrow y$) is combined with a linguistic word “strong” then the fuzzy rule is “if x is strong then

y is strong" [31]. Then, one of inference rules is "if y is not strong, then x is not strong" because inference rules are derived from implication rule. These fuzzy implication rules and inference rules in fuzzy logic are extended for activity recognition; they are applied for checking activity semantics like dominance operator and mutuality operator. In activity semantic fuzzy logic, the fuzzy rule is "if an activity is more clearly performed then the semantics of the activity are more satisfied".

Table 6-5. Fuzzy rules with two variables x and y

Rule	Rule expression	Rule in fuzzy membership
Implication	$x \rightarrow y$	min(1, 1+y-x)
Inference - modus ponens	$(x \wedge (x \rightarrow y)) \rightarrow y$	IF (u(x) > 0 AND $u(x \rightarrow y)$) THEN u(y) > 0
Inference - modus tollens	$((x \rightarrow y) \wedge \neg y) \rightarrow \neg x$	IF ($u(x \rightarrow y)$ > 0 AND (1 - u(y)) > 0) THEN u(y) > 0

Table 6-6, Table 6-7, Table 6-8, and Table 6-9 show examples of activity semantic rules.

Dominance semantic rules. To perform an activity completely, all key components of the activity should be performed. This semantic is implemented by an activity semantic fuzzy operator. However, it can also be implemented by using semantic fuzzy rule. Table 6-6 shows implication rule and inference rule derived from the implication rule of dominance semantic. A is an activity. The fuzzy value of the activity A is u(A). Key_A indicates all key components of A. $Key_{(A,i)}$ is the i^{th} key component of A.

Table 6-6. Fuzzy rules for dominance semantic

Rule	Rule expression
Implication (Rule1)	$(u(A) > 0) \rightarrow (Key_{(A,i)} > 0), \forall Key_{(A,i)} \in Key_A$
Inference (Rule2)	$(Rule1) \wedge (Key_{(A,i)} == 0, \exists Key_{(A,i)} \in Key_A) \rightarrow (u(A) == 0)$

Mutuality semantic rules. Two activities that can be concurrently performed should not be an element of an exclusive activity set. The mutuality semantic fuzzy rules are shown in Table 6-7. In the table, A and B are activities. The time during which

activities A and B are performed are t(A) and t(B), respectively. $Activity_{Exclusive}$ is a set of all exclusive activities.

Table 6-7. Fuzzy rules for mutuality semantic

Rule	Rule expression
Implication (Rule3)	$((A \in Activity_{Exclusive}) \wedge (t(A) \cap t(B) \neq \varnothing)) \rightarrow (u(B)==0)$
Inference (Rule4)	$(Rule3 \wedge (u(B)==0)) \rightarrow (u(A)==0)$

Order semantic rules. If an activity is performed, all prerequisite activities in strong relationship with this activity should be performed. A_i, Aj are activities in strong order relationship $StrongOrder(A_i, A_j)$ in Table 6-8.

Table 6-8. Fuzzy rules for order semantic

Rule	Rule expression
Implication (Rule5)	$(u(A_i) > 0) \rightarrow (u(A_j) > 0), \forall StrongOrder(A_t, A_j)$
Inference (Rule6)	$(Rule5 \wedge u(A_j) > 0) \rightarrow u(A_{ti} > 0), \forall StrongOrder(A_t, A_j)$

Effect semantic rules. If an activity is truly performed, the effect of the activity should also be recognized. A is an activity. $Effect_{(A, i)}$ is the ith effect of the activity. $Effect_A$ is a set of all effects of A in Table 6-9.

Table 6-9. Fuzzy rules for effect semantic

Rule	Rule expression
Implication (Rule7)	$(u(A) > 0) \rightarrow (\exists Effect_{(A,i)} > 0), Effect_{(A,i)} \in Effect_{(A)}$
Inference (Rule8)	$(Rule7 \wedge (\forall Effect_{(A,i)} == 0)) \rightarrow (u(A) == 0)$

6.3 Applying Activity Semantic Fuzzy Logic to Activity Recognition

In this section, we discuss how to apply semantic fuzzy logic described in the previous section to activity recognition. As mentioned in Chapter 4, activity model is a hierarchical structure; the hierarchical structure is illustrated in the following example. During the initial phase of recognizing an activity, only low-level sensor data is available. From these sensor data, AR algorithm should recognize performed activities by computing the fuzzy value of an activity using semantic fuzzy logic techniques and

activity model. Semantic fuzzy logic techniques that were developed as part of this research effort are described next.

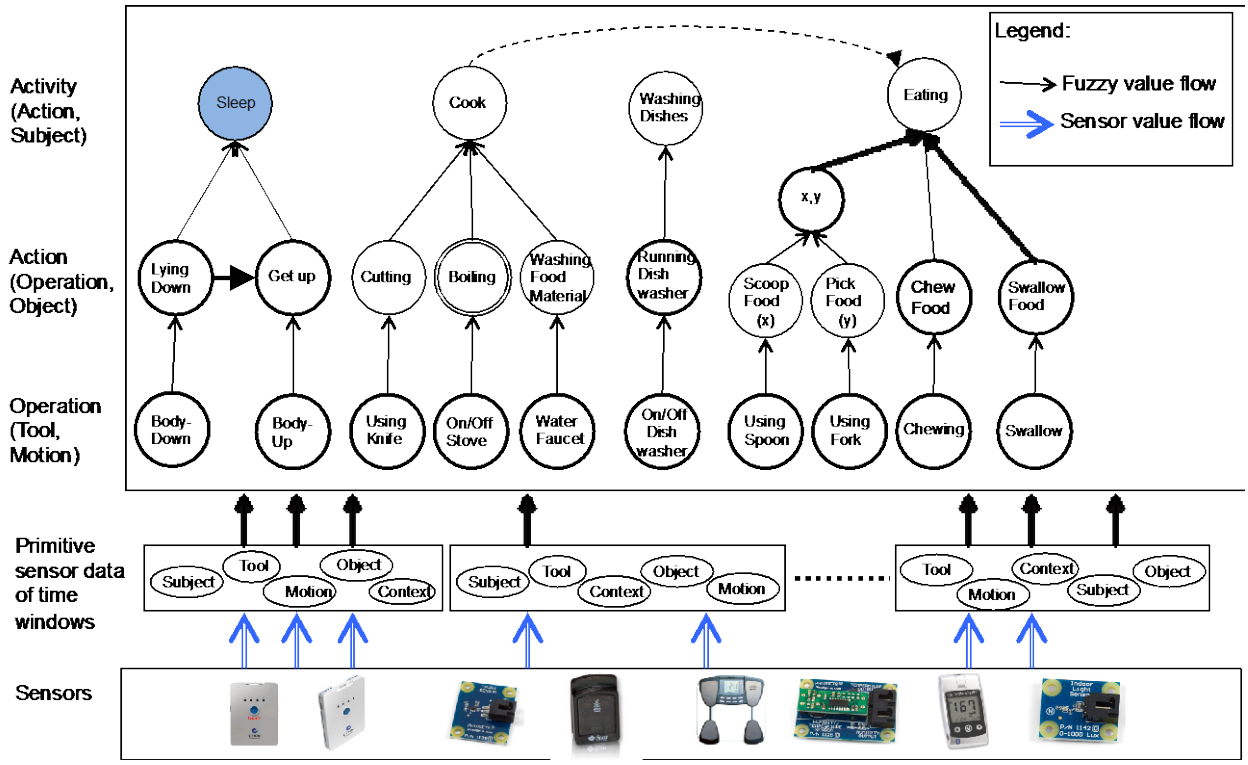


Figure 6-2. An example of activity model

6.3.1 Fuzzy Membership Function

Fuzzy membership function is used at two points in the activity recognition process. First, it is used to convert sensor values to fuzzy membership values. When a sensor event is triggered, the sensor value is not initially a fuzzy value. Different sensors have a different output range. For example, a light sensor has sensor value from -10 to 10 whereas motion sensor has a value from 0 to 360. Fuzzy membership function converts these sensor values to fuzzy membership values between 0 and 1. There are popular fuzzy membership functions such as triangular or trapezoidal fuzzy membership functions [31].

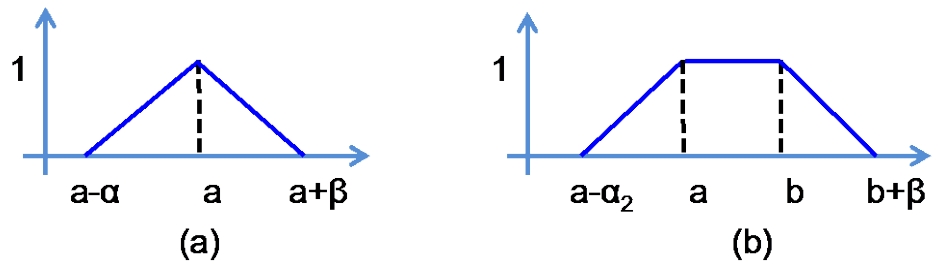


Figure 6-3. Examples of triangular fuzzy membership (a) and trapezoidal fuzzy membership (b) with three fuzzy membership indices

The following equation (6-1) and equation (6-2) define the triangular and trapezoidal fuzzy membership function of "middle" index corresponding to Figure 6-3. Fuzzy membership functions of other indices are defined in a similar fashion.

$$A(t)=\begin{cases}1-\dfrac{a-t}{\alpha} & \text{if } a-\alpha\le t\le a\\ 1-\dfrac{t-a}{\beta} & \text{if } a\le t\le a+\beta\\ 0 & \text{otherwise}\end{cases} \tag{6-1}$$

$$A(t)=\begin{cases}1-\dfrac{a-t}{\alpha} & \text{if } a-\alpha\le t\le a\\ 1 & \text{if } a\le t\le b\\ 1-\dfrac{t-b}{\beta} & \text{if } b\le t\le b+\beta\\ 0 & \text{otherwise}\end{cases} \tag{6-2}$$

Where, t is a real value that needs to be transformed to a fuzzy value.

Second, fuzzy membership function is used to combine the fuzzy value of an activity with linguistic terms. After computing the numeric fuzzy values of activities, the values are transformed to corresponding linguistic terms. It is easier for people to understand. For example, if the fuzzy value of an activity is 0.6, the fuzzy membership of the activity is $u_{activity} = \{u_{Verylittle}, u_{Little}, u_{Somewhat}, u_{Much}, u_{Verymuch}\}=\{0,0,0.75,0.25,0\}$. And

the linguistic term is “activity is somewhat performed” according to an example of decision index shown in Figure 6-4.

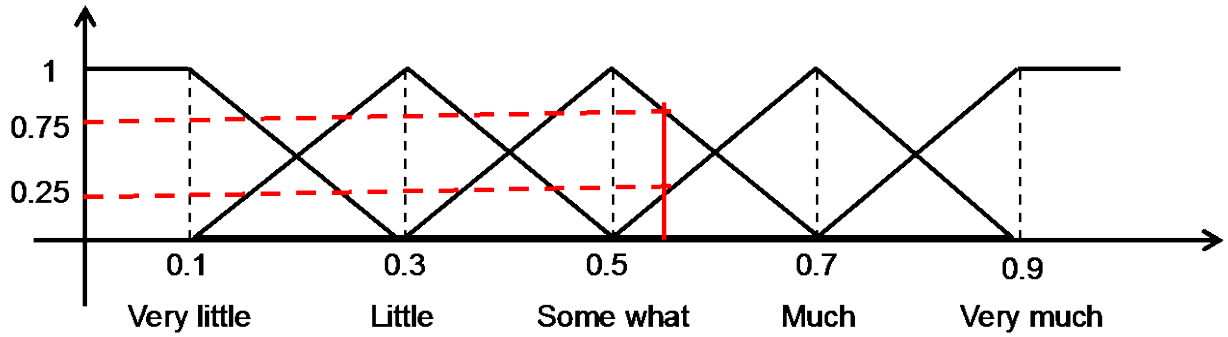


Figure 6-4. An example of decision Indices to decide fuzzy membership of an activity

Defuzzification of fuzzy value. Defuzzification function converts a fuzzy membership value to a quantified value. Among several methods for defuzzification, centroid function is used in this dissertation.

$$\frac{\sum_{i} FV_i \times D_i}{\sum_{i} FV_i} \tag{6-3}$$

Where, FV_i is a fuzzy value and D_i is a decision index.

6.3.2 No Training Weight Computation

Even though an activity is composed of several components, their contributions to the activity vary. Some components are very crucial to determine a performed activity whereas some are not. The weight value of a component determines how strongly it impacts the activity. If the weight value of a component is high, the component has powerful influence; therefore, the weight value should be computed carefully. Many supervised machine-learning algorithms compute a weight value via training. They collect sample data and train the algorithm with the sample data. Training is a reasonable approach for certain applications if sufficient training data can be collected

in a cost and resource effective manner. However, it is difficult to apply training based algorithms to human activity recognition because human activities are very complex and it is difficult to collect sufficient data with the constraints of time and resources. In this dissertation, weights are computed based on activity model.

The weight (w_i) of a child component (c_i) is determined by contribution and evidential power of the child component. Because the contribution and evidential power of the child component are computed based an activity model, this approach does not require training.

$$w_i = \text{contribution}_i \times \text{evidential\_power}_i \tag{6-4}$$

The contribution (contribution_i) of a child (c_i) indicates how much this child component contributes to its parent component. For instance, in Figure 6-5, p1 has three children c1, c2, and c3; therefore, the contribution value of each child is 0.33.

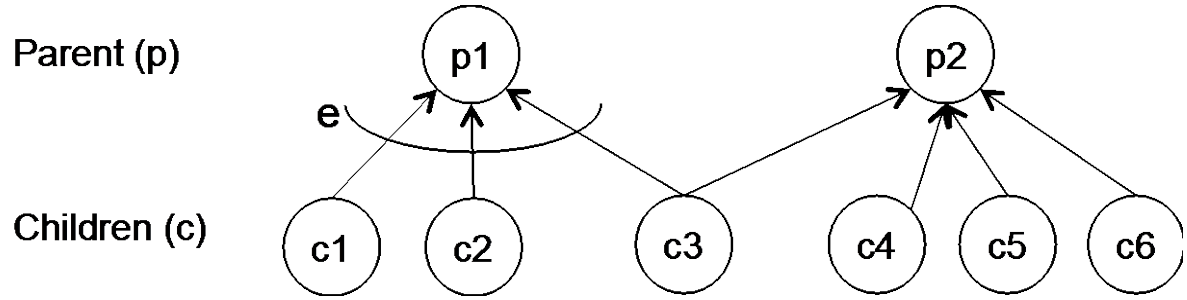


Figure 6-5. Contribution and evidential_power relationship between children components and parents.

In Equation (6-5), assuming that every child component contributes equally to their parent, n is the number of children components of a parent.

$$\text{contribution}_i = \frac{1}{n} \tag{6-5}$$

Where, n is the number of incident edges to p, e(c→p, p)

The evidential power (evidential_poweri) of a child (c_i) represents how much evidential information a child component has to determine a parent component. To illustrate, if a sensor is only used to detect a specific activity, the evidential power of the sensor data is very high. On the other hand, if a sensor is used to detect several activities, the evidential power of the sensor data will be low. In Equation (6-6), m is the number of parents of a child component. For example, in Figure 6-5, c3 has two parents p1 and p2; therefore, the evidential_power of c3 is 0.5.

$$\text{evidential\_power}_i = \frac{1}{m} \tag{6-6}$$

Where, m is the number of outgoing edges from c, e(c→p, p)

The weight of a group. If multiple components are triggered at the same time, they are considered to form a group. When there is a group of children components, the weight may be computed as a group because the fact that they occur at the same time is an important information to determine the performed activity. For example, in Figure 6-6, if two components b and c are detected, it is highly likely that A2 was performed. When b and c are detected at different instants of time, the evidential power is 0.5 for both of them according to Equation (6-3). If b and c are detected at the same time, the evidential power of {b, c} on A2 is 1 and weight of each of them is 0.5.

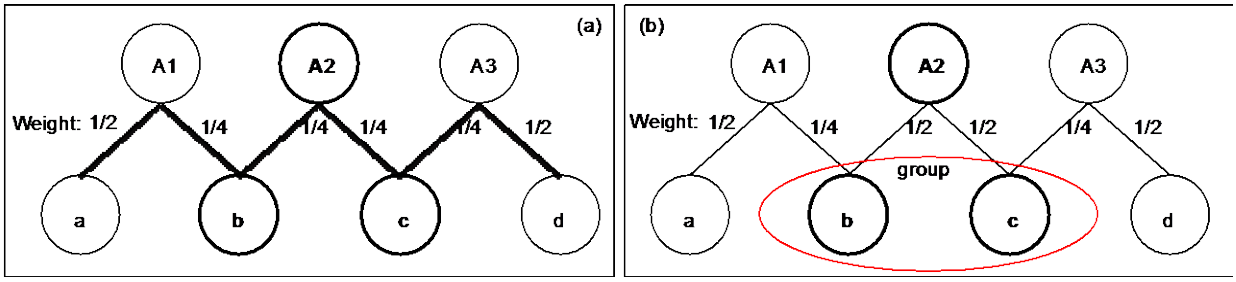


Figure 6-6. Weights of group components. (a) It shows the weight of b and c when they happened in different time; therefore, they are not in a group. (b) It shows the weight when b and c are detected at the same time.

6.3.3 The Computation of Fuzzy Value of Activities

The fuzzy value (FV(p)) of a parent component p is computed using the bounded algebraic sum of T-norm operation. The T-norm operation is the algebraic product of weight (w_i) and the fuzzy value ($FV(c_i)$) of every child component c_i as shown in Equation (6-7).

$$FV_p = 1 \wedge \left(\sum_{i=1}^{n} T\left(w_i, FV\left(c_i\right)\right) \right) = \min\left(1, \left(\sum_{i=1}^{n} w_i \times FV\left(c_i\right) \right) \right) \quad (6\text{-}7)$$

Where, $FV(c_i)$ is a fuzzy value of a child component.

6.3.4 Enforcing Semantics Using Semantic Fuzzy Operators or Rules

After computing fuzzy value of activities, there may be several activities whose fuzzy value is positive. Among them, some activities are truly performed. However, they may not be actually performed even though they have a positive value. Such positive values may be attributed to recognition error and uncertainty sources. To eliminate false recognition, activity semantics are evaluated according to activity semantic fuzzy operations and rules described in Section 6.2. If an activity does not satisfy pre-established semantics, it is eliminated from the recognition process.

6.3.5 Evaluation of Activity Life Cycle

Activity takes time from start to finish. Sometimes, an activity may be paused and restarted at a later time. When an activity is recognized, the life cycle of the activity is evaluated. If the activity is an on-going activity, all historical records since the activity is performed in activity life cycle heap. When an activity is completed, then the activity is permanently recorded and it is removed from the heap. These techniques are implemented in activity recognition system described in the next section.

6.4 Fuzzy Logic Based Activity Recognition System

Activity model framework, activity semantics, extended fuzzy logic for activity recognition, and activity model based weight computation are integrated in fuzzy logic based activity recognition algorithm. Before implementation, fuzzy logic based AR system is designed. In Figure 6-7, fuzzy membership function and activity recognition engine utilize knowledgebase.

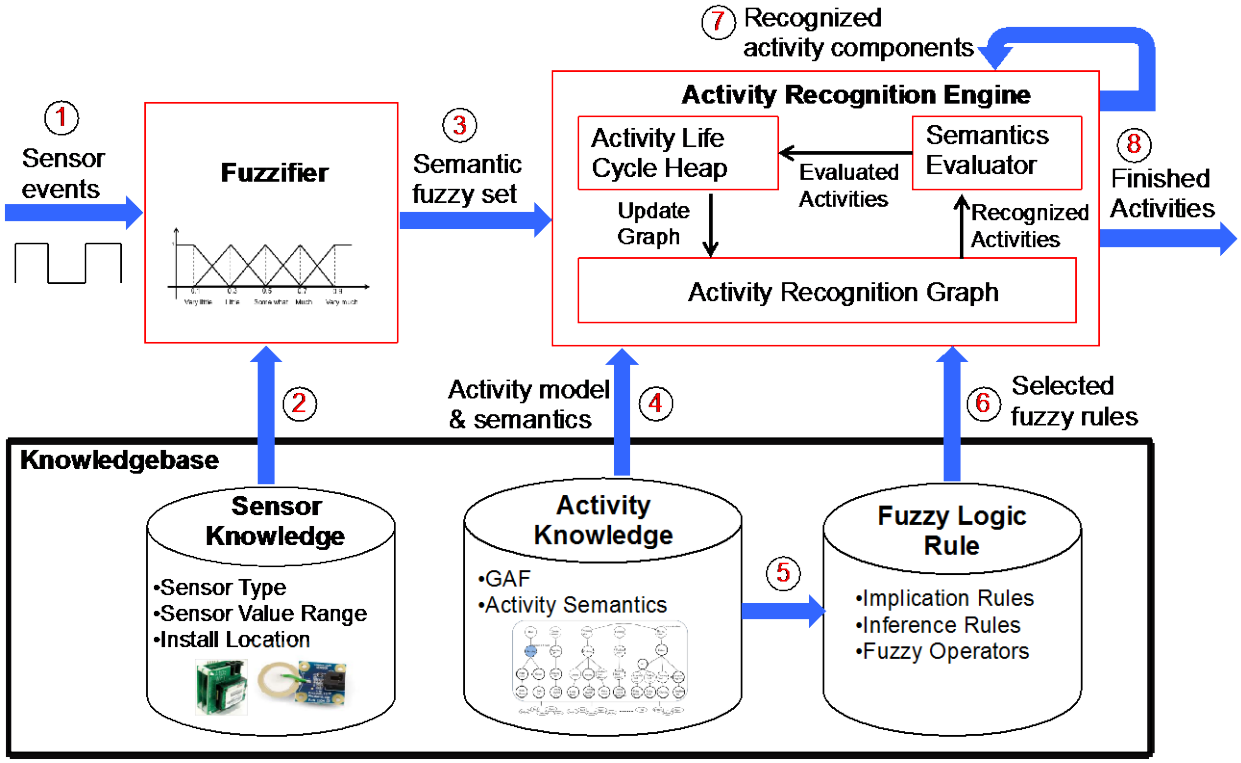


Figure 6-7. The architecture of fuzzy logic and activity semantics based activity recognition system and activity recognition flow

When these fuzzy membership functions are applied to sensors, these functions need to know sensor information such as sensor value type or sensor value range. This sensor information is available in sensor knowledgebase.

Fuzzifier. According to the fuzzy membership functions described in Section 6.3.1, it transforms sensor data values to fuzzy values. When these fuzzy membership functions are applied to sensors, these functions need to know sensor information such

as sensor value type or sensor value range. This sensor information is available in sensor knowledgebase.

Activity recognition engine performs fuzzy operation and fuzzy reasoning rules to determine performed activities using sensor data. These operations are performed based on activity model and activity semantic knowledge. Activity recognition engine is composed of two sub systems.

Activity Recognition Graph (AR Graph). Activity recognition graph is a hierarchical graph and it is a subset of activity model. When sensor event is detected, AR system searches activity model and finds all related activity components and builds an activity recognition graph. After building a graph, it computes the weights for every edge as described in Section 6.3.3 and it performs fuzzy operation from the bottom layers to the activity layer.

Semantics Evaluator. To verify recognized activity from activity recognition graph, AR engine enforces activity semantics described in Section 6.3.4.

Activity Life Cycle Heap (ALC Heap). After activity recognition graph computes the fuzzy values of activities, it send all activities to activity life cycle heap to evaluate the activity life cycle of recognized activities as shown in Figure 4-3 in Section 4.3 and Figure 6-8. An activity is performed over a finite duration from activity initiation to activity completion. To determine whether an activity is completed, on going, or paused, it is necessary to track its life cycle states.

As shown in Figure 6-7, there are three types of knowledge to assist activity recognition subsystems.

Sensor Knowledge. Sensor Knowledge contains information of all installed sensors such as sensor type, sensor value range or installed location. The sensor information is utilized for transforming raw sensor values to fuzzy values.

Activity Model Knowledge. Activity Model Knowledge is the knowledge about generic activity framework and activity semantics, along with some instances of activity models in a specific activity domain, which follow the framework and semantics.

Fuzzy Logic and Rules. Fuzzy logic rules are applied to determine activities. The fuzzy rules are derived from the activity model in activity knowledgebase.

Figure 6-8 shows overall procedure of activity recognition.

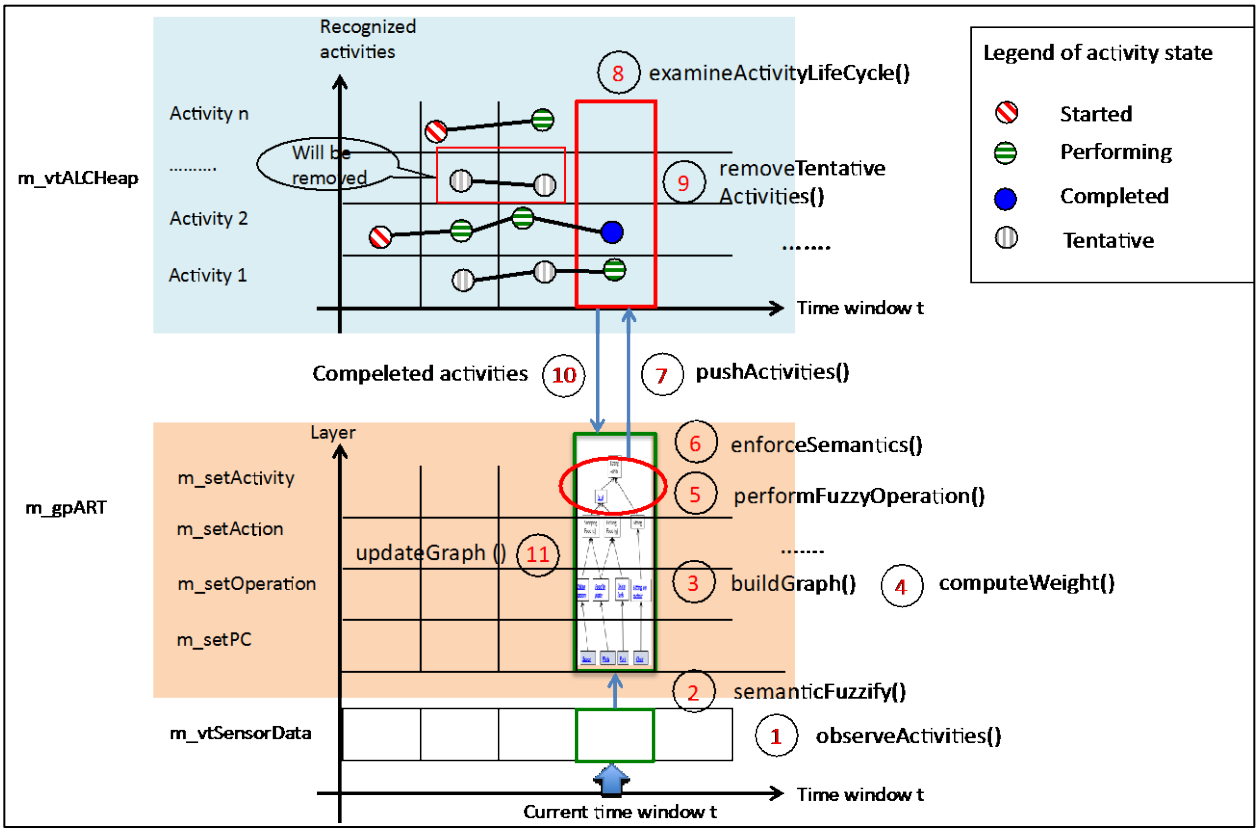


Figure 6-8. The activity recognition procedure of AR Graph and ALC Heap

The detailed algorithm of the activity recognition engine is described in Table 6-10. The procedure numbers in Figure 6-8 is matched with the step numbers in Table 6-10.

Table 6-10. The activity recognition algorithm of activity recognition engine

```
void AREngine()
{
    int t=0;
    while(true)
    {
        //Step (1). observe sensor events in a time window w and collect sensor data into V,
        //DURATION: time duration to collect sensor data, e.g. 60 seconds
        m_vtSensorData = observe (DURATION);
        if (m_vtSensorData == null && checkStopCondition()) exit;

        //Step (2). transform input sensor data to semantic fuzzy set
        //fuzzify input values, assign the fuzzy set to relevant Primitive
        //Components(PCs), and find their semantics
        m_setPC = semanticFuzzify(m_vtSensorData );

        //Step (3). build a graph
        //3-1. initialize an activity recognition graph of current time window t;
        //remove all existing compositional components (CCs)
        //in the graph, and add m_setPC to m_gpART.m_layerPrimitive
        m_gpART.m_dTimeWindow = t;
        m_gpART.m_ActivityLayer.removeAllComponents();
        m_gpART.m_ActionLayer.removeAllComponents();
        m_gpART.m_OperationLayer.removeAllComponents();
        m_gpART.m_PrimitiveLayer.m_setComponent += m_setPC;
        //3-2. find the relevant CCs of primitive components (PCs) in the graph to,
        //& include the CCs to m_gpART
        m_gpART.buildGraph(m_gpART.m_layerPrimitive);

        //Step (4). compute the weight of each component (vertical edges) in the graph.
        //Weight: the degree of contribution X information strength of a child component for its parent CCs
        m_gpART.computeWeight();

        //Step (5). for each layer, compute the fuzzy value of CCs using activity semantic fuzzy operators
        //Layer[0]: Primitive layer, Layer[1]: Operation layer, Layer[2]: Action layer, Layer[3]: Activity layer
        for (int nLayer = 1; nLayer <= 3; nLayer++)
        {
            Layer currentLayer = m_gpART.getLayer(nLayer);
            for each compositional component  in currentLayer
            {
                Component CC = getNextComponent(currentLayer);
                //find all children components of CC
                Vector<Component> vtChildrenComponent = lookupChildrenComponents(CC, m_gpART);
                //compute the fuzzy value of CC using children components.
                //update variable cc.m_dtimeLastUpdate with the most recent time of components
                CC.m_nFuzzyValue = performFuzzyDominanceOpertion(CC, vtChildrenComponent);
                CC.m_dTimeLastUpdate = getMaxTime(vtChildrenComponent);
            }

            //Step (6). enforce semantics of current layer and update the graph
            m_gpART = enforceSemantics(currentLayer);
        }

        //Step (7). push each activity in m_gpART.m_layerActivity to m_htACLHeap
        for (int I = 0; i < m_gpART.m_layerActivity.size(); i++)
        {
            Activity recognizedActivity = m_gpART.m_layerActivity.elementAt(i);
            String strHashTableKey = recognizedActivity.getId();
            ACLHeap aclHeap = m_htACLHeap.get(strHashTableKey );
            aclHeap.m_dTimeWindow = t;
            aclHeap.m_vtACLHistory.addElement(recognizedActivity);
            m_htACLHeap.push(strHashTableKey, aclHeap);
```

Table 6-10. Continued

```
            }

            //Step (8). exmineActivityLifeCycle;
            //Step (9). if there is an expired tentative activity, then remove it from m_htACLHeap
            //Step (10). if there are expired or completed activities ,
            //then remove their children components from the graph recursively;
            Vector<Activity> vtRecognizedActivity = new Vector<Activity>();
            Vector<Activity> vtRemovedActivity = new Vector<Activity>();
            //for (int i = 0; i < m_htACLHeap.size (); i++)
            {
                    Vector<AlcHeap> alcHeap  = m_htALCHeap.elementAt(i); //activity life cycle history of activity [i]
                    vtRecognizedActivity  = evaluate (aclHealp, vtRemovedActivity);
                    vtRemovedActivity = alcHeap.collectFinishedActivities();
            }

            //Step (11). updateGraph
            for( i = 0; i < vtRemovedActivity.size(); i++)
            {
                    Activity activity = vtRemovedActivity.ememtAt(i);
                    Set<CompositionalComp> setAction = lookupChildrenComponents(activity);
                    Set<CompositionalComp> setOperation = lookupChildrenComponents(setAction);
                    Set<PrimitiveComp> setPC= lookupChildrenComponents(setAction, setOperation);
                    removeFromGraph(activity);
                    removeFromGraph(setAction);
                    removeFromGraph(setOperation);
                    removeFromGraph(setPC);
            }

            //Step (12). print recognized activities
            for( int i = 0; I < vtRecognizedActivity.size(); i++)
            {
                    Activity activity = vtRecognizedActivity.ememtAt(i);
                    printf(activity.m_dStartTime, activity.m_dLastUpdateTime, activity.m_strName);
            }
            t++;
        } //end of while
    } //end of function
```

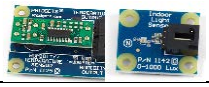
6.5 Validation of Fuzzy Logic Based Activity Recognition Algorithm

For validating the fuzzy logic based activity recognition (AR) approach and estimating the performance, we developed four activity recognition algorithms and compared the their accuracy. Four algorithms are classified according to algorithm type and whether they utilized activity semantics or not. They are FL (Fuzzy Logic), FL+S (Fuzzy Logic and Semantic knowledge of activities), MLNNK (Multi Layer Neural NetworK), and MLNNK+S (Multi Layer Neural Network and Semantics of activities).

6.5.1 Experiment

Activity dataset for this validation was collected at Gator Tech Smart House at the University of Florida. Among several collected activity datasets, “MakingHotTea” activity datasets that were generated by three residents for four days were used for this experiment. For collecting “MakingHotTea”, Bluetooth enabled RFID devices, android smart phones, and several sensors were used. The Bluetooth enabled RFID reader shown in Table 6-11 was well suited for collecting the activity data set, its light weight (75g) ensures that it is wearable and it has a range of 50cm [73], which is sufficient for the experiment. In total, “MakingHotTea” was performed 17 times. Some of them are only partially performed to see how activity recognition algorithm can recognize these partially performed activities. They are 11 cases out of 17 cases and actors decided randomly if they performed the activity partially or completely.

Table 6-11. Sensors and devices used to collect “Making hot tea” activity data

Sensors and devices used		Installed location
Android phone		Portable
RFID reader [73] (Bluetooth enabled)		Wearable device
RFID Tags		Kettle, Bottles, Cups, Utensils, etc.
Touch sensor [74]		Stove switch
Sensor interface board [74] Temperature sensors [74] Humidity sensors [74]		Kitchen

6.5.2 Comparison and Analysis of Experiment Results

In this validation, the accuracy of four activity recognition algorithms is compared. According to equations in (6-8) and (6-9), activity accuracy and time slice accuracy are computed. Activity accuracy represents how accurately a target activity is recognized. Time slice accuracy represents the number of time slices that are correctly recognized

out of all time slices in which the activity is performed. To compute accuracy, we measure the following four cases.

- TP (True Positive): the number of activities that are performed and recognized.
- TN (True Negative): the number of activities that are neither performed and nor recognized.
- FP (False Positive): the number of activities that are recognized by the system, even though they are not really being performed.
- FN (False Negative): the number of activities that are performed, but not recognized.

$$\text{Accuracy}_{\text{Activity}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6-8}$$

$$\text{Accuracy}_{\text{Timeslice}} = \frac{\sum_{i=1}^{i=n} TP}{N} \tag{6-9}$$

Where, N is the total number of time slices.

Table 6-12 and Figure 6-9 compare the activity accuracy for activity recognition algorithms.

Table 6-12. Accuracy for activity classes

Algorithm	Activity	TP	TN	FP	FN	Accuracy
Fuzzy Logic	WashingDishes (Not Performed)	0	17	0	0	1
	MakingHotTea	2	6	0	9	0.47
	MakingHotTea (Completed)	5	9	2	1	0.82
Fuzzy Logic & Semantics	WashingDishes (Not Performed)	0	17	0	0	1
	MakingHotTea	4	6	0	7	0.59
	MakingHotTea (Completed)	5	10	1	1	0.88
Multi Layer Neural Network	WashingDishes (Not Performed)	0	17	0	0	1
	MakingHotTea (Partially Performed)	0	6	0	11	0.35
	MakingHotTea (Completed)	5	8	3	1	0.76
Multi Layer Neural Network & Semantics	WashingDishes (Not Performed)	0	17	0	0	1
	MakingHotTea(Partially Performed)	4	6	0	7	0.59
	MakingHotTea (Completed)	5	10	1	1	0.88

All AR algorithms recognized well that "WashingDish" is not actually performed even though some sensors are used for both "WashingDish" and "MakingHotTea". FL shows better accuracy for both completed and partially completed activities. When activity semantics are used, both FL+S and MLNNK+S show high accuracy. For all algorithms, partially performed activities show lower accuracy than completed activity.

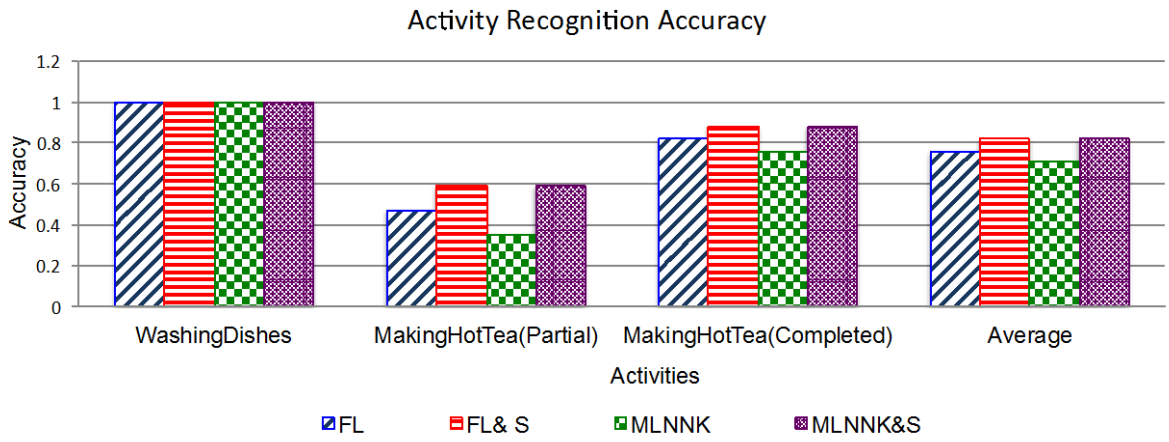


Figure 6-9. The comparison of activity recognition accuracy for algorithms

Table 6-13 and Figure 6-10 compare time slice accuracy for four activity recognition systems. In Table 6-13 and Figure 6-10, fuzzy logic based AR algorithms show higher time slice accuracy than MLNNK based algorithms even though fuzzy logic based algorithm and MLNNK based algorithm show comparable accuracy when they utilize activity semantics in Figure 6-8. This difference of time slice accuracy results implies how consistently an AR algorithm recognizes an activity during the period in which the activity is performed. MLNNK algorithm shows markedly low time-slice accuracy when it does not utilize activity semantics. The higher accuracy of fuzzy logic based algorithm may be attributed to its ability to recognize ambiguously performed activities.

Table 6-13. Time Slice Accuracy

Dataset #	Number of Time Slices	FL	FL & Semantics	MLNNK	MLNNK & Semantics
1	13	13	13	3	13
2	9	7	7	2	7
3	2	2	2	0	2
4	6	3	6	1	4
5	3	3	3	1	3
6	7	6	7	2	6
7	4	0	0	0	0
8	3	0	0	0	0
9	4	0	4	0	0
10	3	3	3	0	0
11	1	0	0	0	0
12	3	0	0	0	0
13	6	4	6	1	4
14	18	16	16	2	16
15	2	2	2	0	2
16	5	5	5	1	5
17	2	0	0	0	0
Total	91	64	74	13	63
Accuracy	1.00	0.70	0.81	0.14	0.68

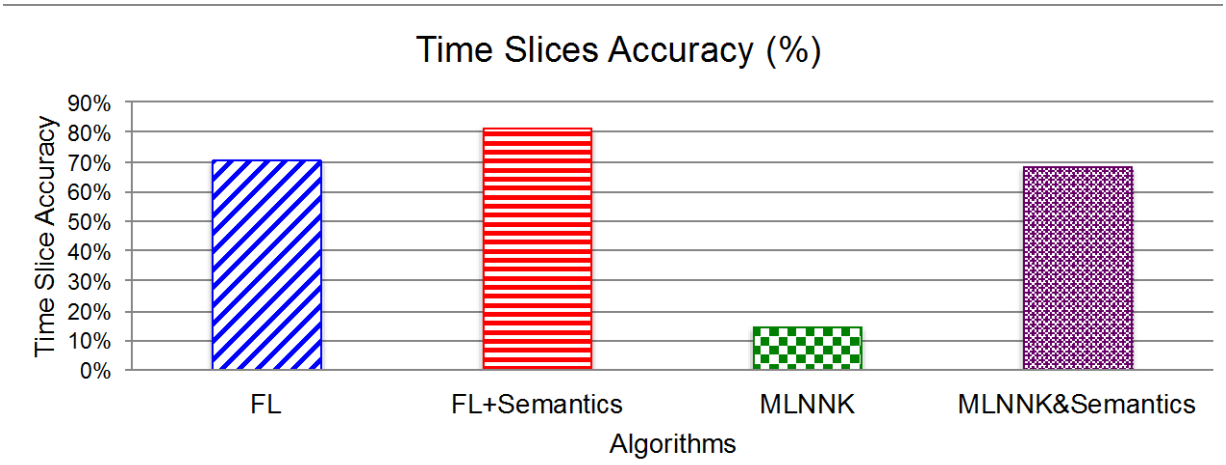


Figure 6-10. The comparison of activity recognition accuracy for algorithms

6.6 Summary and Discussion

To support SGAF activity model, fuzzy logic based activity recognition algorithms are developed. Fuzzy logic based AR algorithm can deal with ambiguous characteristics of activity, activity knowledge, or activity observation data. Also, it uses knowledge

inference to achieve high tolerance to uncertainty in activity. Therefore, the developed algorithm will significantly improve the performance of activity recognition.

Moreover, the new fuzzy logic based AR algorithm does not require training because it computes weights between activity components using the activity model. It can improve the programmability of activity recognition system significantly. Through experimental validation, we proved that our algorithm shows comparable or better accuracy than multi layer neural network based AR algorithms.

CHAPTER 7
UNCERTAINTY ANALYSIS IN ACTIVITY RECOGNITION

A well-known phrase “You can’t manage what you can’t measure!” represents the importance of uncertainty measurement [62]. There are instances of time when limited knowledge exists and it is impossible to accurately describe and recognize an activity. It is important to minimize these uncertainties. However, a straightforward approach to eliminate or accommodate every instance of uncertainty has not been developed yet. Therefore, current AR systems suffer from a convoluted accuracy problem due to the many sources of uncertainties that could profoundly affect the system. From a conceptual point of view, if the impact of uncertainty across the activity recognition system could be reduced, the overall accuracy of the AR could potentially be improved significantly. Additionally, it is clear that high uncertainty sources need to be addressed first. To prioritize the uncertainties according to their impact, quantification of the sources of uncertainty is required. It is therefore essential to identify the most detrimental uncertainty sources and measure their impact on activity recognition system. Current accuracy measuring system does not address this issue satisfactorily.

Many researchers quantify accuracy by comparing the activities recognized by their system and the activities that were actually performed and manually observed. As shown in Figure 3-1 in Chapter 3, an AR system involves several subsystems and techniques; this complexity further obscures the detection of uncertainty sources. Hence, it is necessary to develop the theory and diagnostic tools for accuracy measurement and the identification of sources of uncertainty. To address the problem, our focus in this chapter is the identification of uncertainty sources and quantitative analysis of the impact of uncertainties in an activity recognition solution. We account for

all sources of uncertainties in Section 7.2. Integrated uncertainty measurement through developed metrics is introduced in Section 7.3. Experiment and analysis are described in Section 7.4.

7.1 Goal: Developing More Practical Metrics and Measures of Accuracy

There are several challenging goals that need to be addressed to quantify the impact of uncertainty sources and subsequently adapt and improve the overall system.

First, we identify all possible uncertainty sources in an AR system. it is important to identify all major uncertainty sources to guarantee that quantitative analysis is meaningful.

Second, adequate measurements should be taken for quantifying the impact of each uncertainty source. Due to the diversity of uncertainty sources and their characteristics, multiple measures and measurement methods will need to be developed for each source. For example, the measurement method for uncertainty in the activity model is (as we will show) quite different from the method for measuring uncertainties in the sensor data. Rigorous criteria for selecting the most appropriate measurement methods need to be developed.

Third, through impact analysis, we find and eliminate uncertainty sources that have the most detrimental impact on AR performance. By only computing the uncertainty metrics, it is difficult to provide adequate information for finding the uncertainty source with the highest impact. In other words, high uncertainty may not mean high impact. For instance, low uncertainty in an activity model may have a higher impact than high uncertainty in activity sensor data. Therefore, impact analysis of uncertainty sources is important.

7.2 End-to-End Analysis of Various Uncertainty Sources

In this section, we analyze possible sources of uncertainty in activity recognition. We identify all possible uncertainty sources in an AR system. A summary list of such uncertainty sources is depicted in Figure 7-1 along with the various steps and phases of designing and implementing an AR system.

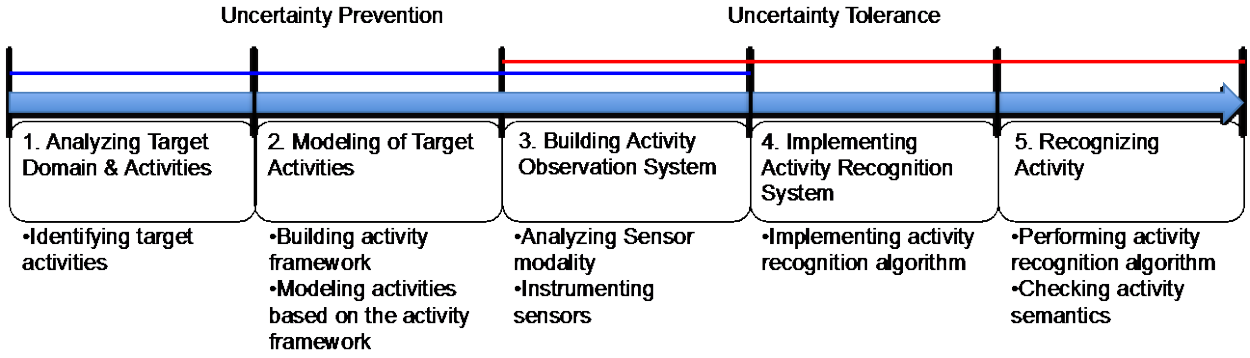


Figure 7-1. Process flow of activity recognition and uncertainty sources in activity recognition system.

Activity recognition (AR) system is composed of activity observation sub system and activity recognition sub system. Activity observation system collects activity sensor data. Activity recognition system classifies activities from the sensor data. These sub systems utilize domain modeling technology, sensor technology, and sensor network technology. We have identified 12 sources of uncertainty and subsequently classified them into three categories.

7.2.1 Uncertainties in Human Activities

Because human beings are highly intelligent, activities performed by them have concurrent, interleaving, ambiguous, and diverse characteristics. Additionally, if multiple people perform an activity together, activity recognition becomes more complex because the detected activity must be associated with more than one resident. These characteristics increase the uncertainty in detecting a performed activity. The

complexities of human activities are a consequence of the intricacies of our intelligence. The detailed description of concurrent, interleaving, ambiguous, and diverse characteristics of activities are given in Section 1.3.1.

7.2.2 Uncertainties in Sensor Technology

An AR system is an integrated result of several related technologies. If the technologies themselves contain uncertainty, then the AR system that uses them will be influenced from this limitation. Uncertainty may exist in the following components:

- Sensor technology. Activity sensing ability is limited by the underlying sensor technology. In other words, the accuracy and richness of the activity data collected depends on the specifications and tolerances of the sensors. For example, many sensors such as RFID, sound or motion sensors can be used to detect an "eating" activity. Nevertheless, gathering the details related to nutritional intake is not straightforward. Even though there are sensors that provide some of the necessary details, many sensors have limited usability in terms of size and modality.

- Sensor network. Given that people move from one place to another while performing activities, wireless networks such as Bluetooth and Wi-Fi are used to collect the sensor data. Nevertheless, due to the instability of the connection within the network, data packets may be lost.

- Limited energy of sensor technology. Maintaining sensors in an active state requires energy consumption. The sensor network itself also consumes energy. Nevertheless, limited energy can lead to uncertainty in activity data if it affects the performance of the measurement process or limits the transmission of the messages.

7.2.3 Uncertainties in AR Techniques

Existing AR techniques are themselves major sources of uncertainty. Careful examination and analysis of these techniques are required to understand the nature of their uncertainties and to improve overall recognition performance.

- Activity domain analysis. Defining a target domain is important for AR given that activities are different in different domains. For example, activities conducted indoors are different from activities conducted outdoors. Another possible source of uncertainty is missing target activities in a domain; it is not easy to identify every activity in a domain in a few attempts.

- Activity modeling. Once target activities are determined, it is necessary to represent them. Different activities have different features to characterize them. The activity model should represent these characteristics. Nevertheless, current activity modeling techniques are missing many of the features because it is not easy to capture and represent all features of human activities in an activity model.

- AR algorithms. AR algorithms have limited performance due to several challenges primarily the wide variability inherent in performing activities, and the complexity of tracking multiple overlapping activities simultaneously. Also, AR algorithms could perform better if they utilize domain knowledge [11]. But even then, the lack or the degree of adequacy of such knowledge would impact uncertainty.

- Sensor function and modality analysis for target activities. Before installing sensors in a real space, sensor function and sensor use modality should be analyzed to find the effective sensor sets for the target activities. In other words, it is necessary to decide the information, which needs to be sensed, the number of sensors, and the type of sensors required. Also sensor use modality which addresses the feasibility of its use, where it is worn, or the location it is placed at is very important and does account for another source of uncertainty in AR systems.

Table 7-1. Possible sources of uncertainties

	Source of uncertainty	Potential uncertainties
Human activities	Concurrently performed activities	Mixed activity dataset of multiple activities by a person
	Interleaving activities	Blended activity dataset of multiple activities by a person
	Ambiguity of activities including partially performed activities	Ambiguity Partially collected activity dataset
	Multiple people	Mixed activity dataset of multiple activities by multiple people
	Variety of ways for an activity	Missing some of the variety ways for an activity when analysis and design the activity
Sensor technology	Sensor	No adequate sensor for an activity
	Sensor network	Sensor data loss
	Limited energy	Sensor failure and sensor data loss
Activity recognition techniques	Sensor analysis for target activities	Casually analyzed sensor functionality and modality analysis
	Activity domain analysis	Missing activities in a domain
	Activity modeling	Inaccurate activity model or Insufficient representation of activity semantic characteristics
	Activity recognition algorithm	Lack of intelligence of algorithm such as inference or knowledge Lack of robustness in face of activity performance variability Lack of capability to track multiple simultaneous activities

The aforementioned uncertainty sources are summarized in Table 7-1. Among the uncertainty sources, human activities are related to Type A uncertainty. Other uncertainty sources are related to Type B uncertainty that are eliminated or decreased.

7.3 Uncertainty Metrics and Measures

The following metrics are developed to measure uncertainties. Wherever possible, we have attempted to utilize existing metrics and measures for characterizing uncertainty sources. In addition to the existing metrics, we introduce a range of new metrics and their measures as necessary for uncertainties, which have not been considered before.

7.3.1 Certainty Factor

The certainty factor was adapted as the uncertainty metric of the activity model. The certainty factor (CF) is a very effective evaluative analysis technique used in several areas such as diagnostics and medicine. To compute the certainty factor of an activity model, we measure the degree of belief and disbelief of the model. The detailed description of Belief and Disbelief is given in Section 2.4.

$$\text{CertaintyFactor (CF)} = \text{Belief} - \text{Disbelief} \tag{7-1}$$

7.3.2 Correlation Degree of Sensors and Activities

To recognize an activity, selected sensors are chosen. If there are multiple sensors per activity, the collected sensor data will have a higher degree of certainty than a single sensor given that a single sensor working in isolation is vulnerable to failure. Similarly, if a single sensor is used for detecting multiple activities (or one activity per sensor), the sensor will have a higher level of efficiency. Nevertheless, the

evidential power of the sensor will be low. Therefore, the degree of correlation between sensor data and activities is presented in Equation (7-2).

$$\text{Sensor - Activity Correlation} = \frac{1}{M}\sum_{j=1}^{M}\text{The number of related activities}\left(\text{sensor}_j\right) \quad (7\text{-}2)$$

Where M is total number of sensors.

7.3.3 Sensor Data Certainty

Loss of sensor data due to for example sensor data failure and sensor network failure results in high uncertainty. Therefore, sensor data certainty is related to the missing rate of sensor data, which can be expressed as presented in Equation (7-3).

$$\text{Sensor Data Certainty} = 1 - \frac{\text{The amount of missing data}}{\text{Total amount of data}} \quad (7\text{-}3)$$

7.3.4 Activity Certainty

Concurrent, interleaving, ambiguous performance of multiple activities is more complex and uncertain than that of a single activity. The certainty can therefore be expressed as:

$$\text{ActivityCertainty} = 1 - \min\left(1, \frac{\sum_{i=1}^{N}\left(C_i \times P_i\right)}{S}\right) \quad (7\text{-}4)$$

Where S is the total number of activities or activity scenes. C_i is a concurrent, interleaving or partially performed activities or scenes and N is the total number of these activities or scenes. P_i is the number of actors of the i^{th} activity or scene.

7.3.5 Metrics for AR Performance

The uncertainties of AR system components contribute directly to the overall uncertainty of the AR performance (ARP). There are several well-established ARP

metrics in AR research. The following measures are often used for computing such metrics:

- True positive (TP): the number of positive recognition for performed activities
- True negative (TN): the number of negative recognition for non-performed activities
- False positive (FP): the number of positive recognition for non-performed activities
- False positive (FN): the number of negative recognition for performed activities

Using the ARP measures, the following ARP performance metrics are computed. These metrics are frequently used to quantify the performance of an AR system. For the same TP, recall depends on FN whereas precision is affected by FP.

Accuracy for ARP. The accuracy of an AR system determines its overall performance. As shown in Equation (7-5), accuracy accounts for both false positive and false negative data [49, 56].

$$\text{ARP Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{7-5}$$

Recall (Sensitivity). Recall of an AR system determines how well it can recognize performed activities. Therefore, recall accounts for false negative recognition as shown in Equation (7-6); false positive recognition does not affect recall [49, 56].

$$\text{ARP Recall} = \frac{TP}{TP + FN} \tag{7-6}$$

Precision. Precision is the ratio of the number of true positive recognitions to the number of all positively recognized activities. Therefore, precision does not account for false negative recognition as shown in Equation (7-7) [49, 56].

$$\text{ARP Precision} = \frac{TP}{TP + FP} \tag{7-7}$$

Specificity. Unlike recall or precision, which are a measure of true positive recognition, specificity is the ratio of the number of true negative recognitions to the number of all non-performed activities. Referring to Equation (7-8), specificity is affected by false positive recognitions of an AR system [49, 56].

$$\text{ARP Specificity} = \frac{TN}{TN + FP} \tag{7-8}$$

7.3.6 Uncertainty Impact Comparison

To find the uncertainty that has the most powerful impact, we should compare metrics of uncertainty sources. The problem is that even though we compute uncertainty metrics, their impact may be different. For example, if we compare two activity models that have different levels of uncertainty, we can easily expect that higher uncertainty will produce low activity recognition (AR) accuracy. Nevertheless, if we need to compare two different kinds of metrics (e.g. activity model and activity dataset), it is not easy to determine which one has the higher impact on uncertainty than the other. The impact of an uncertainty source is determined by Equation (7-9).

$$\text{Impact}\left(ARP, U_{source}\right) = \frac{\left|ARP_1 - ARP_2\right|}{\left|U_{1,source} - U_{2,source}\right|} \tag{7-9}$$

Where,

- Impact (ARP, U_{source}): the impact of uncertainty U of an uncertainty source on a metric ARP.
- ARP_1, ARP_2: AR performances such as accuracy, recall, precision, or specificity (e.g. accuracy1 and accuracy2).
- $U_{1,source}$, $U_{2,source}$: uncertainties in the activity model, activity dataset, or activities, (e.g. certainty factor1 for $U_{1,\text{activity model}}$, certainty factor2 for $U_{2,\text{ activity model}}$ if uncertainty source is an activity model).

7.4 Validation of Uncertainty Analysis Approaches

The goal of validation in this section is finding the most debilitating uncertainty source through uncertainty measurement. For experiment, three activity scenarios are created based on daily living scenarios [6]. Each scenario corresponds to simple, concurrent, or interleaving activities, respectively as shown in Figure 7-2.

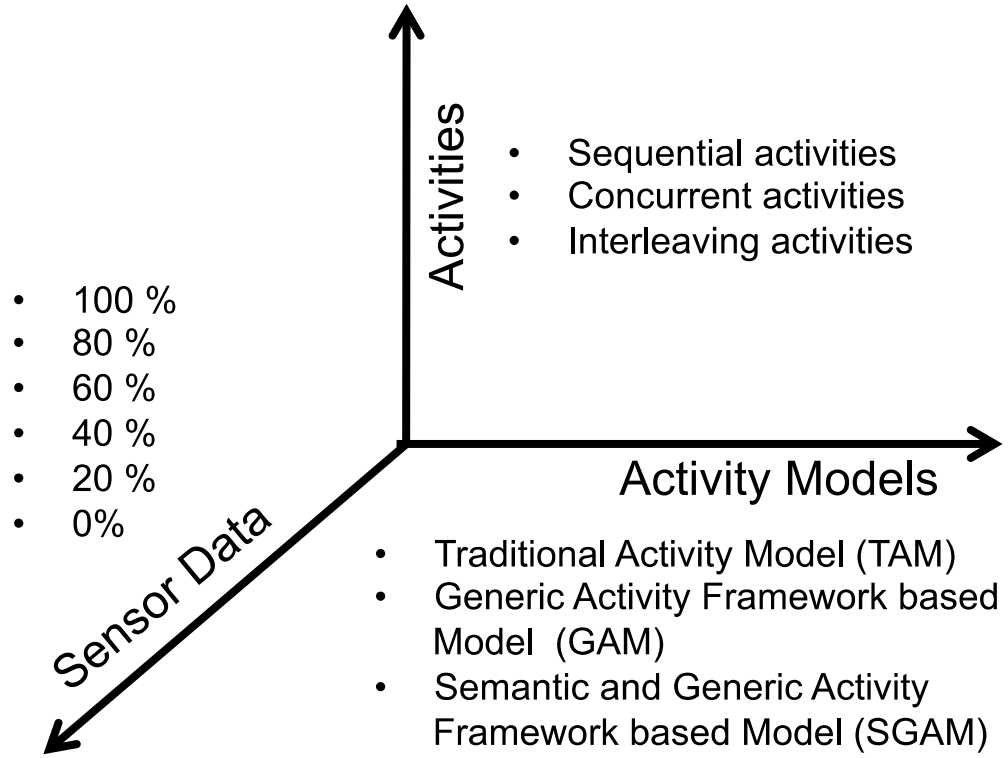


Figure 7-2. Three major uncertainty sources in an activity recognition system.

7.4.1 Experiment

Our experimental environment was established utilizing seven steps in which: (1) target activities are identified, (2) for the target activities, three activity models with different certainty are created, (3) with different combinations of the target activities, three activity scenarios are generated, (4) sensors are instrumented to detect activities, (5) Activity sensor data sets are collected, (6) Three activity recognition algorithms

according to three activity models are implemented and (7) AR algorithms are performed and their results are analyzed. We describe each of these steps in detail in the following sections.

Step 1. Identifying target activities. To establish an activity scenario for comparative purpose, we used the eldercare scenarios described in [6]. The scenario explains daily living activities of an 87-year-old lady who is living alone. From the scenarios, we retrieved target activities such as “Watching TV” and “Cooking”. An excerpt of the target activities is presented in Table 7-2.

Step 2. Designing activity models. For each target activity, we identified components such as actions, operations, motions, tools, objects, and related components including time or location as shown in Table 7-2. We designed three activity models: TAM, GAM, and SGAM based on our generic activity framework [11,17]

- Traditional Activity Model (TAM): TAM model is designed based on activity theory. In activity theory, tools used by people are considered as important pieces of evidence to recognize the performed activities.
- Generic Activity Model (GAM): this activity model is based on a generic activity framework developed in [9]. The generic activity framework expands upon the activity theory by clarifying further activity components and also their compositional structure.
- Semantic Generic Activity Model (SGAM): SGAM model is also designed based on the generic activity framework like GAM model. The difference between the GAM model and the SGAM model is that the SGAM model uses activity semantics, which is human knowledge about activities [11].

Table 7-2 presents the activity components in the three activity models. The detailed descriptions of these three models have been reported in [9].

Table 7-2. Target activities and activity components.

Activities (Location)	Action Operation	Tool	Object	Activity Semantics
Entering Leaving home	Opening a front door	No tool	Front door	Dominance: -Key: Front door Mutuality: -Exclusive activity Order: Entering home->other activities->Leaving home
Watching TV	-Turning on TV	No tool	TV	Dominance: -Key: TV Effect: light, sound Mutuality: Concurrent
Phone call	-Dialing a phone -Holding a phone -Talking on the phone	Phone	No object	Dominance: -Key: Phone
Cooking (Kitchen)	-Preparing food items -Cutting food items -Heating food -Serving food	-Knife -Microwave -Pan -Plates	-Groceries -Refrigerator -Freezer	Weak order: -Taking food items -> Heating food -Cooking->Eating
Eating (Dining room)	-Cutting food -Using spoon or fork -Chewing -Swallowing	-Knife -Spoon, Fork	-Food	Weak order: -Eating -> Washing dishes
Drinking	Taking drink	Cup	Refrigerator	Dominance: Key: Cup
Washing dishes	-Turning on water -Washing dishes	Dishwasher	-Pans -Cup -Dishes	Weak order: -Eating -> Washing dishes

Step 3. Activity scenarios. Three activity scenarios were built, each with a different level of complexity.

- Scenario 1: Sequential activities

In scenario 1, activities are performed sequentially with no concurrent or interleaving activities. Table 7-3 presents the activities comprising Scenario 1. Each activity is named and is described in terms of the sequence of actions that make up the activity.

Table 7-3. Sequential activity scenario

Activity scene #	Activities	Sequence of actions
1	Entering home	1.1 Opening an entrance door
2	Phone call	2.1 Holding a phone 2.2 Dialing phone numbers 2.3 Talking 2.4 Disconnecting a phone call & Hanging up a call
3	Watching TV	3.1 Turning on 3.2 Watching TV 3.3 Turning off
4	Cooking	4.1 Taking out food item from refrigerator 4.2 Washing the food items if necessary 4.3 Cutting, Chopping, Stirring (any order) 4.4 Boiling or Baking and Serving
5	Eating	5.1 Cutting, Picking, Scooping and Chewing, Swallowing food 5.2 Drinking water Repeat randomly 5.1 ~ 5.2
6	Washing dishes	6.1 Turning on water facet 6.2 Washing dishes using a sponge to wash utensils (knife, fork, spoon) and dishes
7	Leaving home	7.1 Opening an entrance door and staying outside for some time

- Scenario 2: Concurrent activities

In Scenario 2, multiple activities are performed concurrently. For example, "Watching TV" can be performed as a concurrent activity with activities such as "Phone call", "Cooking", or "Eating". Table 7-4 presents examples of concurrently performed activities from scene 3 to activity scene 7.

- Scenario 3: Interleaving activities

In Scenario 3, interleaving activities are performed. For example, "Washing dishes" and "Entering/Leaving home" are performed in an interleaving manner. There are two interleaving activity scenes in Table 7-5. In the table, "Washing dishes" and "Watching TV" are interrupted three times and one time by "Entering/Leaving home" respectively.

Table 7-4. Concurrent activity scenario with “phone call” and other activities

Activity scene #	Activities	Sequence of actions
1	Entering home	1.1 Opening an entrance door
2	Phone call	2.1 Holding a phone 2.2 Dialing phone numbers 2.3 Talking
3	Phone call Watching TV	While talking on the phone, 3.1 Turning on 3.2 Watching TV 3.3 Changing TV sound volume 3.4 Making a phone call & talking on the phone
4	Phone call Watching TV Cooking	While talking on the phone and watching TV 4.1 Taking out food item from refrigerator 4.2 Washing the food items if necessary 4.3 Cutting, Chopping, Stirring (any order) 4.4 Boiling or Baking and Serving
5	Phone call Watching TV Eating	While talking on the phone and watching TV 5.1 Cutting, Picking, Scooping, and Chewing, Swallowing food 5.2 Drinking water
6	Phone call Watching TV Washing dishes	While talking on the phone and watching TV 6.1 Turning on water facet 6.2 Using a sponge to wash utensils (knife, fork, or spoon) and dishes
7	Phone call Watching TV Leaving home	While talking on the phone and watching TV 7.1 Opening an entrance door and staying outside

Table 7-5. Interleaving activity scenario with “Entering/leaving home” and other activities.

Activity scene #	Activities	Sequence of actions
1	Entering home Washing dishes Leaving home	1.1 Entering home 1.2 Washing dishes 1.3 Leaving home 1.4 Entering home 1.5 Washing dishes 1.6 Leaving home 1.7 Entering home 1.8 Washing dishes 1.9 Leaving home
2	Entering home Turning on TV Leaving home	2.1. Entering home 2.2 Turning on TV 2.3 Leaving home 2.4 Entering home 2.5 Turning on TV 2.6 Leaving home

Step 4. Smart space instrumentation. We utilized the smart spaces at the Smart Environments Research Group (SERG) in University of Ulster for the purposed of

activity dataset collection. The utilized smart space is composed of four sub spaces: living room, kitchen, meeting room, and robotics and wearable technologies lab. [21]. Figure 7-2 shows the living room and kitchen where our experiments were performed [21].

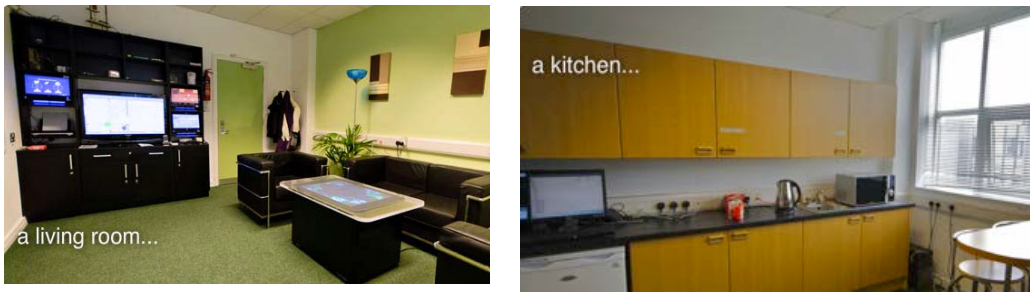


Figure 7-2. A smart space consisting of a living room and a kitchen.

The sensors used for detecting activities included 10 accelerometer sensors and one light sensor those are connected to our measurement system through sun SPOT (Small Programmable Object Technology) sensor platforms [83]. Additionally, four Tynetec contact sensors were utilized [84]. The detailed sensor instrumentation is explained in Table 7-6.

Table 7-6. Instrumented sensors for sensing activities

Sensors	Instrumented location	Activities
Sun SPOT accelerometer 	TV	Watching TV
	Arm	All activities
	Phone	Making a phone call
	Plates	Cooking, Eating,
	Knife, Fork, Spoon	Washing dishes
Tynetec touch sensor 	Front door	Entering/leaving home
	Refrigerator	Cooking, Washing dishes
	Microwave	
	Water facet	

Step 5. Activity dataset collection. After instrumenting sensors in the target smart space, we collected activity datasets automatically using a sensor network, sensor boards, and a computer. Activities were performed according to the three

scenarios described in Step 3. A 27-year-old male from the SERG laboratory in the Ulster University performed the activities over a period of one week. He performed the same activities several times to ensure that it was possible to select three examples per each scenario. Therefore, we collected nine datasets in total. The annotation for the performed activities was undertaken by the Dante system, which is a video camera based automatic annotation system [30]. In addition to the automatic annotations, the activities were manually annotated. Three out of the nine sets of data were selected for further analysis.

Step 6. Activity recognition system implementation. We implemented three activity recognition systems according to the activity models. All activity recognition systems were implemented using a Multi Layer Neural Network (MLNNK). We chose a MLNNK as a groundwork algorithm given that it can support all activity models such as TAM, GAM, and SGAM. Also, MLNNK is less complex than other algorithms such as HMM or CRF which account for too many horizontal and hierarchical relationships between activity components. MLNNK addresses this problem by accounting for hierarchical relationships only. The detailed description of the MLNNK based AR algorithm is given in [10].

Step 7. Executing the AR algorithms. Through step 1 to step 6, we prepared three activity scenarios, three activity models, activity datasets of the different activity scenarios and activity algorithms. We built 45 possible combinations of them as shown in Table 7-10, Table 7-11 and Table 7-12. For example, one possible case is composed of scenario 1, TAM activity model and algorithm and activity dataset whose data missing rate is 20%. For each case, we executed three AR algorithms that were implemented in

the previous step. The execution results of the AR algorithms are compared with the annotation in Step 5.

7.4.2 Measuring and Comparing Uncertainties

In this section, we measure uncertainties in individual uncertainty sources in the AR procedural steps described in Section 7.4.1. The uncertainties in the activity models, performed activity scenarios, collected activity data and the performances of three AR algorithms are measured. Table 7-7 and Table 7-8 present the measurement results of activity model certainty and Table 7-9 presents the measurement results of activity model certainty.

- Certainty factor of activity models

The certainty factor of an activity model is measured using the method introduced in Section 7.3.1. To compute a CF the probabilities of the hypothesis p(H) and the conditional probability p(H|E) need to be determined. For probability computation, we utilize the activity models presented in Table 7-2. For the example “Entering/Leaving home”, there are three kinds of evidence—“Front door”, “Opening/Closing” and semantics such as activity order in the Table. From the three pieces of evidence, we identify six possible cases of sensor events—{(Front door), (Motion), (Front door, Motion), (Front door, Semantics), (Motion, Semantics), (Front door, Motion, Semantics)}. If (Front door, Motion, Semantics) is detected, which means both “Front door” and “Opening/Closing” motions are detected and activity semantics are satisfied, “Entering/Leaving home” activity is assumed to be truly performed. We exclude two cases where —{(null), (Semantics)}—from possible combinations because neither “Front door” nor “Opening/closing” is detected. According to this idea, probabilities of

"Entering/Leaving home" are calculated as shown in Table 7-7. In the Table, hypothesis H is an activity and evidence E is an activity component. For instance, the probability of an evidence of "Front door", p("Front door"), is 4 out of 6 and p(H ∧ E) is 1 out of six. The probabilities of other activities are computed similarly.

Table 7-7. The probabilities of "Entering/Leaving home" activity according to the models.

	Evidence (E)	p(H ∧E)	p(E)	p(H|E)	Sum of p(H|E)
TAM	Artifact (Front door)	0.17	0.67	0.25	0.25
GAM	Tool (no tool)	0	0	0	0
	Motion (Opening/closing)	0.17	0.67	0.25	0.25
	Object (Front door)	0.17	0.67	0.25	0.44
SGAM	Tool (no tool)	0	0	0	0
	Motion (Opening/closing)	0.17	0.67	0.25	0.25
	Object (Front door)	0.17	0.67	0.25	0.44
	Semantics (Order, mutuality)	0.17	0.5	0.33	0.63

Based on Table 7-7, the CFs of an activity are computed. Table 7-8 represents the CF for every activity of three models. Depending on how much a semantic contributes to activity identification, we observe that some semantics are highly evidential whereas some are not.

Table 7-8. Certainty factor metrics of activity models

Activity Name and Activity Model	TAM	GAM	SGAM
Enter/Leaving home	0.33	0.47	0.69
WatchingTV	0.48	0.61	0.87
PhoneCall	0.48	0.74	0.83
Cooking	0.24	0.43	0.62
Eating	0.24	0.43	0.62
WashingDishes	0.23	0.59	0.76
Average	0.33	0.54	0.73

- Activity certainty

Activity certainty is measured according to the method outlined in Section 7.3.4. In this case, the maximum number of activities is six, i.e. the number of target activities. In Table 7-9, activity certainty of scenario 1 is higher than the others given that the

activities are performed sequentially. Scenario 2 is slightly more uncertain than scenario 3.

Table 7-9. Activity certainty of performed scenarios

	Scenario1	Scenario2	Scenario3
Total scenes	7	7	2
Sequential activities	7	2	0
Concurrent activities	0	5	0
Interleaving activities	0	0	2
Average # of activities per step	1	2.29	2
Maximum possible activities	6	6	6
Activity certainty	1	0.79	0.83

- Sensor data certainty

As described in Section 7.3.3, if there are no missing sensor data and no instances of sensor failure, we consider the sensor data to be reliable. Using the collected sensor data, we generated five activity data sets that have a different sensor data-missing rate from 0% to 80%.

Table 7-10. Measured AR performances (Scenario1)

Activity certainty	Model	Activity Data Certainty	Activity Recognition Performances			
			Accuracy	Recall	Precision	Sensitivity
Scenario1 (certainty: 1)	TAM (0.33)	1	0.82	0.87	0.74	0.80
		0.8	0.85	0.92	0.79	0.82
		0.6	0.80	0.82	0.74	0.80
		0.4	0.70	0.25	0.33	0.90
		0.2	0.70	0.37	0.40	0.87
	GAM (0.54)	1	0.93	0.97	0.88	0.92
		0.8	0.92	0.97	0.87	0.90
		0.6	0.93	0.95	0.89	0.92
		0.4	0.92	0.88	0.92	0.96
		0.2	0.88	0.80	0.92	0.96
	SGAM (0.73)	1	0.97	1.00	0.92	0.96
		0.8	0.97	0.97	0.97	0.97
		0.6	0.98	0.95	1.00	1.00
		0.4	0.92	0.88	0.92	0.96
		0.2	0.88	0.80	0.92	0.96

- AR performance

To find the AR performance, we executed our AR systems for every combinational case. There are in total 45 cases according to three activity models, three activity scenarios and five activity data reliabilities. Table 7-10, Table 7-11 and Table 7-12 show the activity recognition performance of each scenario. Their corresponding graphs are shown in Figure 7-3, Figure 7-4 and Figure 7-5 respectively.

Table 7-11. Measured activity recognition performances (Scenario2)

Activity certainty	Model	Activity Data Certainty	Activity Recognition Performances			
			Accuracy	Recall	Precision	Sensitivity
Scenario2 (certainty: 0.82)	TAM (0.33)	1	0.65	0.32	0.49	0.76
		0.8	0.67	0.27	0.44	0.82
		0.6	0.68	0.32	0.56	0.80
		0.4	0.70	0.24	0.38	0.90
		0.2	0.70	0.34	0.43	0.78
	GAM (0.54)	1	0.90	0.70	0.70	0.89
		0.8	0.90	0.70	0.70	0.89
		0.6	0.88	0.67	0.70	0.89
		0.4	0.83	0.49	0.57	0.93
		0.2	0.80	0.42	0.57	0.93
	SGAM (0.73)	1	0.90	0.70	0.70	0.89
		0.8	0.90	0.70	0.70	0.89
		0.6	0.88	0.67	0.70	0.89
		0.4	0.83	0.54	0.57	0.91
		0.2	0.83	0.54	0.55	0.89

Table 7-12. Measured activity recognition performances (Scenario3)

Activity certainty	Model	Activity Data Certainty	Activity Recognition Performances			
			Accuracy	Recall	Precision	Sensitivity
Scenario3 (certainty: 0.71)	TAM (0.33)	1	0.85	0.48	0.69	0.94
		0.8	0.83	0.42	0.56	0.94
		0.6	0.83	0.42	0.58	0.97
		0.4	0.85	0.35	0.63	1.00
		0.2	0.83	0.48	0.63	0.91
	GAM (0.54)	1	0.90	0.60	0.69	0.94
		0.8	0.90	0.54	0.71	0.97
		0.6	0.90	0.48	0.63	1.00
		0.4	0.90	0.48	0.75	1.00
		0.2	0.88	0.42	0.75	1.00
	SGAM (0.73)	1	0.92	0.60	0.71	0.97
		0.8	0.90	0.54	0.71	0.97
		0.6	0.92	0.54	0.63	1.00
		0.4	0.90	0.48	0.75	1.00
		0.2	0.87	0.42	0.75	1.00

In Figure 7-3, Figure 7-4, and Figure 7-5, the performance of AR for scenario1, scenario2, and scenario3 are compared respectively. In every case, the TAM model clearly shows lower performance than the other activity models. The SGAM model shows slightly better accuracy than the GAM model in every scenario. Also, AR accuracy does not depend significantly on sensor data certainty, which is contrary to what we expected. Each figure is analyzed in detail below.

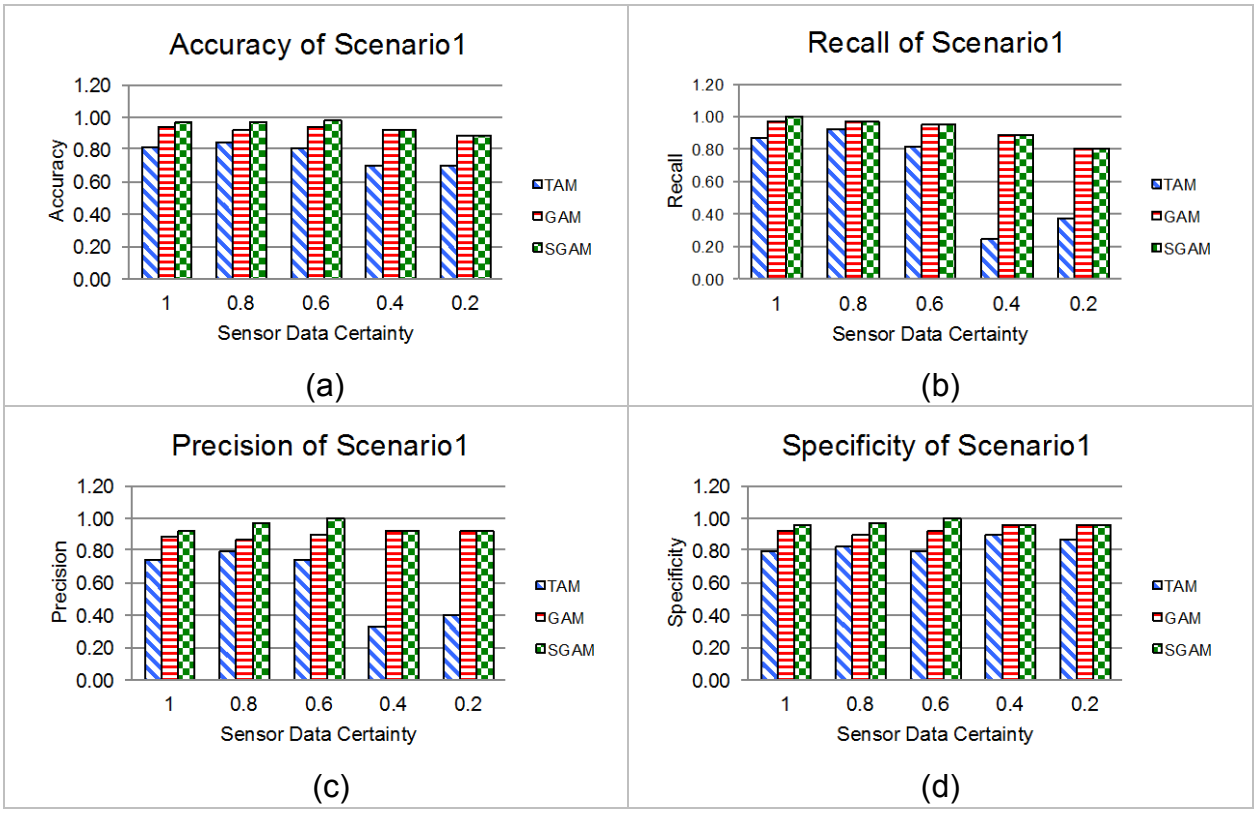


Figure 7-3. Performance comparison of AR system for sequential activities in scenario1 for various activity model and sensor data certainty combinations. (a) accuracy, (b) recall, (c) precision, and (d) specificity.

Figure 7-3 shows the accuracy, recall, precision, and specificity of AR corresponding to scenario 1 for a combination of activity models and activity datasets. Because activities are sequentially performed in scenario 1, performance is primarily determined by the certainties in activity model and activity data set as shown in Figure 7-3. For every case, SGAM shows better performance than the other models. Activity

data certainty may also be related to the overall performance. Nevertheless, when activity dataset certainty is 0.6, accuracy, precision, and specificity are higher than other cases as shown in Figure 7-3 (a), (b) and (c). Therefore, even though AR performance depends strongly on activity data set certainty, they are not always proportional to each other. If the sensor data certainty falls below 40 %, recall and precision decrease drastically because there are very few true positives as shown in Figs. 6 (b) and (c).

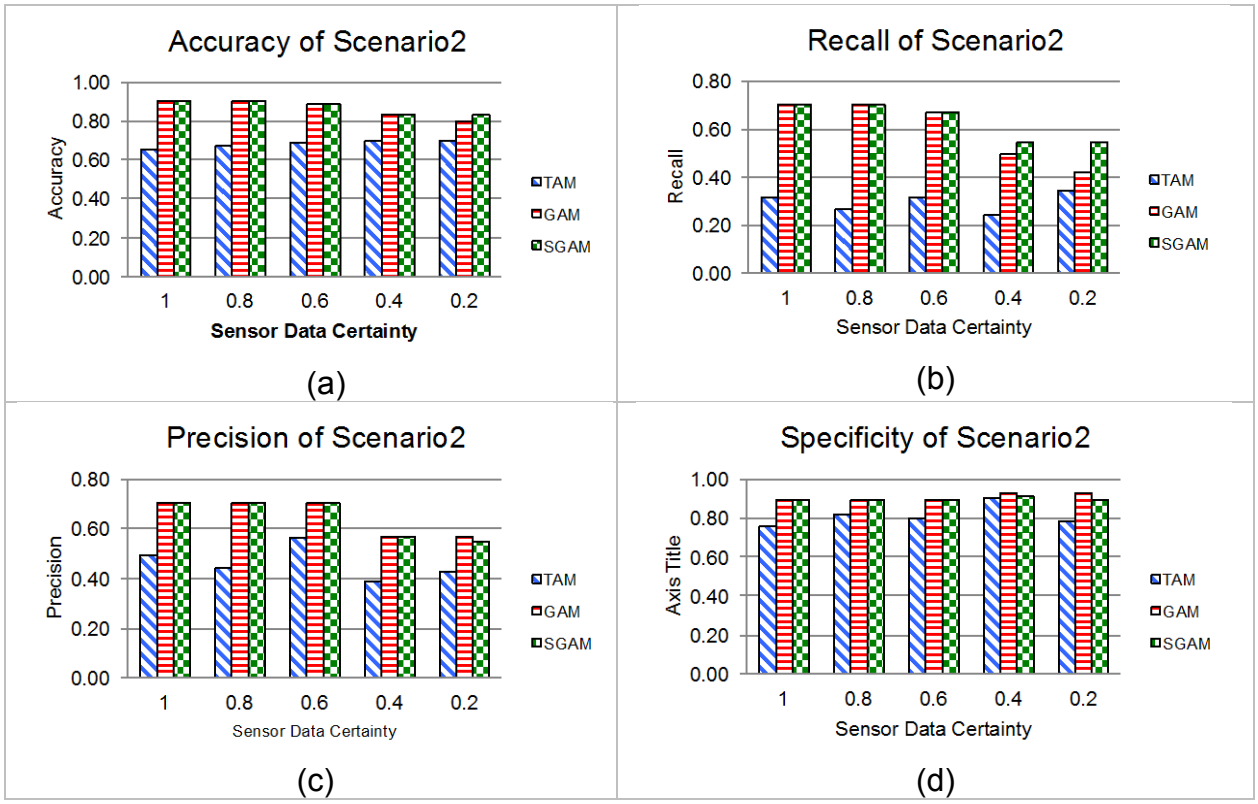


Figure 7-4. Performance comparison of AR system for concurrent activities in scenario2 for various activity model and sensor data certainty combinations. (a) accuracy, (b) recall, (c) precision, and (d) specificity.

Figure 7-4 presents the AR performance for scenario 2. In this scenario, many activities are concurrently performed; therefore, AR performance depends on not only activity model or activity data set but also activity certainty. SGAM in scenario 2 does not outperform GAM much for overall performances compared to scenario 1. However,

SGAM shows better accuracy and recall when sensor data certainty is very low. According to the activity dataset certainty, its accuracy, recall, and precision decrease gradually. Specificity does not show much difference given that specificity depends on the number of true negatives and false positive. It implies that this AR system performed well to distinguish which activities are truly unperformed activities. The Recall of TAM is lower compared with the recall of GAM or SGAM because the TAM output has fewer true positives.

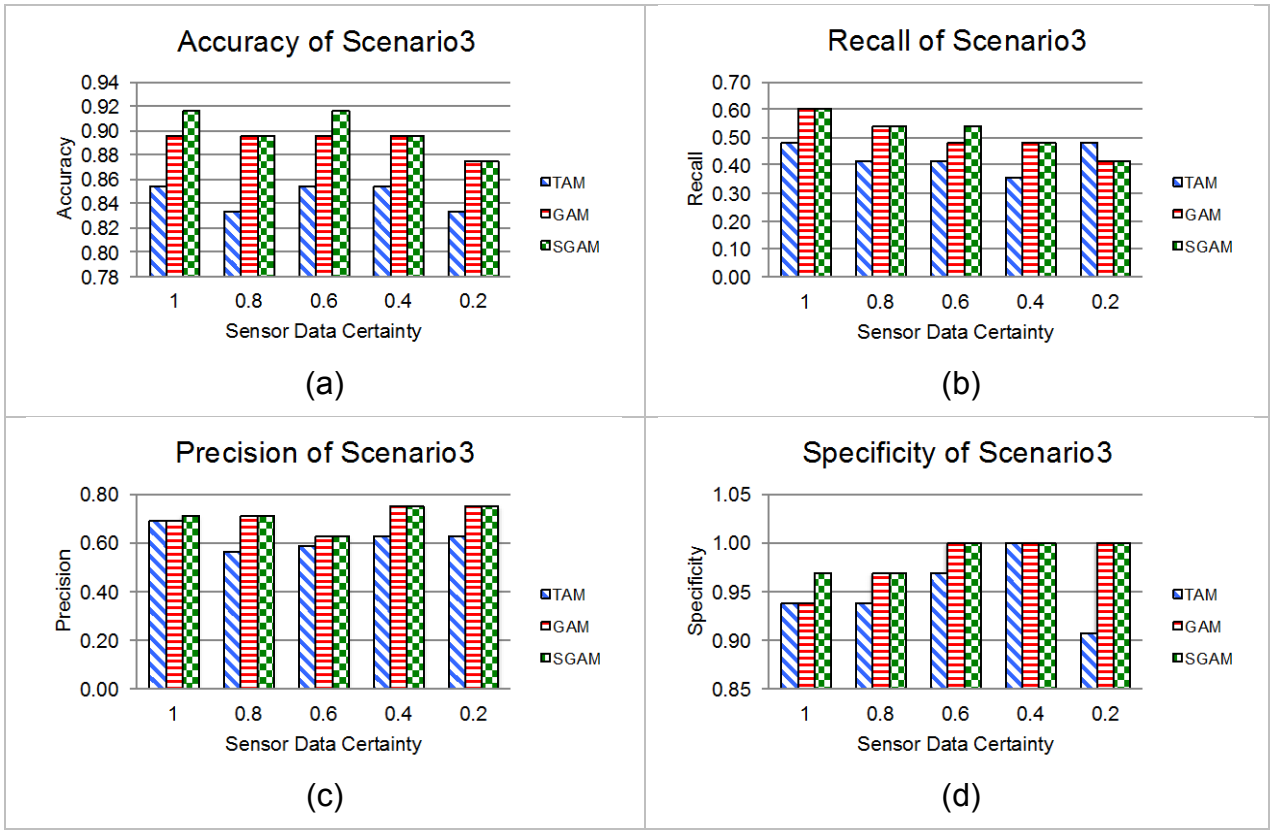


Figure 7-5. Performance comparison of AR system for interleaving activities in scenario3 for various activity model and sensor data certainty combinations. (a) accuracy, (b) recall, (c) precision, and (d) specificity.

The activity recognition performance for scenario 3 is shown in Figure 7-5. The SGAM model has better performance compared to GAM when activity data certainty is

greater than 0.6 in Figure 7-5 (a), (b) and (d). When sensor data certainty is below 0.4, SGAM and GAM activity models show similar performance.

7.4.3 Comparison and Analysis of Uncertainty Impacts

Figure 7-6 shows the average impact of uncertainty sources. For every performance metric, activity complexity has the higher impact compared to other uncertainty sources. Activity data doesn't have much impact on accuracy, which is the contrary to our expectations. One of the possible reasons is that the sensors used in this experiment produce data more than 20 times per minute. Therefore, even though some parts of the sensor data are missing, there are still enough sensor data that can be utilized to recognize an activity. Nevertheless, other than precision, activity data certainty presents a higher impact than the activity model.

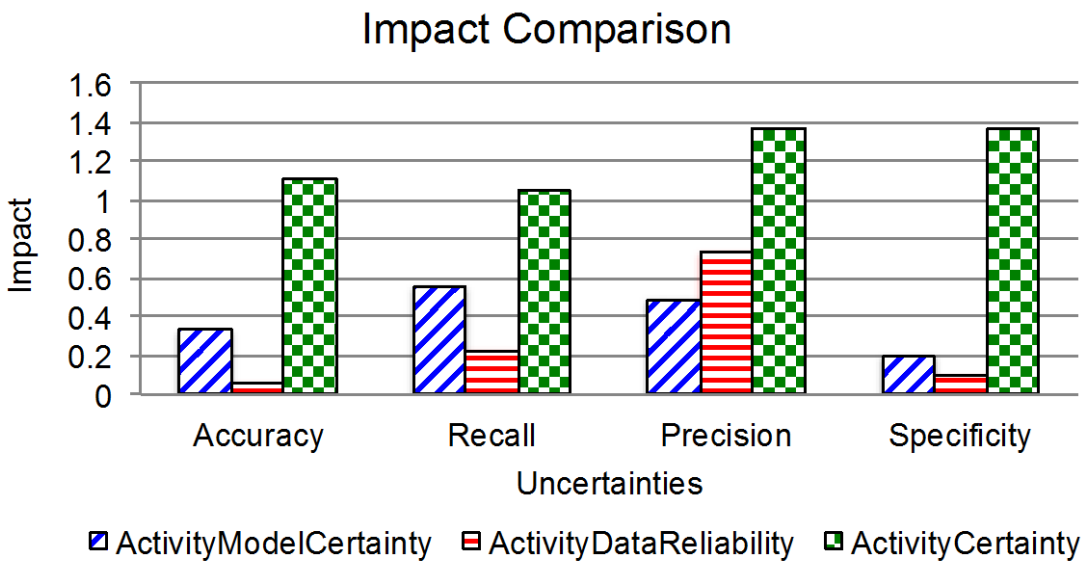


Figure 7-6. Impact comparison of uncertainties.

A more detailed analysis and comparison of uncertainty impact is presented in Figure 7-7. It presents the impact on AR performance according to the changes of the activity models, activity datasets, and activity scenarios. When the activity scenario is

changed from scenario 2 to scenario 3, the impact on activity performance is the most powerful. The next highest impact occurs when the activity model is changed from TAM to GAM. An activity model change from GAM to SGAM does not have a large impact on AR performance. Even though there may be several reasons, one of them is related to the designed activity semantics are not evidentially strong enough; therefore, it may not be easy to obtain significant performance improvement by incorporating the semantics into the model. Nevertheless, for all scenarios, the SGAM model shows a higher performance than other activity models as shown in Section 7.4.2.

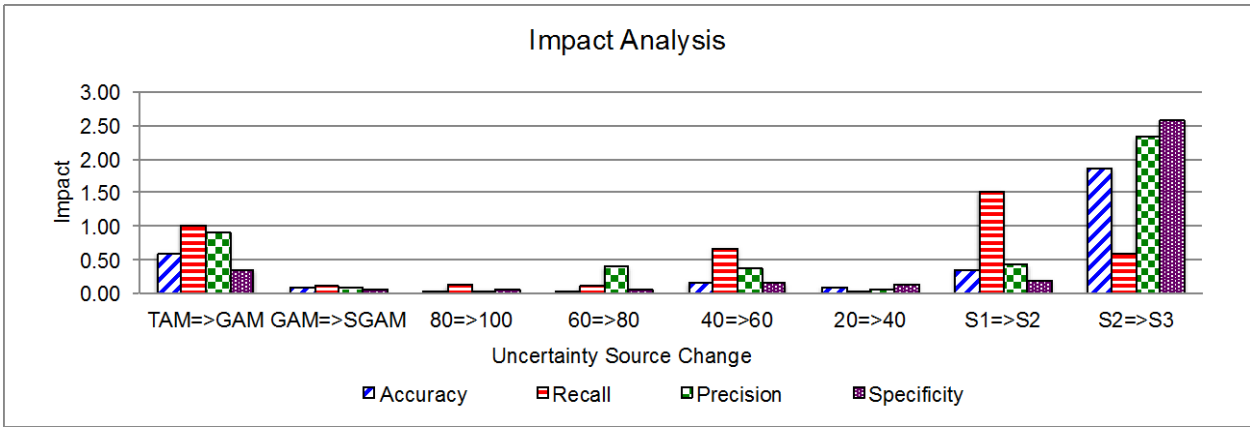


Figure 7-7. Impact of uncertainty source changes on activity model, activity data, or activity scenario.

7.5 Summary and Discussion

Insufficient accuracy is a major issue hindering the application of AR technologies in real world applications involving extensive or continuous user-smart space interactions. To solve this problem, minimizing uncertainties existing in an activity system is important. To manage uncertainty, it should be appropriately analyzed and measured. Therefore, uncertainty metrics and their measures are identified in this research. Also, a method for uncertainty impact analysis is introduced. In addition to the uncertainty analysis methods, this research provides an experimental case study for the

developed uncertainty analysis method. An uncertainty analysis example is shown along with experimental validation. In our experiments it was found that activity semantics and activity data certainty did not influence AR performance significantly, which is counter intuitive. It was found, however, that activity complexity affects AR performance the most. The activity model shows considerably high impact on AR performance, which implies that it is important to improve the activity model for improving AR performance. Even though activity complexity showed the highest impact on AR performance, it is difficult to control a user's activity complexity in a real situation.

CHAPTER 8
ACTIVITY RECOGNITION PROGRAMMING MODEL

Programmability is important for developing practical activity recognition (AR) technology based applications. Programmability in activity recognition is the capability to support activity recognition system design change according to new application requirements or activity recognition environment changes. Even though activity recognition is important to enhance and customize human-centric services, real world activity recognition system is still in its infancy. Current technology lacks the sophistication and abstraction required by programmers and developers. And, activity recognition system should be able to adapt to evolving technology.

Table 8-1. Disparity of AR programmability between AR researchers and application developers

	AR Researchers	Application programmers
AR Knowledge	Usually experts	Maybe limited
Purpose	Experiment and validation their AR approaches - Measure true and false rates	Developing applications that utilize activity recognition - Tracking activity life cycle - Verifying performed activities
Time	When AR technique is changed - Sensors - Activity modeling - AR algorithm	When application requirement changes - Target activities
Concern	Cost effective AR system setup	- APIs they can use with trust

However, technological upgrade such as sensor replacement should be cost effective. Therefore, activity recognition programming model needs to be developed. Based upon the stakeholder groups of activity recognition, there are two types of programmability. One group consists of AR researchers. The other one is application developer group. AR researchers may need to find the best possible combination by conducting a variety of experiments. Application programmers may wish to add more activities to the target domain after initial instrumentation of an activity recognition

system. The detailed disparities of AR programmability between two groups are given in Table 8-1.

Next section insists the goal of this Chapter. Section 8.2 describes activity recognition programmability in three layers. Section 8.3 shows case studies and effect of enhanced programmability through experiments. Validation will be presented in Section 8.4.

8.1 Goal: Increasing Activity Recognition Programmability

The goal of this chapter is developing an activity recognition programming model that can increase the programmability of an activity recognition system for both researchers and application programmers. To achieve this goal, a layering approach is applied because activity recognition is composed of several sub systems such as observation systems including sensors, activity models, and recognition algorithms. The layering approach can separate concerns and decouple complexities related to observation system, models and algorithms.

8.2 Activity Recognition Programmability

In our approach, an activity recognition system has a hierarchical structure, which is composed of three layers—an activity observation sub system layer; an activity model and algorithm layer; and an application layer, which utilizes recognized activities. These hierarchical layers are encapsulated into each other for maximum modularity and abstraction, which achieves independent programming of each layer. Thus, programmers in each layer can focus only on designing and implementing their layer sub-system. However, to achieve this encapsulation, it is necessary to provide interfaces between the layers. We introduce knowledge-assisted programmability,

which utilizes knowledgebase as linking interfaces between these layers. In the following we describe our approach in more details.

8.2.1 Programmability in Activity Observation Subsystem Layer

An activity observation system may frequently need addition or replacement of sensors during system upgrade or modification (e.g., due to availability of a better or more accurate sensor). Observation system programmers therefore need to be scalable to incorporate such new sensors. Also, sensors have a finite lifetime. Hence, programmers of activity observation systems should be able to replace worn out or damaged sensors. Such sensor changes are very costly in most activity recognition systems because the sensing sub-system is incorporated into and an integral part of the activity model itself. To the contrary, in our approach [4][10], the observation sub-system is decoupled from the model and changes made to it propagate systematically and automatically update the model. This represents significant savings in efforts and reduces human errors tremendously. We utilize a sensor knowledgebase, which contains sensor information critical to the activity recognition system. Whenever a new sensor is added or an existing one is replaced, sensor knowledge is updated by the programmer of this layer. Programming by updating knowledgebase is a clear and a well-defined role.

8.2.2 Programmability in Activity Model and Algorithm Layer

Activity recognition system classifies activities from sensor data according to an activity model and an activity recognition algorithm. Arriving at a perfect model and recognizer is not an easy task. Programmers and developers should be supported with a mechanism that allows them to easily change the model and recognize repeatedly over time as their conceptual knowledge about the activities evolves over time. Existing

approaches is inadequate for power programming and they do not allow for isolated changes to be made to the model or the recognizers. The slightest change usually requires careful examination of the entire model and many laborious changes and updates. In our approach, we utilize activity knowledge to support localized updates and to encourage developers to improve and refine their models and algorithms without the painful penalty of full remodeling.

To achieve this programmability, we classify commonsense knowledge to general knowledge of every activity in a given assistive environment and specific semantic knowledge of each activity. Even though both are common sense knowledge, general activity knowledge is relevant to most activities whereas semantic activity knowledge is related to only specific activities. To illustrate, activities are composed of several sub-activities called actions. Thus, the knowledge about activity-action relationship is general knowledge. On the other hand, the knowledge, “sleeping is an exclusive activity because people cannot do other activities while sleeping”, is a semantic knowledge because it characterizes the “sleeping” activity specifically. Using general knowledge, we developed a generic activity framework [9], on which basis we introduced several types of activity semantics such as dominance semantics, mutuality semantics, and order semantics [11]. Using the generic activity framework and semantic modeling techniques, programmers of an activity recognition system can design and implement human activities. If an activity model or an activity recognition algorithm is implemented or changed, the activity knowledgebase is updated accordingly. The recognition algorithm is also dynamically adjusted to the new model. Therefore, activity

model or algorithm changes do not require modifying the whole activity recognition system.

8.2.3 Programmability by Third Party in Application Layer

Recognized activities are used as input information for an application. In our approach, we provide programmers with two interfaces: “ActivityRecognition” and “ActivityVerification” [12]. The former provides a simple API interface to recognizing a specific activity. The latter interface provides much richer semantics that are needed by programmers in practice. Using “ActivityVerification” the programmer is able to discern between activities that have definitely been performed, or those performed, performing, not completed, or definitely have not been performed. In either interfaces, application programmers do not need to know detailed implementation of the other layers.

8.3 Case Study of Programmability

When sensor environment of an Activity Recognition (AR) system changes because of insertion or deletion of sensors, an AR system needs to change a neural network with the new sensor. Also it needs to retrain the neural network with new training data. These tasks represent a huge burden for many machine learning based algorithms because training and re-training require a lot of human effort. Our AR system supports sensor changes by avoiding human intervention and re-training, using as an alternative approach, a knowledge database. In the following, we describe activity knowledge database first, and then show how the knowledge can be utilized by the MLNNK algorithm during sensor changes and neural network training.

Activity Knowledge Schema. An activity knowledge database defines and stores activity entities such as action, activity or meta activity and precise relationships between the various activity layers. This knowledge is used to maintain a multi- layer

neural network. It is also used to generate training data for the neural network. The followings are parts of the activity KB schema.

- Sensor(sensor_id, sensor_type, type_name, value_type, min_value, max_value, installed_place)
- Tool(name, feature_function)
- Motion(name, type, feature_function)
- Object(name, feature_function)
- Action(action_name, motion_name, tool_name)
- Activity(activity_name, action_name, object_name)
- MetaActivity(meta_activity_name, activity_name, context).

Support of sensor insertion and Deletion. When a system detects a newly added sensor, the AR system sends queries to the knowledge database to obtain the actions, activities, and meta activities corresponding the new sensor, to build a new sub-neural network. Then, the new and existing neural networks are merged together. Therefore, sensor insertion affects only an isolated part of the neural network. If the new sensor cannot be found in the knowledge database, the user is prompted to edit the knowledge database and add the required information about the sensor, including the relationships between the sensor and activity components such as motion, tool, object, and action. When a sensor is deleted, it doesn't have any effect on the neural network. The action related to the sensor is simply inactivated. Only when the activity model is changed, the neural network is affected.

Activity Verification API Design. We designed two interfaces that can have several methods for programmers as shown in Figure 9. The interface methods have date and time as parameters. "recognizeActivity" will return the activity list performed

between the time periods being considered. “verifyActivity“ will verify if the activity is really performed between the specific time periods [12].

```
interface ActivityRecognition {
//recognize activities in start time ~ end time
//s: start date e: end date  format: mm/dd/yy,hh:mm:ss
Vector<Activity> recognizeActivity (Date s, Date e);

//recognize an activity that is currently performing
Vector<Activity> recognizeActivity ();
   …….
}

interface ActivityVerification {
//verify an activity. Return value is integer as follows
//1: definitely performed, 2: performed, 3: performing
//4: not completed, 5: definitely not performed
int verifyActivity(Date s, Date e, String activityName);
int verifyActivity(String activityName);
   …….
}
```

Figure 8-1. APIs for activity recognition and activity verification

8.4 Validation of Activity Recognition Programming Model

To examine the programmability of our approach in comparison with others, we accounted for and compared the human effort required in updating the model in response to sensor changes. As mentioned earlier in the chapter, this programmability is very important to researchers who wish to experiment with various combinations of sensors to discover the sensor set that boosts the AR performance. We considered only sensor insertion in this study. Sensor deletion and sensor attribute changes could be studied similarly.

To measure human effort of our MLNNK approach, we counted the number of changes we had to make to the knowledge database and MLNNK [10]. We also estimated the human effort required in updating the CRF model for the same set of

sensor changes we applied to our approach. Model changes in MLNNK means the number new relationships that must be added by the researcher to the knowledgebase (through a GUI interface), for each sensor insertion. We inserted 30 new sensors one by one, in a random order. We then counted the number of changes that needed to be made for both approaches. We used three sets of 30 sensors (each), repeated this experiment three times and calculated an average. For the MLNNK approach actual changes were made and counted. For the CRF approach, a graph was manually maintained and manipulated and every change to the graph was counted.

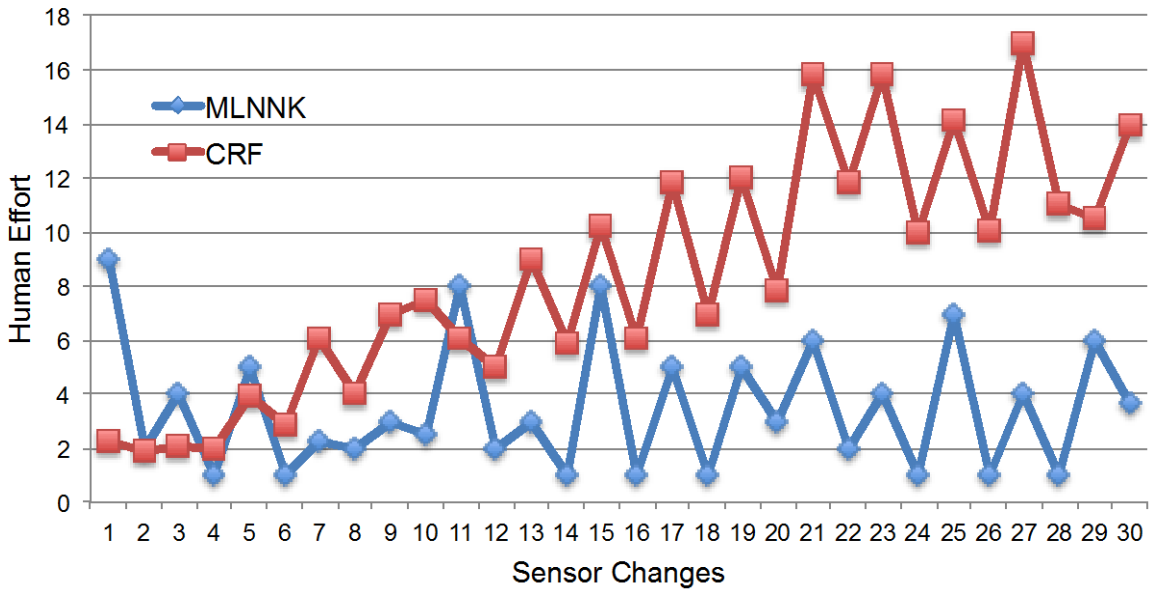


Figure 8-2. Human effort and sensor changes

Figure 8-2 and Figure 8-3 show the result of this experiment [10]. Inserting all 30 sensors costs 105 changes under MLNNK, and 250 changes under CRF, an almost 58% advantage for MLNNK (Figure 8-3). Also it was observed that under MLNNK, initial human effort was higher than CRF because of the initial updating cost of activity knowledgebase. However, human effort in CRF increased with the insertion of more

sensors, and it overwhelmed the effort in MLNNK (Figure 8-2). This is because MLNNK utilizes the knowledge such that some of the inserted sensors do not have any relationships with some of the activities. For example, insertion of a sensor installed in a cup is not related to sleeping activity. Since MLNNK has this knowledge (of no relationship), it could avoid unnecessary changes and hence reduce the number of changes per single sensor insertion. However, without knowledgebase, CRF had, by definition, to consider all possible relationships between the new sensor and existing activities.

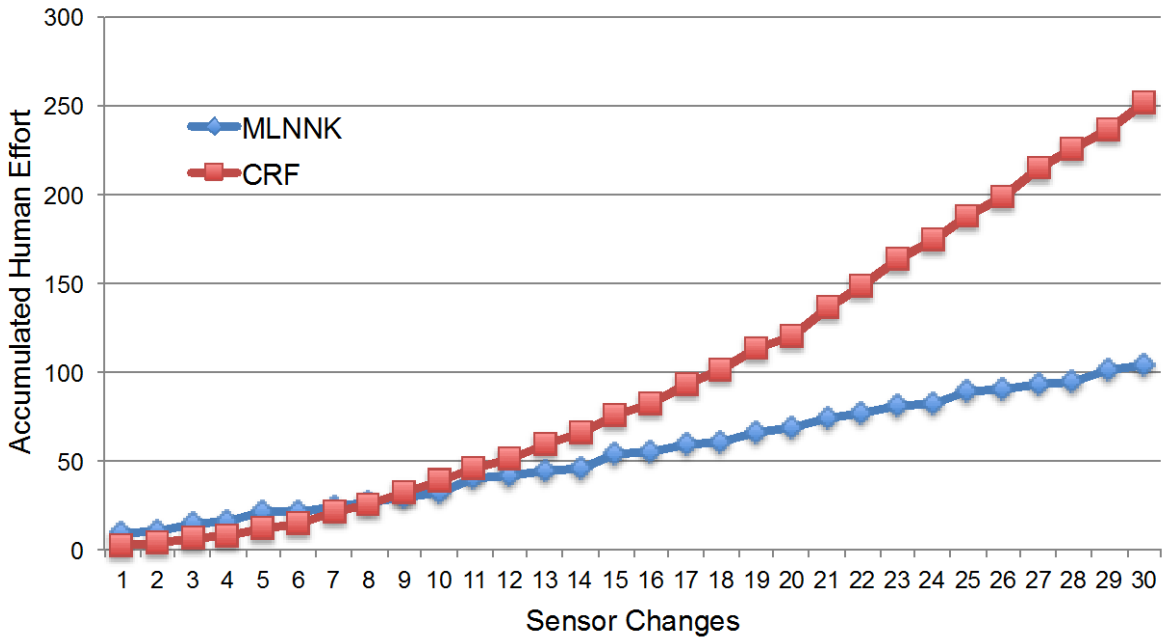


Figure 8-3. Accumulated human effort and sensor changes

8.5 Summary and Discussion

Activity recognition system requires continuous maintenance and management. Therefore, to be used in a practical application, it should be programmable. In this chapter, an approach to improve programmability is developed. The developed approach utilizes hierarchical structure to isolate the effect of changes among layers.

Knowledge is utilized to provide interface between layers. Experimental results clearly demonstrate that the increased programmability can save significant human effort.

CHAPTER 9
CONCLUSION AND FUTURE WORK

This dissertation deals with many techniques related to activity recognition (AR) such as the generic activity framework (GAF), activity semantic knowledge, activity model based on the framework and activity semantics, multi layer neural network based activity recognition algorithm, fuzzy logic based activity recognition algorithm, uncertainty sources and metrics in activity recognition systems, and activity recognition programming model.

The goal of this research is improving activity recognition accuracy and programmability. To achieve the goal, this research tried to solve problems from bottom-to up because fundamental problems in the lower layer like activity model propagate to the upper layers of an activity recognition system such as activity recognition algorithm or activity observation system.

Experiments show that the developed methods show improved accuracy and programmability. To increase activity model accuracy, generic activity framework based activity modeling technique is developed. It increases activity model accuracy by 11% compared to the activity model based on activity theory. Later, activity semantic knowledge is combined with generic activity framework for enhancing accuracy. This shows 18% increase in activity model accuracy compared to the activity model based on generic activity framework only.

Multi layer neural network (MLNNK) based AR algorithm shows better accuracy than both hidden Markov model (HMM) and conditional random field (CRF) based algorithms. The accuracy of MLNNK based AR algorithm is over 23% higher than CRF and about 14.6% higher than HMM.

Fuzzy Logic shows improved recognition accuracy compared to MLNNK. Fuzzy logic based AR algorithm shows 11% improvement in time slice activity recognition accuracy. When AR algorithms utilize activity semantic knowledge, both MLNNK and fuzzy logic algorithm show at least 27% improvement in activity recognition accuracy.

Even though activity model and activity recognition algorithm improve activity recognition accuracy significantly, many uncertainties in AR system remain unaccounted for. To measure and manage the uncertainties effectively, 6 uncertainty metrics are developed.

To improve programmability, a layering approach is developed that can benefit for both AR researchers and AR application programmers. Also, fuzzy logic based training free AR algorithm is developed.

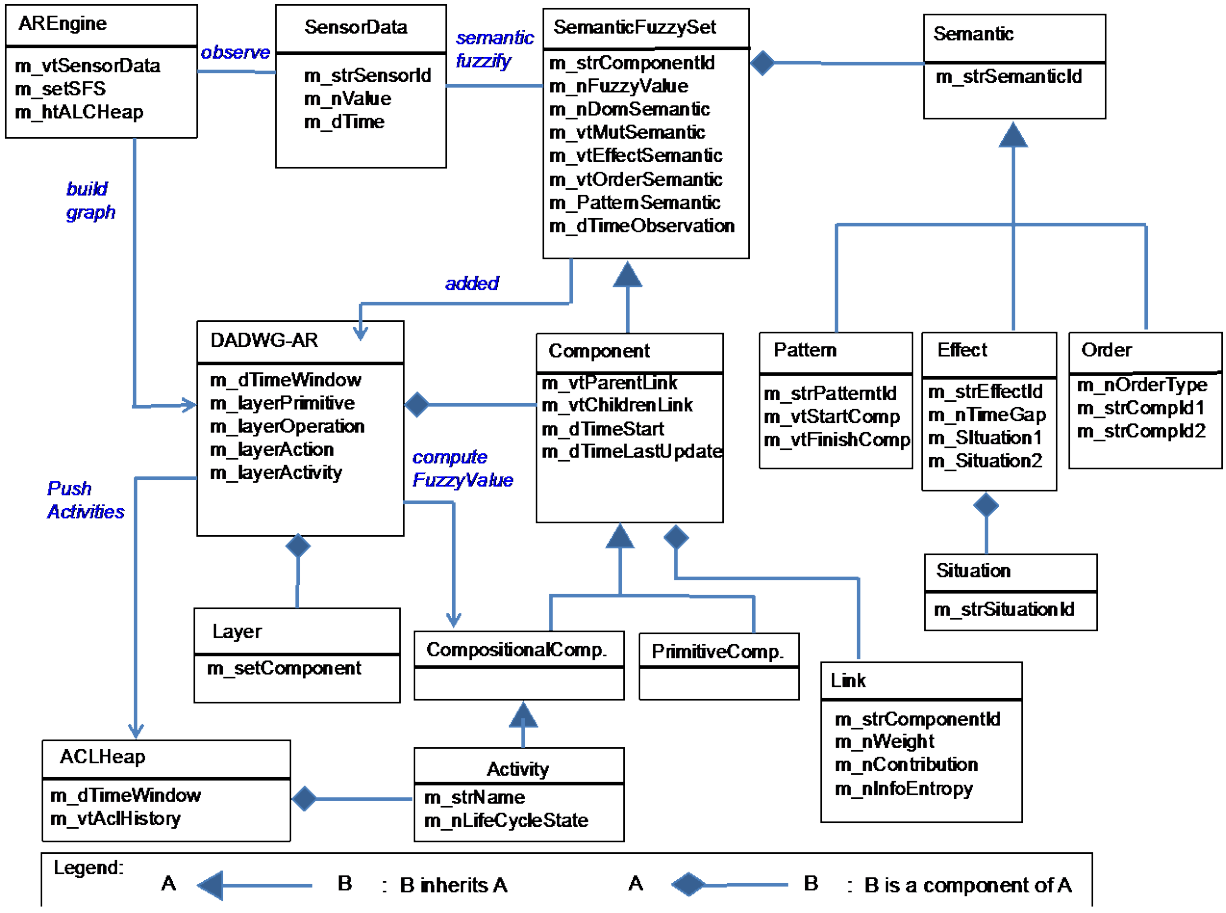
First, activity recognition research using large variety types of sensors should be performed. In this research, many sensors including wearable RFID reader, motion sensors are used. However, the wearable sensor may not be the best choice for long-term activity recognition because people may not be comfortable wearing a sensor at all times. Cutting edge sensors such as nano sensors or micro-electro-mechanical audio sensors may offer potential solutions to this problem.

Several activity models and activity recognition algorithms have been developed as part of this research. Also, several activity data sets are collected. These results are the groundwork for further experimentation. The analysis of uncertainty sources and their impact require a lot of experiments under a variety of conditions. Therefore, this research may be extended to more reliable uncertainty analysis in real world situations.

An activity recognition programming model for mobile platform should be developed in future. Improved mobility of AR systems can expand activity recognition domain from controlled indoor smart spaces to outdoor activities.

APPENDIX
IMPLEMENTATED RESULTS

The classes of fuzzy Logic and activity semantic knowledge based activity recognition system are designed as follows.



Implemented source code of key functions such as semantic fuzzification or weight computation are enumerated below:

```
//Function: for each sensor data (1) look up the primitive components relevant to the input sensor data
//(2)convert the sensor value to fuzzy membership value (3)look up semantics of the primitive components
//Input : sensor data, vtSensorData={(sensorId0,v0), (sensorId1,v1), (sensorId2,v2), …….}
//Output: set of relevent primitive components (vtPC)

public void semanticFuzzify()
{
    Fuzzifier fuzzifier = new Fuzzifier();
    for(int i=0; i<this.m_vtWindowSensorData.size(); i++)
    {
        Vector<Component> vtPC = new Vector<Component>();
        SensorData sensorData = m_vtWindowSensorData.elementAt(i);
        double nFuzzyValue = fuzzifier.fuzzifySensor(sensorData.getM_strId(), sensorData.getM_nValue());
        sensorData.setM_nValue(nFuzzyValue);
        vtPC = SGAF.lookupPC(sensorData);
        for(int j=0; j<vtPC.size(); j++)
        {
            Component component = vtPC.elementAt(j);
            Component comp = m_setSFS.get(component.getM_strComponentId());
            if( comp == null)
            {
                if(vtPC.size() > 1)
                component.setM_nLifeCycleState(Constant.ALC_TENTATIVE);
                m_setSFS.put(component.getM_strComponentId(), component);
            }
            else
            {
                if (comp.getM_nFuzzyValue() > component.getM_nFuzzyValue())
                    component.setM_nFuzzyValue(comp.getM_nFuzzyValue());
                if (comp.getM_acStartDateTime().compareTo(component.getM_acStartDateTime()) < 0)
                    component.setM_acStartDateTime(comp.getM_acStartDateTime());
                if(comp.getM_acEndDateTime().compareTo(component.getM_acEndDateTime()) > 0)
                    component.setM_acEndDateTime(comp.getM_acEndDateTime());
                if(comp.getM_acLastUpdateTime().compareTo(component.getM_acLastUpdateTime()) > 0)
                    component.setM_acEndDateTime(comp.getM_acLastUpdateTime());
                if (comp.getM_acRealStartDateTime().compareTo(component.getM_acRealStartDateTime()) <
                    0)
                    component.setM_acRealStartDateTime(comp.getM_acRealStartDateTime());
                if(comp.getM_acRealEndDateTime().compareTo(component.getM_acRealEndDateTime()) > 0)
                    component.setM_acRealEndDateTime(comp.getM_acRealEndDateTime());
                if(vtPC.size() > 1)
                    component.setM_nLifeCycleState(Constant.ALC_TENTATIVE);
                m_setSFS.put(component.getM_strComponentId(), component);
```

```
//Function: build an activity model graph using primitive component set. The graph is a subset of the entire activity
//model.
//Input: graph in previous time window and primitive components
//Output: relevant graph

public void buildGraph()
{
	buildLayer(Constant.L_OPERATION, m_layerPrimitive.getM_setComponent(), null);
	buildLayer(Constant.L_ACTION, m_layerOperation.getM_setComponent(),
	m_layerPrimitive.getM_setComponent());
	buildLayer(Constant.L_ACTIVITY, m_layerAction.getM_setComponent(), null);
}

public void buildLayer(int nTargetLayer, Hashtable<String,Component> setChildrenComponent1,
Hashtable<String,Component> setChildrenComponent2)
{
	Component child = null;
	Enumeration<Component> iterator = setChildrenComponent1.elements();
	while(iterator.hasMoreElements())
	{
		child = iterator.nextElement();
		Vector<Component> vtParent = new Vector<Component>();
		if (nTargetLayer == Constant.L_OPERATION)
		{
			vtParent = m_layerOperation.lookupPotentialOperationComponents(child);
			if(vtParent != null)
			{
				for(int i=0; i< vtParent.size(); i++)
				{
					Component parentPotential = vtParent.elementAt(i);
					Component parent =
					m_layerOperation.getComponent(parentPotential.getM_strComponentId());
					if(parent == null)
						parent = parentPotential;
					parent.updateTime(child.getM_acStartDateTime(), child.getM_acLastUpdateTime(),
					child.getM_acEndDateTime(), child.getM_acRealStartDateTime(),
					child.getM_acRealEndDateTime());
					if(m_layerOperation.mark(parent.getM_strComponentId(), child.getM_strComponentId()) ==
					false)
					{
						parent.makeChildLink(child.getM_nComponentType(), child.getM_strComponentId());
						m_layerOperation.addComponent(parent);
						child.makeParentLink(parent.getM_nComponentType(),
						parent.getM_strComponentId());
						m_layerPrimitive.updateComponent(child);
						//System.out.println("build operation: "+ parent.getM_strComponentId()+"
					"+child.getM_strComponentId());
					}
				}
			}
		}
		else if (nTargetLayer == Constant.L_ACTION)
		{
				//…..  similar to OPERATION layer
		}
		else if (nTargetLayer == Constant.L_ACTIVITY)
		{
				//…..  similar to OPERATION layer
		}
	}
}
```

```
//Function: compute weight of each link in a graph
//Input: a graph, which has link without weight value
//Output: relevant graph which has weight value for every link

public void computeWeght()
{
	computeWeight(Constant.L_OPERATION, m_layerOperation, m_layerPrimitive, null);
	computeWeight(Constant.L_ACTION, m_layerAction, m_layerOperation, m_layerPrimitive);
	computeWeight(Constant.L_ACTIVITY, m_layerActivity, m_layerAction, null);
}

public void computeWeight(int nTargetlayer, Layer parentsLayer, Layer childrenLayer1, Layer childrenLayer2)
{
	Vector<String> vtChildrenKey1 = new Vector<String>();
	Vector<String> vtChildrenKey2 = new Vector<String>();
	Vector<String> vtChildrenKey = new Vector<String>();

	int nChildrenKeySize1 = 0;
	int nChildrenKeySize2 = 0;
	int nChildrenKeySize = 0;
	Hashtable<String, Integer> htChildrenLayer = new Hashtable<String, Integer>(); //id, which layer

	if(childrenLayer1 != null)
	{
		vtChildrenKey1 = childrenLayer1.getKeys();
		nChildrenKeySize1 = vtChildrenKey1.size();
		for(int i=0; i<vtChildrenKey1.size(); i++)
		{
		String strChildrenId = vtChildrenKey1.elementAt(i);
		vtChildrenKey.add(strChildrenId);
		htChildrenLayer.put(strChildrenId, new Integer(childrenLayer1.getM_nLayerId()));
		}
	}

	if(childrenLayer2 != null)
	{
		vtChildrenKey2 = childrenLayer2.getKeys();
		nChildrenKeySize2 = vtChildrenKey2.size();
		for(int i=0; i<vtChildrenKey2.size(); i++)
		{
			String strChildrenId = vtChildrenKey2.elementAt(i);
			vtChildrenKey.add(strChildrenId);
			htChildrenLayer.put(strChildrenId, new Integer(childrenLayer2.getM_nLayerId()));
		}
	}

	nChildrenKeySize = nChildrenKeySize1 + nChildrenKeySize2;
	int m = nChildrenKeySize1+nChildrenKeySize2;
	while(m > 0 && vtChildrenKey.size() > 0)
	{
		int nMaxCases = (int)Math.pow(2, nChildrenKeySize);
		int n= nMaxCases - 1;
		while (n > 0 && vtChildrenKey.size() > 0)
		{
			//System.out.println("n= "+n);
			String strBinary = Util.numberToBinaryString(n);
			int nCountOne = Util.countOne(n);
			if(m == nCountOne)
			{
```

```
Vector<String> subSetKey1 = null;
Vector<String> subSetKey2 = null;
Vector<String> subSetKey = new Vector<String>();
HashSet<String> setParent1 = null;
HashSet<String> setParent2 = null;
HashSet<String> setParent = null;
boolean bIsTentativeAllChildren = false;
boolean bIsTentativeAllChildren1 = false;
boolean bIsTentativeAllChildren2 = false;
subSetKey1 = composeSubsetKey(vtChildrenKey, strBinary, 0, nChildrenKeySize1-1);
subSetKey2 = composeSubsetKey(vtChildrenKey, strBinary, nChildrenKeySize1,
                nChildrenKeySize-1);
if(subSetKey1 != null)
{
    //find parent that is all subSetKey1's parent
    setParent1 = childrenLayer1.lookupParent(nTargetlayer, subSetKey1);
    subSetKey.addAll(subSetKey1);
    bIsTentativeAllChildren1 = childrenLayer1.isTentativeAllChildren(subSetKey1);
}
if(childrenLayer2 != null && subSetKey2 != null)
{
    setParent2 = childrenLayer2.lookupParent(nTargetlayer, subSetKey2);
    subSetKey.addAll(subSetKey2);
    bIsTentativeAllChildren2 = childrenLayer2.isTentativeAllChildren(subSetKey2);
}
//intersection
if(subSetKey1 != null && subSetKey2 != null)
{
    if(subSetKey1.size() > 0 && subSetKey2.size() > 0)
        setParent = Util.intersect(setParent1, setParent2);
    else if(subSetKey1.size() > 0 && subSetKey2.size() == 0 && setParent1 != null)
        setParent = setParent1;
    else if(subSetKey1.size() == 0 && subSetKey2.size() > 0 && setParent2 != null)
        setParent = setParent2;

    if(bIsTentativeAllChildren1 == true && bIsTentativeAllChildren2 == true)
        bIsTentativeAllChildren = true;
    else
        bIsTentativeAllChildren = false;
}
else if(subSetKey1 != null && subSetKey2 == null)
{
    if(subSetKey1.size() > 0 && setParent1 != null)
    setParent = setParent1;
    bIsTentativeAllChildren = bIsTentativeAllChildren1;
}
else if(subSetKey1 == null && subSetKey2 != null && setParent2 != null)
{
    if(subSetKey2.size() > 0)
    setParent = setParent2;
    bIsTentativeAllChildren = bIsTentativeAllChildren2;
}
if(setParent.size() > 0)
{
    int nLifeCycleState = Constant.ALC_UNDEFINED;
    if(setParent.size() > 1)
        nLifeCycleState = Constant.ALC_TENTATIVE;
    else if(bIsTentativeAllChildren == true)
        nLifeCycleState = Constant.ALC_TENTATIVE;
```

```
                    Iterator<String> iterator = setParent.iterator();
                    while(iterator.hasNext())
                    {
                        Component parent = parentsLayer.getComponent(iterator.next());//parent
                        if(parent != null)
                        {
                            parent.setM_nLifeCycleState(nLifeCycleState);
                            if(subSetKey1 != null)
                            {
                                for(int i=0; i< subSetKey1.size(); i++)
                                {
                                    String strChildId = subSetKey1.elementAt(i);
                                    Component child = childrenLayer1.getComponent(strChildId);
                                    double nInfoEntropy = 1.0/(double)setParent.size();
                                    double nContribution =
                                    1.0/(double)SGAF.getChildrenSize(nTargetlayer,
                                    parent.getM_strComponentId());
                                    updateLink(nTargetlayer, parent.getM_nComponentType(),
                                    parent.getM_strComponentId(), child.getM_nComponentType(),
                                    child.getM_strComponentId(), nInfoEntropy, nContribution);
                                }
                            }
                            if(subSetKey2 != null)
                            {
                                for(int i=0; i< subSetKey2.size(); i++)
                                {
                                    String strChildId = subSetKey2.elementAt(i);
                                    Component child = childrenLayer2.getComponent(strChildId);
                                    double nInfoEntropy = 1.0/(double)setParent.size();
                                    double nContribution = 1.0/(double)SGAF.getChildrenSize(nTargetlayer,
                                    parent.getM_strComponentId());
                                    updateLink(nTargetlayer, parent.getM_nComponentType(),
                                    parent.getM_strComponentId(), child.getM_nComponentType(),
                                    child.getM_strComponentId(), nInfoEntropy, nContribution);
                                }
                            }
                        }
                    }
                    if(subSetKey1 != null)
                    {
                        vtChildrenKey1.removeAll(subSetKey1);
                        vtChildrenKey.removeAll(subSetKey1);
                        nChildrenKeySize1 = vtChildrenKey1.size();
                    }
                    if(subSetKey2 != null)
                    {
                        vtChildrenKey2.removeAll(subSetKey2);
                        vtChildrenKey.removeAll(subSetKey2);
                        nChildrenKeySize2 = vtChildrenKey2.size();
                    }
                    nChildrenKeySize = nChildrenKeySize1 + nChildrenKeySize2;
                    nMaxCases = (int)Math.pow(2, vtChildrenKey.size());
                    n=nMaxCases;
                }
            }
            n--;
        }
        m--;
    }//m
}
```

```
//Function: perform fuzzy dominance operation to compute the fuzzy value of a compositional component
//Input: compositional components (CC) which is a target compositional component),
//graph: graph with activity components that are related to CC
//Output: the fuzzy value of CC,  outputComp.m_nFuzzyValue

public void performFuzzyOperation()
{
	for (int nLayer = Constant.L_OPERATION; nLayer <= Constant.L_ACTIVITY; nLayer++)
	{
		performFuzzyDominanceOperation(nLayer);
		if(Config.ACTIVITY_MODEL == Constant.SGAM)
		{
			enforceOrderSemantics(nLayer);
			enforceMutualitySemantics(nLayer);
			enforceEffectSemantics(nLayer);
		}
	}
}

//Function: it keeps all history of on-going activity in heap
//Input: current time window
//Output: all finished activities including both completed and not completed activities
public void evaluateLifeCycle(long nTimeWindow) //nTimeWindow: current time window
{
	Vector<Activity> vtRemovedActivity = new Vector<Activity>();
	Vector<Activity> vtRemovedTentativeActivity = new Vector<Activity>();
	Enumeration<ALCHeap> enumAlcHeap = m_htAlcHeap.elements(); //one heap per each activity
	while(enumAlcHeap.hasMoreElements())
	{
		ALCHeap alcHeap = enumAlcHeap.nextElement();
		Vector<Activity> vtActivityHistory = alcHeap.getM_vtAlcHistory();
		Activity lastActivity = vtActivityHistory.lastElement();
		Activity previousActivity = null;
		double nCurrentFuzzyValue = 0.0; double nPreviousFuzzyValue = 0.0;
		double nFuzzyValueChange = 0.0;long nRestartedPauseDuration = 0;
		long nPauseDuration = 0;
		if(vtActivityHistory != null)  //recognition history of an activity
		{
			if(vtActivityHistory.size() == 0) m_htAlcHeap.remove(lastActivity.getM_strActivityId());
			else if(vtActivityHistory.size() == 1)
			{
				//…
			}
			else if(vtActivityHistory.size() >= 2)
			{
				previousActivity = vtActivityHistory.elementAt(vtActivityHistory.size()-2);
				nPreviousFuzzyValue = previousActivity.getM_nFuzzyValue();
				nRestartedPauseDuration = lastActivity.getM_acStartDateTime().getM_nTimeWindow() -
				previousActivity.getM_acStartDateTime().getM_nTimeWindow();
				nPauseDuration = nTimeWindow - lastActivity.getM_acEndDateTime().getM_nTimeWindow();
				if(nPauseDuration == 0) //currently recognized
				{
					//
				}
				else if(nPauseDuration == 1)
				{
					//
				}
				else if(nPauseDuration > 1)
				{
					nCurrentFuzzyValue = 0.0; nPreviousFuzzyValue = 0.0; nFuzzyValueChange = 0.0;
				}
```

```
                    if(lastActivity.getM_nLifeCycleState() != Constant.ALC_TENTATIVE) //undefined
                    {
                    switch(change( nCurrentFuzzyValue,  nPreviousFuzzyValue,  nFuzzyValueChange,
                    nRestartedPauseDuration, nPauseDuration, lastActivity.getM_nLifeCycleState()))
                    {
                        case Constant.CASE_RESTARTED:
                        {
                            int nStart = (int)previousActivity.getM_acStartDateTime().getM_nTimeWindow();
                            int nEnd = (int)lastActivity.getM_acStartDateTime().getM_nTimeWindow();
                            vtActivityHistory.remove(vtActivityHistory.size()-1);
                            for(int i=(nStart+1); i< nEnd; i++)
                            {
                                if(i<(vtActivityHistory.size()-1))
                                {
                                    Activity activity = vtActivityHistory.elementAt(i-1);
                                    Activity dummyActivity = this.makeDummyActivity(activity,
                                    Constant.ALC_PAUSED);
                                    vtActivityHistory.add(dummyActivity);
                                }
                            }
                            lastActivity.setM_nLifeCycleState(Constant.ALC_RESTARTED);
                            vtActivityHistory.add(lastActivity);
                            //vtActivityHistory.set(vtActivityHistory.size()-1, lastActivity);
                            break;
                        }
                        case Constant.CASE_PERFORMING:
                        {
                            lastActivity.setM_nLifeCycleState(Constant.ALC_PERFORMING);
                            vtActivityHistory.set(vtActivityHistory.size()-1, lastActivity);
                            …//change other tentative activities' fuzzy value to zero
                        }
                        case Constant.CASE_PAUSED: break;
                        case Constant.CASE_COMPLETED:
                        {
                            alcHeap.removePauseAtEnd();
                            Activity dummyActivity = makeLastActivity(lastActivity, Constant.ALC_COMPLETED);
                            vtActivityHistory.add(dummyActivity);
                            m_vtRecognizedActivity.add(alcHeap);
                            vtRemovedActivity.add(lastActivity); //to remove from tree
                            m_htAlcHeap.remove(lastActivity.getM_strActivityId());
                            break;
                        }
                        case Constant.CASE_INCOMPLETED:
                        {
                            alcHeap.removePauseAtEnd();
                            Activity dummyActivity = makeLastActivity(lastActivity,
                            Constant.ALC_INCOMPLETED);
                            vtActivityHistory.add(dummyActivity);
                            m_vtRecognizedActivity.add(alcHeap);
                            vtRemovedActivity.add(lastActivity); //to remove from tree
                            m_htAlcHeap.remove(lastActivity.getM_strActivityId());
                        }
                    }
                    }
                }
            } //end of activity life cycle
            m_graphART.removeTree(vtRemovedActivity);
            m_graphART.removeTree(vtRemovedTentativeActivity);
        }
    }
```

LIST OF REFERENCES

1. S. Helal, D. Cook and M. Schmalz, “Smart Home-based Health Platform for Behavioral Monitoring and Alteration of Diabetes Patients,” Journal of Diabetes Science and Technology, Volume 3, Number 1, January, 2009.

2. S. Helal, J. King, H. Zabadani and Y Kaddourah, "The Gator Tech Smart House: An Assistive Environment for Successful Aging," Book Chapter in "Advanced Intelligent Environments," H. Hagrass, Editor, Springer Verlag. 2008.

3. S. Helal, A. Mendez-Vazquez and S. Hossain, “Specification and Synthesis of Sensory Datasets in Pervasive Spaces,” In Proceedings of the IEEE Symposium on Computers and Communications (ISCC'09), Sousse, Tunisia, July 5-8, 2009.

4. S. Helal, E. Kim, S. Hossain, “Scalable Approaches to Activity Recognition Research,” in 8th International Conference Pervasive Workshop, 2010.

5. S. Helal, J.W. Lee, S. Hossain, E. Kim, H. Hagras, D. Cook, "Persim- Simulator for Human Activities in Pervasive Spaces," Proceedings of the 7th International Conference on Intelligent Environments (IE 11), Nottingham, United Kindom, 25-28 July, 2011.

6. W. Mann and S. Helal, “Smart Technology A Bright Future for Independent Living,” In The Society for Certified Senior Advisors Journal, vol. 21, pp. 15-20, 2003.

7. E. Kim, S. Helal, D. Cook, "Human Activity Recognition and Pattern Discovery," IEEE Pervasive Computing vol. 9, no. 1, pp. 48-52, 2010.

8. E. Kim, S. Helal, J. Lee, and S. Hossain, "Making an activity dataset," Proceedings of the 7th International Conference on Ubiquitous Intelligence and Computing, Xi'an, China, 2010.

9. E. Kim and S. Helal, "Revisiting Human Activity Frameworks," In Proceedings of the 2nd International ICST Conference on Wireless Sensor Network Systems and Software, Miami, USA, 2010.

10. E. Kim and S. Helal, "Practical and Robust Activity Modeling and Recognition," Proceedings of the 8th international conference on wearable micro and nano technologies for personalized health, Lyon, France, 2011.

11. E. Kim and S. Helal, "Modeling Human Activity Semantics for Improved Recognition Performance," Proceedings of the 8th International Conference on Ubiquitous Intelligence and Computing, Banff, Canada, 2011.

12. E. Kim, S. Helal, C. Nugent and J. Lee, "Assurance-Oriented Activity Recognition," Proceedings of the International Workshop on Situation, Activity and Goal Awareness," (Beijing, China). ACM, New York, NY, 2011.

13. A. Abraham, “Rule-based Expert Systems,” in Handbook of Measuring System Design, edited by Peter H Sydenham and Richard Thorn John Wiley & Sons 2005.

14. A. Crandall and D. Cook, D, “Coping with multiple residents in a smart environment,” In Journal of Ambient Intelligence and Smart Environments, pp. 323-334, 2009.

15. A. Dahlbom, L. Niklasson, G. Falkman, and A. Loutfi, “Towards template-based situation recognition,” In Intelligent Sensing, Situation Management, Impact Assessment, and Cyber-Sensing, vol. 7352. SPIE. 2009.

16. AK. Dey and GD. Abowd, “Towards a better understanding of context and context-awareness,” CHI’2000 Workshop on the What, Who, Where, When, and How of Context- Awareness, 2000.

17. A. Rozan, M. Zaidi, and M. Yoshiki, “The Presence of Beneficial Knowledge in Web Forum: Analysis by Kipling's Framework. In: Knowledge Management International Conference & Exhibition (KMICE 2006), Kuala Lumpur, Malaysia, 2006.

18. B. Logan, J, Healey, M. Philipose, E. M. Tapia, S. S. Intille, “A long-term evaluation of sensing modalities for activity recognition,” In: Proceedings of the International Conference on Ubiquitous Computing. Berlin Heidelberg: Springer-Verlag, pp. 483–500, 2007.

19. B. Nardi, “Studying context: A comparison of activity theory, situated action models, and distributed cognition. Context and consciousness: activity theory and human-computer interaction,” Cambridge, MA., The MIT Press, 1996.

20. C. A. Halverson, “Activity Theory and Distributed Cognition: Or What Does CSCW Need to DO with Theories?,” Computer Supported Cooperative Work, 11 2002.

21. C. D. Nugent, M. D. Mulvenna, X. Hong, and S. Devlin, “Experiences in the development of a Smart Lab.,” In International Journal of Biomedical Engineering and Technology, 2, 4, Apr, 2009, 319-331.

22. C. Dousson, P. Gaborit, and M. Ghallab, “Situation recognition: Representation and algorithms,” In Proc. of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93). Chambery, France, pp. 166-172, 1993.

23. C. K. Chang, H. Jiang, H. Ming and K. Oyama, “Situ: A Situation-Theoretic Approach to Context-Aware Service Evolution,” In Proc of IEEE T. Services Computing, 261-275. 2009.

24. C. Sutton and A. McCallum, “An introduction to conditional random fields for relational learning,” In L. Getoor and B. Taskar, editors, Introduction to Statistical Relational Learning. MIT Press, 2006.

25. D. G. Altman and J. M. Bland, “Statistics notes: diagnostic tests 1: sensitivity and specificity,” in Br. Med. J. 308:1552, 1994.

26. D.W. Hubbard, How to measure anything. Hoboken: John Wiley & Sons, 2007.

27. D. Kulkarni and A. Tripathi. A framework for programming robust context-aware applications. IEEE Trans. Softw. Eng., 36:184–197, 2010.

28. D. Surie, T. Pederson, F. Lagriffoul, L. E. Janlert, D. Sjolie, “Activity Recognition using an Egocentric Perspective of Everyday Objects,” in UIC (2007) Proceedings of IFIP 2007 International Conference on Ubiquitous Intelligence and Computing. LNCS, vol. 4611, pp. 246–257. Springer, Heidelberg, 2007.

29. E. M. Tapia, S. S. Intille, L. Lopez, and K. Larson, “ The design of a portable kit of wireless sensors for naturalistic data collection,” In Proceedings of PERVASIVE 2006. Berlin Heidelberg: Springer-Verlag, pp. 117-134, 2006.

30. F. Cruciani, M. Donnelly, C. D. Nugent, G. Parente, C. Paggetti and W. Burns, “DANTE: A Video Based Annotation Tool for Smart Environments,” in Sensor Systems and Software Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 57, 5, 179--188.

31. G. Chen and T. T. Pham, Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems, CRC Press 2000.

32. H. M. Wallach. “Conditional random fields: An introduction,” Technical Report MS-CIS-04-21, University of Pennsylvania CIS 2004.

33. H. Yang, J. King, A. Helal and E. Jansen, “A Context-Driven Programming Model for Pervasive Spaces,” Proceedings of the 5th International Conference on Smart Homes and Health Telematics (ICOST), Nara, Japan, 21-23 June, 2007.

34. J. D. Irwin, “The industrial electronics handbook,” pp. 829. IEEE Press, 1997.

35. J. Jantzen, “Tutorial On Fuzzy Logic,” in Tech. report no 98-E 868, Aug 1998, Technical University of Denmark, 1998.

36. J. M. Nigro and M. Rombaut. Idres: A Rule-Based Systemfor Driving Situation Recognition with Uncertainty Management, Information Fusion, Vol 4., 2003.

37. K. Gregory, G. Bibbo, and J. E. Pattison, “A Standard Approach to Measurement Uncertainties for Scientists and Engineers in Medicine,” Australasian Physical & Engineering Sciences in Medicine, 28(2), 131-139, (2005).

38. K. Kusrin, “Question Quantification to Obtain User Certainty Factor in Expert System Application for Disease Diagnosis,” In Proceedings of the International Conference on Electrical Engineering and Informatics,” pp. 765-768, 2007.

39. K. Kuutti, “Activity theory as a potential framework for human-computer interaction research,” in B.A Nardi (eds.), Context and consciousness: Activity theory and human-computer interaction.Cambridge, MA: MIT Press, 1996.

40. L. Chen, CD. Nugent, D. Cook, and Z. Yu, “Knowledge-drivem Activity Recognition in Intelligent Environments,” Special Issue of the International Journal of Pervasive and Mobile Computing, ISSN:1574-1192, Elsevier, 2011.

41. L. Constantine, “Human Activity Modeling: Toward A Pragmatic Integration of Activity Theory and Usage-Centered Design,” in Human-Centered Software Engineering, p24-50, 2009.

42. L. P. Swiler, L. T. L. Paez, and R. L. Mayes, “Epistemic Uncertainty Quantification Tutorial,” In Proceedings of the IMAC-XXVII, Orlando, Florida USA 2009.

43. L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2):257–286, 1989.

44. L. S. Ravindranath, A. Thiagarajan, H. Balakrishnan and S. Madden, “Code In The Air: Simplifying Sensing and Coordination Tasks on Smartphones,” In Proceedings of HotMobile, La Jolla, CA, USA 2012.

45. M. Kim and M. Kim, “ A Formal Definition of Situation towards Situation-Aware Computing,” In Proc WSKS (1). 553-563, 2009.

46. M. Mitchell, “Machine Learning,” in WCB/McGraw-Hill, p 88-108, 1997.

47. M. R. Endsley, “Theoretical underpinnings of situation awareness: a critical review,” In: Endsley, M.R., Garland, D.J. (Eds.), Situation Awareness Analysis and Measurement. Lawrence Erlbaum, Mahwah, NJ, 2000.

48. M. Skubic, G. Alexander, M. Popescu, M. Rantz, J. Keller, "A smarthome application to eldercare: current status and lessons learned," Technology and health care : official journal of the European Society for Engineering and Medicine, Vol. 17, No. 3., pp. 183-201, 2009.

49. M. Sokolova, N. Japkowicz, N and S. Szpakowicz, “Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation,” In AI 2006: Advances in Artificial Intelligence Lecture Notes in Computer Science, 4304, pp. 1015—1021, 2006.

50. M. Tentori, J. Favela, " Activity-Aware Computing for Healthcare," IEEE Pervasive Computing Magazine, vol. 7 no. 2, pp. 51-57, Apr-Jun, 2008.

51. M. Varshney and R. Bagrodia, “Detailed Models for Sensor Network Simulations and their Impact on Network Performance,” In Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems, ACM, pp. 70-77, October, 2004.

52. P. J. F. Lucas, “Certainty-factor-like structures in Bayesian belief networks,” Knowledge-Based Systems, 14, pp327-335, 2001.

53. P. Lefrere, ”Activity-based scenarios for and approaches to ubiquitous e-Learning,” in Personal and Ubiquitous Computing, vol. 13, n. 3, pp. 219-227, 2009.

54. P. Lingras, C. J. Butz, “Precision and Recall in Rough Support Vector Machines,” in 2007 IEEE Int. Conf. on Granular Computing, pp. 654–658, 2007.

55. R. Böhme and F. C. Freiling, “On metrics and measurements,” in I. Eusgeld, F. C. Freiling and R. Reussner (Eds.): Dependability Metrics, LNCS 4909, Springer-Verlag, pp. 7-13, 2008.

56. S. A. Alvarez, “An exact analytical relation among recall, precision, and classification accuracy in information retrieval,” in Technical Report BCCS-02-01, Computer Science Department, Boston College, 2002.

57. S. Bell, "Measurement Good Practice Guide: A Beginner's Guide to Uncertainty of Measurement," National Physical Laboratory, Teddington, Middlesex, United Kingdom, ISSN 1368-6550, Issue 2 with amendments March (2001).

58. S. McKeever, J. Ye, L. Coyle, and S. Dobson, “A multilayered uncertainty model for context aware systems,” In Adjunct proceedings of the international conference on Pervasive Computing: Late Breaking Result, May 2008, pp. 1-4 (2008).

59. S. S. Intille, K. Larson, M. E. Tapia, J. Beaudin, P. Kaushik, J. Nawyn, R. Rockinson, “ Using a live-in laboratory for ubiquitous computing research,” In Proceedings of PERVASIVE, Fishkin, K. P., Schiele, B., Nixon, P., Quigley, A. (eds) LNCS vol. 3968, pp. 349-365, Springer, Berlin Heidelberg, 2006.

60. S. S. Wang, W. Pentney, A-M. Popescu, T. Choudhury, M. Philipose, “Commonsense-based joint training of human activity recognizers,” In Proceedings of international joint conference on artificial intelligence (IJCAI), 2007.

61. S. S. Yau, D. Huang, H. Gong, and S. Seth, “Development and runtime support for situation-aware application software in ubiquitous computing environments,” In 28th Annual International Computer Software and Application Conference (COMPSAC), Hong Kong, 452–457, 2004.

62. T. Kasteren, A. Noulas, G. Englebienne, and B.Krose, “Accurate Activity Recognition in a Home Setting,” Proceedings of the Tenth International Conference on Ubiquitous Computing (Ubicomp 2008), Seoul, Korea, pp 1-9, 2008.

63. T. DeMarco, Controlling Software Projects, Yourdon Press, New York, 1986.

64. V. V. Davydov, V. P. Zinchenko, N. F. Talyzina, “The Problem of Activity. In the Works of A. N.Leontjev,” Soviet Psychology, vol. 21, n. 4, pp. 31-42, 1983.

65. W. Banks, “Linguistic Variables, Clear Thinking with Fuzzy Logic,” in Toronto. Fuzz-C, 2008.

66. W. Pentney, “Sensor-Based Understanding of Daily Life via Large-Scale Use of Common Sense,” in Proceedings of AAAI ‘06, Boston, MA, USA, Jul, 2006.

67. Y. Qiang, “ Proceedings of the 11th international conference on Ubiquitous computing, Orlando, Florida, USA, September 30-October 03, 2009.

68. Y. Y. Yao, "Semantics of fuzzy sets in rough set theory," LNCS Transactions on Rough Sets,vol.I,pp.310-331,2004.

69. Z. Vincent, H. Derek, and Y. Qiang, “Cross-domain activity recognition,” Proceedings of the 11th international conference on Ubiquitous computing, September 30-October 03, 2009, Orlando, Florida, USA, 2009.

70. National Diabetes Information Clearinghouse (NDIC)

71. http://diabetes.niddk.nih.gov/dm/pubs/hypoglycemia/

72. A service of the U.S. National Library of Medicine From the National Institutes of Health (NIH) http://www.nlm.nih.gov/medlineplus/ency/article/003400.htm

73. Ceyon, Korea, http://www.ceyon.co.kr/

74. Phidgets plug & play sensors, http://www.phidgets.com/

75. Tanita, Japan, http://www.tanita.com/en/bc590bt/

76. Myglucohealth, USA, www.myglucohealth.net

77. University of Amsterdam Activity Data set. http://www.science.uva.nl/~tlmkaste/

78. Neuroph library.http://netbeans.dzone.com/articles/neurophmdashsmart-apps/

79. Washington State University Activity dataset, http://ailab.eecs.wsu.edu/casas/datasets.html

80. Persim – A simulator for human-activities in Pervasive Spaces. Project web site: http://www.icta.ufl.edu/projects_persim

81. SDDL – Sensory Dataset Description Language. Specifications and resources web site: www.icta.ufl.edu/persim/sddl/

82. International Organization for Standardization, “Guide to the expression of uncertainty in measurement,” International Organization for Standardization, Geneva, 1995.

83. Sun SPOT sensor, http://www.sunspotworld.com/

84. Tynetec sensor, Tynetec.http://www.tynetec.co.uk/

BIOGRAPHICAL SKETCH

Eunju Kim received her MS degree from Dongguk University in South Korea in 1997 and pursued a software engineering career until she started her Ph.D. program in 2006. She is currently working on human activity recognition in smart spaces for health care and elder care applications. She is a member of the Mobile and Pervasive Computing Laboratory at the University of Florida. Her research goals are focused on developing practical activity recognition technology that offers high accuracy and programmability required by real world applications.