國立臺灣大學電機資訊學院資訊工程學研究所

碩士論文

Graduate Institute of Computer Science and Information Engineering

College of Electrical Engineering ＆ Computer Science

National Taiwan University

Master Thesis

動態環境下以主動式學習加強的自行重構之行為辨識

Active Learning Assisted Self-reconfigurable Activity Recognition in Dynamic Environment

何育誠

Yu-Chen Ho

指導教授：傅立成 博士

Advisor: Li-Chen Fu, Ph.D.

中華民國 97 年 7 月

July, 2008

誌謝

時光飛逝，兩年的研究生活轉眼間就過了，在這段時而有趣時而苦悶的過程中，首先要感謝的是指導教授傅立成老師，不管多忙碌、多累，傅老師總是會撥出時間給予我細心的指導與教誨，在研究上，也提供了我一個很自由的環境，讓我能無顧慮的在自己有興趣的領域發揮。另外，我要感謝四位口試委員，臺大資工所朱浩華教授，文化大學建築系溫琇玲教授，臺大社工系馮燕教授及資策會網多所馮明惠主任，對於我論文上提供了許多寶貴的建議，使得本篇論文更加完善。

除此之外，也要感謝實驗室的夥伴們。尤其是小陸學長及黃老學長，能夠不辭辛勞的聽我講述自己不完善的想法並引導我，兆麟學長及峻鋒學長也在智慧家庭組上給予我很多指引和鼓勵，對於我的論文研究上提供了許多的建議與方向，讓我受益匪淺。也感謝一同奮鬥的同學們，佳委、振勛、嘉銘、禹安與一航，謝謝你們陪我走過這段求學歷程。還有學弟意函，在我的論文實做上幫了我不少忙，讓我減輕了不少負擔。還有實驗室的其他成員，宗哲學長、恩偉學長、益銘學長、士桓學長、亞文、忞蔚、思穎、擂茶、macaca 與學謙，陪我在實驗室生活。還有其他系上的同學，brooky、salt、家俊、文芝、祖佑、atwood、188、啟嘉、栗子，陪我一起討論、玩樂。

最後，我要感謝我的家人，尤其是我的父母，有您們全力的支持與鼓勵，我才能無後顧之憂的完成學業。

中文摘要

本論文提出在智慧環境中學習及便是人類日常生活的問題，先前大部分的方法先收集人類行為的資料並學出其模型，然後再使用學得的模型來辨識人的行為，然而，人的行為習慣及環境的佈置可能會隨著時間而發生改變，造成行為的模式發生改變，這時舊有的辨識用的行為模型便過時了，使辨識率降低，必須要重新學習新的行為模型，但是重新收集學習用的行為資料並給予對應的行為標籤是件非常煩人且容易出錯的工作，在這樣的情況下，在更新行為模型時能降低人為的指導工作份量是件非常重要的事，本論文提出一個可以自我調整行為模型的行為辨識方法，它可以在動態的環境下同時辨識多種行為，並以較少的人為指導來跟著環境變動調整行為模型。

關鍵字: 行為辨識、機率推論、動態貝氏網路、主動式學習

ABSTRACT

This thesis addresses the problem of learning and recognizing human daily activities in smart environment. Most approaches offline learn the activity model and recognize the activity in an online phase. However, the activity models can be outdated when the human behavior and environment deployment change. It is a tedious and error-prone job to recollect data for retraining the activity models. In such case, it is important to adapt the learnt activity models under one context to another context without too much supervision. In this thesis, we present a self-reconfigurable approach for activity recognition can reconfigure a previously learned activity model to infer multiple activities under a dynamic environment meanwhile requiring minimal human supervision for labeling training data.

Keyword: Activity Recognition, Probabilistic Reasoning, Dynamic Bayesian Network, Active Learning

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Motivation

As the advancement of computer technology brings computer into human daily life, developing various context-aware applications is a key issue to improve the quality of life.

- **Health Care**

As the aging of population and the lacking of manpower, it is more and more important to make elderly people be able to live independently. The system will firstly monitor and gather the statistics of human daily behavior. Then, it will use the information to give health promotions or to notify elderly people when they forget to perform routine activities of daily living.

- **Children Care**

There are more and more double-income families, and the parents are suffering from giving consideration to both work and children. The system can monitor the children's activities, and it will launch a warning immediately, when the children do dangerous or inappropriate activities (referring to their ages). Moreover, it can record the children's activities and report the information to their parents.

- **Security and Surveillance**

  The system can build models of users' behavior over time and detect unusual events. Moreover, the system can differentiate users' identity by matching the behavior models.

(1.1)

- **Home Automation**

  In order to make inhabitants feel more comfortable, the system will predict users' activities and provide adequate services automatically (e.g. switching on a light when a resident is studying and the environment is dim).

- **Human-computer Interface**

  The smart environment systems have to interact with users. Real time activity recognition can help the system to present information to the users at the right time, and recognize user input.

Recognizing human activities is a key part to facilitate the context-aware applications. In general, an activity is a sequence of interactions between residents and objects in the environment. Sensors are deployed in the environment to sense these interactions such that the activity models can be learned by the activity recognition system. The activity models, which represent the relationship between interactions and activities, are further used to infer residents' activities. In order to associate semantic meaning with activities, we have to manually label the training data from sensor during the learning stage. It is a tedious, interruptive and error-prone job for end users to label the training data. Since we can not predefine the users' environment, the system can not be learned in advance. Instead, it needs to be learned under the users' situation. Even worse, the environment is usually dynamic in nature (e.g. adding or removing some objects, using objects with different purposes, etc.), which means that the relationship between interactions and activities may change with time. In this condition, the activity models need to be updated. In this thesis, we want to minimize the labeling effort from the end users when the activity models need to be updated.

1.2 Challenges of Activity Recognition

To recognize activities, the activity models that capture the structure of activities must be developed. However, the high variations in both human actions and the environment make the structure of activities hard to predefine. There are many human behavior and environment attributes that present the challenges for activity recognition:

- **Multiple residents**

  When there are multiple residents in the environment, the complexity of data association makes it hard to track every resident at the same time.

- **Multitasking**

  The resident often performs multiple activities simultaneously while some of them only engage little attention.

- **Periodic variations**

  Human behavior affects how they perform activities, and the behavior varies periodically (days, weeks, months, and even seasons). For example, a resident may wake up early on weekdays and late at the weekend.

- **Time scale**

  Humans perform activities in a very wide range of time scale, and it may vary with many reasons. It is impossible to know and consider all reasons.

- **Incomplete activity**

One activities may be interrupted by another that has caught resident's attention, and it is hard to know whether he will come back to finish the original one.

- **Environment variations**

The development of the environment also affects the residents' behavior. The high variations of environment make humans have many ways to perform activities, and we can not know every kind of variation in advance.

- **Representation power and complexity**

Different types of activities have different complexity. So many variables may affect how an activity will be performed, such as sequential order, time, location, human habit, etc. As we consider more variables, the representation ability of the activity recognition system is higher, but the complexity is also higher and causes more problems. Because we can not now the users' situation in advance, the trade-off between representation power and complexity is very hard to decide.

1.3 Objectives

The main goal of the activity recognition system in this thesis is: "On-line recognizing and recording daily activities in the home setting." To make the system have the capability to recognize activities, we need to build activity models that represent knowledge about the environment and the relationships between the environment and human activities first. In addition, the activity models can be outdated when the human behavior and environment deployment a change. In such case, it is important to adjust the learnt activity models under one context to fit the new context without too much supervision. We propose an activity recognition system which reaches the following objectives:

- **To build a flexible activity model**

Our system will gather two kinds of data: environment sensor data and activity labels. We use these data to build models for the relationships between interactions and activities. Our system can recognize activities based on the activity model. In addition, the activity model needs to have high flexibility so that it can easily be updated.

- **To on-line recognize activities in the environment**

Some context-aware applications, such as home automation, security, and human-computer interface, need real-time activity recognition. Thus, our system must on-line recognize activities and provide the information to those applications.

- **To self-reconfigure the activity model**

In a real smart home, the environment is dynamic in nature. Deployment of various devices may change over time. These changes we categorized into wto groups:

1. Changes of sensor deployment: This will directly affect the measurement space from sensors. For example, after adding or removing some sensors, the activity recognition system should properly respond to such a change so that the previously learned activity model can be adapted and then be applied to the new environment (, which is similar to but not totally the same as the old one).
2. Changes of object deployment: This affects users' interactions with objects; namely, the relationship between sensor measurements from the objects and human activities may change. For example, after the contents of a cabinet are changed, and then opening the same cabinet does not necessarily mean the same purpose any longer.

In a dynamic environment, an activity model needs to be retrained each time when any deployment changes. It becomes important for the system to keep as much knowledge from the old activity model as possible in response to the deployment change so that the activity model can automatically adapt (hereafter called self-reconfigurable) to the change. In addition, collecting training data and labeling them is a tedious and error-prone job. This motivates us to minimize the number of labeled instances for retraining.

- **To interact with residents actively for improving the system performance**

  In order to further improve the performance of self-reconfiguring, the system selects training data which are recognized with low confidence and actively request correct labels from users. Meanwhile, it can reduce the overall retraining effort.

1.4 Related Work

From our survey of prior approaches for activity recognition, we roughly divide them into two main categories. In the first category [1-8], they predefined what activity classes they want to recognize, and designed the activity models to capture the temporal characteristics of activity from their prior knowledge of the environment and activity definition. Most of previous approaches in this category were fundamentally grammar-driven (see e.g. [1, 5, 6, 9] and the references therein). They explicitly modeled activity structures followed by learning model parameters from training data, or by mining from the Web or commonsense databases [10, 11]. However, the activity structures were generally not known in advance (although [10, 11] automatically mining activity structures from the Web, but the mined models were for general purpose, and may not suit to all users' situations), because we often cannot know the situation of the environment beforehand, which makes it unlikely to design a generalized structure for all environments. Therefore, it is more desirable to discover the activity structures from training data, not statically defined in advance. Moreover, if the environment changes, it is necessary for users to adapt to the change by re-labeling the data for training the models. That is, each time any change in users' behaviors or the environment, users need to label new data for retraining the models. However, the change may exceed developers' prior expectations; for example, users may perform an activity in different ways from time to time, which will cause bad performances of the static model.

In the second category[12-17], they automatically extracted activity structure from training data by computing the local event statistics rather than by pre-defined (see e.g. [16, 17] using Latent Semantic Analysis [18-20], [13, 14] using n-grams [21], and [22-24]using Vector Space Model [25]). Because the resultant activity categories were automatically mined from the data, these approaches did not know or may be not able to label resultant activities with their corresponding semantic meaning. The representation power of the feature spaces was limited by the ability to capture the characteristics of activity structures only up to some fixed (gross-grained) resolution. Furthermore, since the representation power of the feature is the order over event statistics, the computational complexity would grow exponentially. Moreover, these approaches entailed a unique feature space that computed from training data (not easy to change); when the environment changes, the original feature space may not be appropriate for the new environment; therefore, it needed to reconstruct a new feature space completely.

In order to address the challenges of the high variation of activity structures and dynamic environment, we proposed an activity recognition approach that can be flexible to adapt to different contexts. Our approach first generates various features with different representation power. It will learn the corresponding meaning (activity) of the each feature in learning stage and select features with high discriminative power that further used in recognition stage. All kinds of features will be stored for training and dynamically used for recognition after training, thus they can be flexible to adapt to different conditions. When the environment changes, we assume that the change happens locally, not entirely covering the whole environment, therefor, we can apply Expectation-Maximization (EM) [26] strategy to self-reconfigure the activity model. In order to further improve the performance of self-reconfiguration from the EM strategy, the system takes advantages of active learning [27-31] to select potentially good training data for querying correct labels from users, thus reducing the overall retraining effort. Finally, to sum up, this self reconfigurable approach for activity recognition can reconfigure a previously learned activity model to infer multiple activities from multiple residents under a dynamic environment meanwhile requiring minimal human supervision for labeling training data.

1.5 Thesis Organization

This thesis consists of six chapters. The rest of this thesis is organized as follow: Chapter 2 presents the problem statement, system overview, and background knowledge of this thesis, including Dynamic Bayesian Networks (DBNs), semi-supervised learning, active learning, and incremental learning.

Chapter 3 explains how and why the components of the activity recognition system are design and implemented.

Chapter 4 analyzes how the environment dynamics affect the activity recognition system, and shows how the system self-reconfigure with less supervision.

In Chapter 5, the proposed activity recognition system is actually experimented and evaluated in a real home environment.

Finally, Chapter 6 summarizes conclusions of this work, and offers some suggestions for improving the current system in the future.

Chapter 2

Preliminaries

2.1 Problem Statement

In a dynamic environment such as homes and offices, there can be multiple activities occurring simultaneously. The problem of recognizing multiple activities can be formulated as estimating $P(A_t \mid z_{1:t}, M^{(k)})$, which denotes the probability distribution of activity vector A_t at time *t* given the sensor observations $z_{1:t}$ from time *1* to time *t*, and the current activity model *M*$^{(k)}$. To be more specific, A_t is an activity vector of *N* activities that we want to recognize and is defined as $A_t = \{A_t^1, A_t^2, ..., A_t^N\}$, $z_{1:t}$ are the measurements collected so far and can be expressed as $z_{1:t} = \{z_1, z_2, ..., z_t\}$ where $z_t = \{z_t^{ID_1}, z_t^{ID_2}, ..., z_t^{ID_{E(t)}}\}$ is the measurements of sensors at time *t*, and the number of sensors $E(t)$ will vary with time due to deployment change. $M^{(k)}$ is indexed with *k* (which is the times of the system reconfiguring) and models the relationship between sensor measurements and activities. It can be updated from old model $M^{(k-1)}$ and the previously collected training data set $D^{(k)}$.

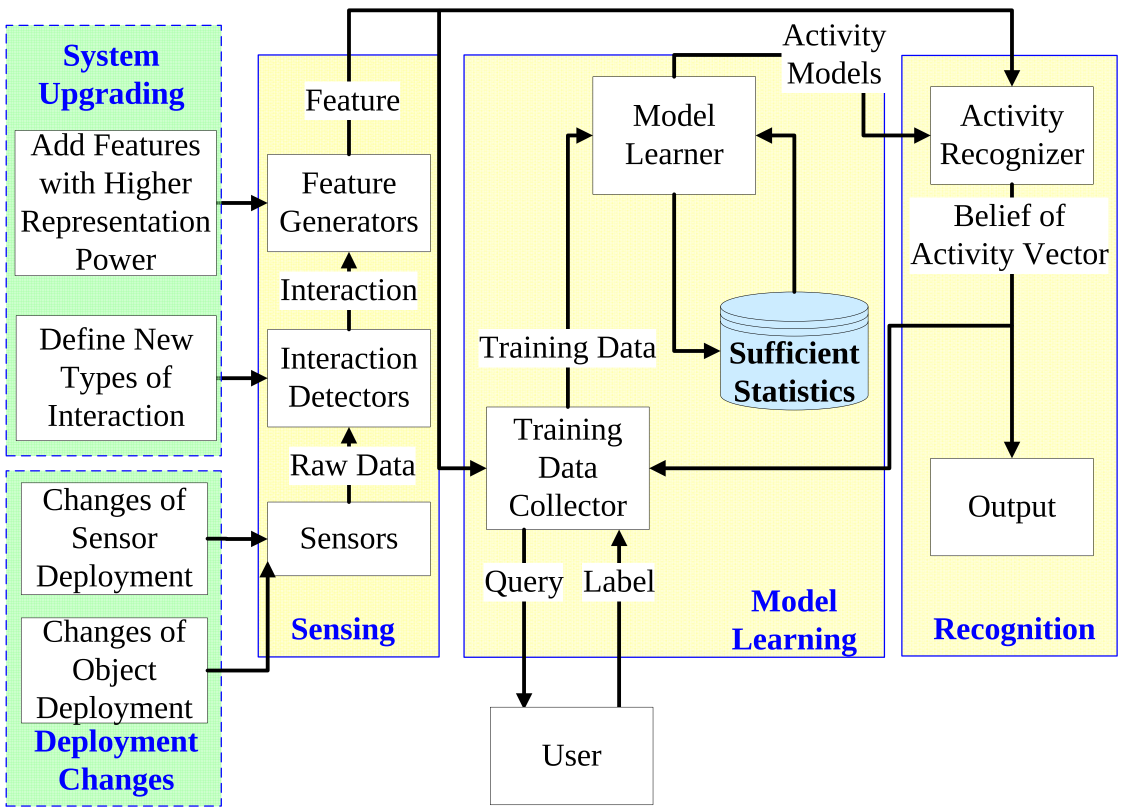


Fig. 2-1 System overview for the proposed self-reconfigurable activity recognition

Activity model $M^{(k)}$ needs to be reconfigured as soon as possible in response to any change of deployment. One of our goals is to minimize the labeling effort from end users. Each time the activity model $M^{(k)}$ is to be reconfigured, we want to select the instances that can maximally reduce the expected error rate from the training data set $D^{(k)}$ to query the users about their corresponding labels.

2.2 System Overview

As show in Fig. 2-1, the purposed activity recognition system has three main components, including sensing, model learning, and recognition. In the sensing component, sensors are installed on objects to sense the status of the objects. Sensor are connected to a circuit board, which will preprocess the raw data into interactions by interaction detectors (which are defined by the designer of the sensors) and then wirelessly send the interaction information to the recognition system. An interaction is a description about how the object is used; it is the basic element of an activity. The recognition system collects the interactions from sensors around the environment and then finds the relationships between interactions by feature generators. A feature is a description about how multiple interactions are triggered.

In the model learning component, the recognition system correlates the observed features and activity labels by learning the activity model from training data. The training data collector continuously stores the observed features and then off-line requests the activity labels from the users. The model learner processes the training data into sufficient statistics (which are the basic elements used to construct the activity model) and then stores them instead of entire training data into the database. After processing the training data, the model learner dynamically constructs the activity model from the sufficient statistics in the database.

Finally, the recognition component infers on-going activities from current observed features using the learned activity model and outputs the estimated belief of activities to further activity-aware applications.

As show in Fig. 2-1, there are two types of causes that force the recognition system to update its knowledge about activity: environment changes and system upgrading. Environment changes will change the measurement space or affect the meaning of sensor measurement. System upgrading can enhance the recognition system ability to capture more complex characteristics of activities. It includes defining new types of interactions and adding features with higher representation power to capture more complex relationships between interactions.

All those causes described above will increase the feature space or change the meaning of the features, and thus the recognition system need to re-correlate the observed features and activity labels by retraining the activity model from new collected training data. As the red line in Fig. 2-1 indicates, the recognition system feeds the output to the training data collector for further self-reconfiguring the activity model. The model learner will self-reconfigure the activity model by fusing the new training data and the old sufficient statistics in the database. In addition, users can correct the training data for improving recognition performance. The recognition system will actively request the activity labels of the training data instances which the system hard to predict the activity labels. By doing this, we can improve system performance with less supervision.

The system self-reconfigures the activity model every fixed period or after an intentional trigger. The procedure of the self-reconfiguration process is as follow:

1. Initially, there is an activity model $M^{(0)}$, which is initialized by the system or learned from a pilot training set under a given (and static) environment.
2. In each time period $k \geq 1$:
   - i. The system online recognizes activities using the activity model $M^{(k)}$ and outputs the results to the data collector and other applications.
   - ii. The training data set $D^{(k)}$ in the training data collector contains observed features and corresponding activity belief collected during this time period k, and the users can correct them from the active query if necessary.
   - iii. At the end of this time period k, the system will reconfigure the activity

model $M^{(k)}$ to a new one $M^{(k+1)}$ using training data $D^{(k)}$. The data collection pool will be reset to empty, and k increases by one.

2.3 Dynamic Bayesian Networks (DBNs)

Sequential data analysis arises in many areas of science and engineering. For example, in robotics, one may be interested in estimating the location of the robot from sequential sensor measurements; in speech recognition, one may be interested in recognizing words from sequential audio input. In this thesis, we are interested in knowing what people are doing in the environment from sequential sensor measurements.

Dynamic Bayesian networks (DBNs) [32] are approaches to analyze sequential data, which extends the Bayesian network (BN) [33] to handle time series by modeling sequences of variables. DBNs assume that there is some underlying hidden state of the world that generates the observations, and the hidden state evolves in time. In this thesis, the hidden states are activities we want to estimate, and observations are sensor measurements. In online analysis, where the data arrives in real-time, one common task is to estimate the current hidden states X_t, given all the observations up to the present time, denoted as $z_{1:t} = \{z_1, z_2, ..., z_t\}$. More precisely, the goal is to compute $P(X_t \mid z_{1:t})$, which is referred to as belief state.

In the following subsections, we will discuss how to represent DBNs, how to use them to update the belief state and perform other related inference problems, and how to learn such models from data.

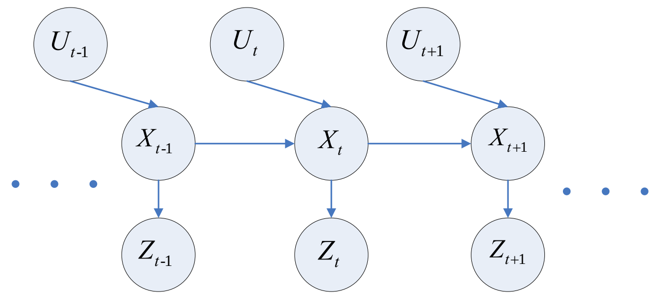


Fig. 2-2　An example of a DBN with first order Markov assumption

2.3.1 DBNs: Representation

A DBN models probability distributions over semi-infinite collections of random variable, $V_1, V_2, \cdots$, where $V_t = \{U_t, X_t, Z_t\}$ are input, hidden and output variables at time t of a dynamic system. A DBN is defined as $\{B_1, B_{\rightarrow}\}$, where B_1 is a BN which defines the prior $P(V_1)$, and $B_{\rightarrow}$ is a two-slice temporal Bayes net which defines the temporal dependencies $P(V_{t+1} | V_t)$ by a directed acyclic graph (DAG) as follows:

$$P(V_{t+1} | V_t) = \prod_{i=1}^{N} P(V_{t+1}^i | parents(V_{t+1}^i)) \tag{2.1}$$

where V_t^i is the *i*th node at time *t*, and $parents(V_t^i)$ are the parents of V_t^i in the graph. The edges in the graph represent dependencies; and there are two kinds of dependencies, namely, within a time slice and across the time slice, each of them associated with a conditional probability distribution (CPD). We assume the parameters of the CPDs are time-invariant, and thus a DBN can be defined by three types of parameters, initial probabilities, transition probabilities (dependencies across the time slice), and conditional probabilities within a time slice. The joint distribution of a DBN with length T and with N random variable in each time slice is:

$$P(V_{1:T}) = \prod_{t=1}^{T} \prod_{i=1}^{N} P(V_t^i | parents(V_t^i)) \tag{2.2}$$

Fig. 2-2 is an example of a DBN with first order Markov assumption (which only dependency is only on the last time slice). The parameters of this DBN are initial probability $P(X_1)$, control probability $P(X_t \mid U_t)$, observation probability $P(Z_t \mid X_t)$, and transition probability $P(X_{t+1} \mid X_t)$. If we use this DBN to localize the location of a robot, U_t is the control input at time *t*, X_t is the location of the robot at time *t*, and Z_t is the sensor measurement of the robot at time *t*. The joint distribution of the hidden random variables given control input and sensor measurements is:

$$P(X_{1:T}) = P(X_1)P(Z_1 \mid X_1)\prod_{t=2}^{T} P(X_t \mid X_{t-1}, U_t)P(Z_t \mid X_t) \qquad (2.3)$$

Hidden Markov models (HMMs) is the basic type of DBNs. We can design various types of DBNs by adding more random variables or dependencies to the DBN to model more complex dynamic system. For example, in activity recognition; we may consider higher order of the dependencies, or jointly estimate location and activity [7], or use multiple sensors [6]. There are many types of DBNs which have been proposed for various purposes. Factorial HMMs [34] jointly estimate more then one hidden variables by assuming they are independent (for reducing the computational complexity). Coupled HMMs [35] model the binary interactions between hidden variables. Hierarchical HMMs [36] model domains with hierarchical structure and/or dependencies at multiple time scales. Variable-duration (semi-Markov) HMMs [3] model the duration as other arbitrary distribution rather than exponential one.

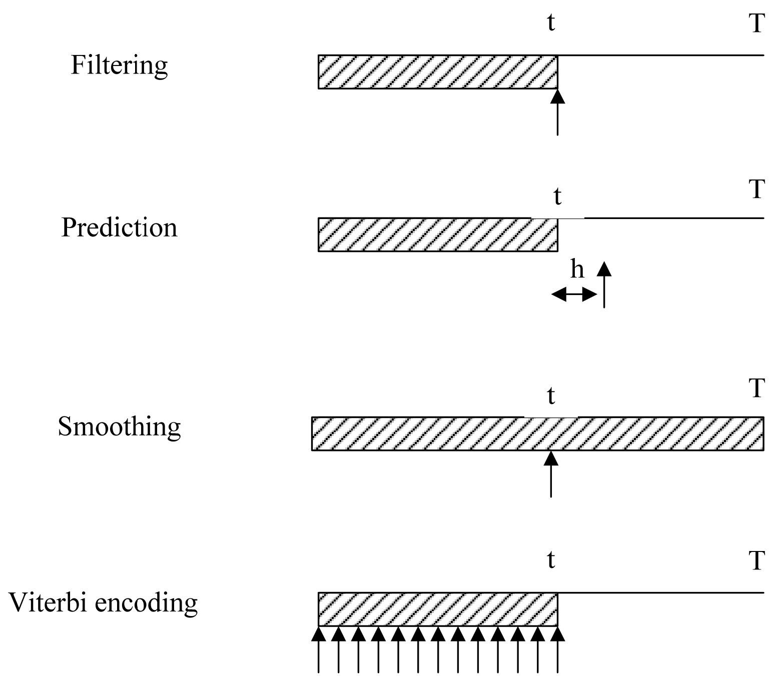


Fig. 2-3 Four main kinds of inference in DBNs. The slash region is the interval we have data, *t* is the current time instant, *T* is the sequence length, and the arrow is the time step we want to estimate.

2.3.2 Inference in DBNs

As summarized in Fig. 2-3, there are four main kinds of inference problem:

- **Filtering**

This is the most common inference problem in online analysis. Given the observations collected so far, we want to estimate current belief state of the hidden variables, using Bayes' filter:

$$\begin{aligned} P(X_t \mid z_{1:t}) &\propto P(z_t \mid X_t, z_{1:t-1}) P(X_t \mid z_{1:t-1}) \\ &= P(z_t \mid X_t) \sum_{x_{t-1}} P(X_t \mid x_{t-1}) P(x_{t-1} \mid z_{1:t-1}) \end{aligned} \tag{2.4}$$

There are two assumptions (Makrov assumptions); replacing $P(z_t \mid X_t, z_{1:t-1})$ by

$P(z_t \mid X_t)$ by assuming current observations only depend on current hidden variable; and decomposing $P(X_t \mid z_{1:t-1})$ by assuming current hidden variable only depend on pervious one (order one Markov assumption). The filtering task can be decomposed into two steps of recursive computation: prediction, which computes $P(X_t \mid z_{1:t-1})$, and update, which computes $P(X_t \mid z_{1:t})$.

- **Prediction**

Prediction is to predict the future state, i.e. compute $P(X_{t+h} \mid z_{1:t})$, where $h > 0$ is how far we want to look-ahead. We also can predict the future observations by marginalizing out hidden variable X_{t+h}:

$$P(Z_{t+h} = z \mid z_{1:t}) = \sum_x P(Z_{t+h} = z \mid X_{t+h} = x) P(X_{t+h} = x \mid z_{1:t}) \tag{2.5}$$

- **Smoothing**

Smoothing is to estimate the state of in the past given all the observations, i.e. compute $P(X_t \mid z_{1:T})$ for all $1 \le t \le T$.

- **Viterbi decoding**

Viterbi decoding is to compute the “most probable explanation”, i.e. to compute the most likely sequence of hidden states given observations collected so far:

$$x^*_{1:t} = \arg\max_{x_{1:t}} P(x_{1:t} \mid z_{1:t}) \tag{2.6}$$

By distributive law of multiplication and dynamic programming, we can compute the Viterbi decoding using forward pass filtering (replace sum with max):

$$\alpha_t(j) = P(z_t \mid X_t = j) \max_i P(X_t = j \mid X_{t-1} = i) \alpha_{t-1}(i) \tag{2.7}$$

where

$$\alpha_t(j) = \max_{x_{1:t-1}} P(X_{1:t-1} = x_{1:t-1}, X_t = j \mid z_{1:t}) \tag{2.8}$$

In activity recognition, these four kinds of inference mechanisms have their corresponding meaning and applications. Filtering means online tracking human's current activity from sensor measurements; prediction means to predict human activity, which can be used for automatically providing services; smoothing is important for learning; Viterbi decoding is used to offline recognize human activity, and the output can be used to gather the statistics of human activity.

2.3.3 DBNs: Learning

A DBN usually has some free parameters θ, which are used to define initial probability $P(X_1)$, transition probability $P(X_t \mid X_{t-1})$, and observation probability $P(Z_t \mid X_t)$. Learning is to estimate these parameters from training data. Suppose that we have training data $D = \{D^{(1)}, D^{(2)}, ..., D^{(K)}\}$, where $D^{(k)} = \{x_{1:T^{(k)}}, z_{1:T^{(k)}}\}$ and all sequences are iid, then maximum likelihood estimation (MLE) is to find parameters that maximize the likelihood to observe the training data:

$$\theta^*_{MLE} = \arg\max_{\theta} P(D \mid \theta) = \arg\max_{\theta} \log P(D \mid \theta) \tag{2.9}$$

where the log likelihood of the training data is:

$$\log P(D \mid \theta) = \log \prod_{k=1}^{K} P(D^{(k)} \mid \theta) = \sum_{k=1}^{K} \log P(D^{(k)} \mid \theta) \tag{2.10}$$

Another variation is maximum a posteriori (MAP) which includes a prior on the parameters:

$$\theta^*_{MAP} = \arg\max_{\theta} \log P(D \mid \theta) + \log P(\theta) \tag{2.11}$$

This may be useful when the number of free parameters is much greater than the size of the training data set. The prior term acts like a regularization term to prevent over-fitting.

Those described above is the case with supervised learning. If the training data set does not contain values of hidden variable X, then it is unsupervised learning case; if the training data $D^{(k)}$ arrives with time rather than altogether, it is on-line learning case. In the following sections, we will describe more in detail about various learning problems.

2.4 Sufficient Statistics

In the model learning problems, we want to estimate the free parameters of the model from training data set by a criterion such as MLE. The parameter estimation can be derived from some statistics of the training data. A statistic is a well-behaved function of the data, which is what actually used in calculations or inferences, rather than the full data set; for example, the sample mean, the sample median, the sample variance, etc. A statistic is sufficient if it is just as informative as the full data. Once we have known the sufficient statistic, nothing else, not even the original data, it can tell us anything more about the parameters. This means in parameter estimation, we can only store the sufficient statistics rather than whole training data set. For example, if we use a binomial distribution to model the flipping of a coin, for estimating the parameters of the binomial distribution, we only need to know the total counts of heads and tails, rather than the whole sequence of flipping.

2.5 Semi-supervised Learning

In here, we consider the MLE case as the learning goal, which is finding parameter values that maximize likelihood of the training data. Semi-supervised learning problem is that the training data set consists of labeled and unlabeled samples (or called partially observed, means values of some random variables are missing). This learning problem is useful in the case where getting label of the training sample is expensive (such as in activity recognition, requesting users' current activity is very annoying).

In the partially observed case, the log-likelihood of the training data is:

$$\begin{aligned} L(D \mid \theta) &= \sum_{k=1}^{K} \log P(D^{(k)} \mid \theta) \\ &= \sum_{k=1}^{K} \log \sum_{h} P(H = h, V = D^{(k)} \mid \theta) \end{aligned} \tag{2.12}$$

where we need to sum up the probability of all kinds of assignments of the hidden variable *H*, and $V = D^{(k)}$ means the values of visible nodes are specified by $D^{(k)}$. Because the summation of hidden variables makes this equation unable to be decomposed into a sum of local terms. In the following, we will introduce the expectation-maximization (EM) algorithm to find the local maximum of the likelihood.

2.5.1 Expectation-Maximization (EM) algorithm

The basic idea of EM algorithm is to apply Jensen's inequality to get a lower bound on the log-likelihood of the training data, and then to iteratively maximize this lower bound:

$$\begin{aligned} L(D \mid \theta) &= \sum_{k=1}^{K} \log \sum_{h} P(H = h, D^{(k)} \mid \theta) \\ &= \sum_{k=1}^{K} \log \sum_{h} q(h \mid D^{(k)}) \frac{P(h, D^{(k)} \mid \theta)}{q(h \mid D^{(k)})} \\ &\geq \sum_{k=1}^{K} \sum_{h} q(h \mid D^{(k)}) \log \frac{P(h, D^{(k)} \mid \theta)}{q(h \mid D^{(k)})} \\ &= \sum_{k=1}^{K} \sum_{h} q(h \mid D^{(k)}) \log P(h, D^{(k)} \mid \theta) - \sum_{k=1}^{K} \sum_{h} q(h \mid D^{(k)}) \log q(h \mid D^{(k)}) \end{aligned} \tag{2.13}$$

where q is a function such that $\sum_h q(h \mid D^{(k)}) = 1$ and $0 \le q(h \mid D^{(k)}) \le 1$. Maximizing the lower bound with respect to q gives:

$$q(h \mid D^{(k)}) = P(h \mid D^{(k)}, \theta) \tag{2.14}$$

This is called E (Expectation) step, which means calculating the expectation value of the hidden variable given observation and model parameters; and this makes the bound tight. Maximizing the lower bound with respect to the free parameters θ is equivalent to maximizing the expected complete-data log-likelihood:

$$\sum_{k=1}^{K} \sum_h q(h \mid D^{(k)}) \log P(h, D^{(k)} \mid \theta) \tag{2.15}$$

This is called M (Maximization) step. In this step, the free parameters are calculated from expected sufficient statistics (the values of hidden variables are expectation value).

The whole EM algorithm is iteratively calculating the E-step and M-step until the log-likelihood of the training data converging to a local maximum. Because the initial values of the free parameters will greatly influence the convergence, we can try different initial values of the free parameters to find better local maximum.

2.6 Active Learning

In many machine learning applications, the most time-consuming and costly task is the collection of a sufficiently large training data set. Active learning [28, 31] is a learning mechanism to reduce the requirement of large number of training samples by actively selecting potentially good samples from a pool under request. In here, “potentially good” means it can reduce the most the expected error of the model after training. Most of active learning approaches determine a training sample is good or not by calculating the hardness to classify the sample, such as normalized entropy of the predicted label:

$$H(z) = \frac{-\sum_{X} P(X \mid z) \log B(X \mid z)}{\log N(X)} \tag{2.16}$$

where z is a piece of unlabeled data, $P(X \mid z)$ is the distribution of the corresponding predicted label which is calculated by a model we want to train, and $N(X)$ is number of possible states of the label X.

In this thesis, we apply active learning to semi-supervised learning, which selects good training samples from the unlabeled data pool to request the label, thus reducing the labeling effort from the users.

2.7 Online Learning

In many real applications, the training data $D = \{D^{(1)}, D^{(2)}, ..., D^{(K)}\}$ are received with time, not received totally at once. On-line learning is a learning mechanism that incrementally learns the model as the new training data are received. In here, we consider the on-line learning in a semi-supervised learning case. In this case, the EM algorithm computing the expected sufficient statistics (ESS) needed for the EM update involve summing over all training cases. [37] modified the EM algorithm that updates the parameters per little batch of the training data $D^{(k)}$ (called on-line or incremental EM). In this thesis, because the training data are received continuously, stop less, we modify the incremental EM to prevent the training data set gets too large that makes the learning procedure very long.

Chapter 3

Activity Recognition System in a Static Environment

In this chapter, we will detail how the proposed activity recognition system works in a static environment, including how and why the components are designed and implemented. In a static environment, the recognition system has to learn only once the environment has been completely setup. In the next chapter, we will describe how the proposed recognition system deals with the environment changes.

3.1 Overview

The design of the proposed activity recognition system considers the following two objectives:

- **Easy to install**

Users may not have clues about how much information in the system they can access, and the building blocks for the system may not be cost-effective enough to manufacture, deploy, and maintain. We hope that the system can be easily installed to users' existing environment and do not affect the users too much, thus making the system widely acceptable to the public.

- **Configurable to different environment settings**

Because we can not know the users' environment in advance, the recognition system needs to have the ability to configure itself to various situations flexibly.

Fig. 3-1 shows the overview of the proposed activity recognition system for a static environment (does not include the red dotted lines and the indistinct component). In order to reach the objectives described above, the proposed recognition system contains three major components:

1. The sensing component: The easily installed environment sensors are widely

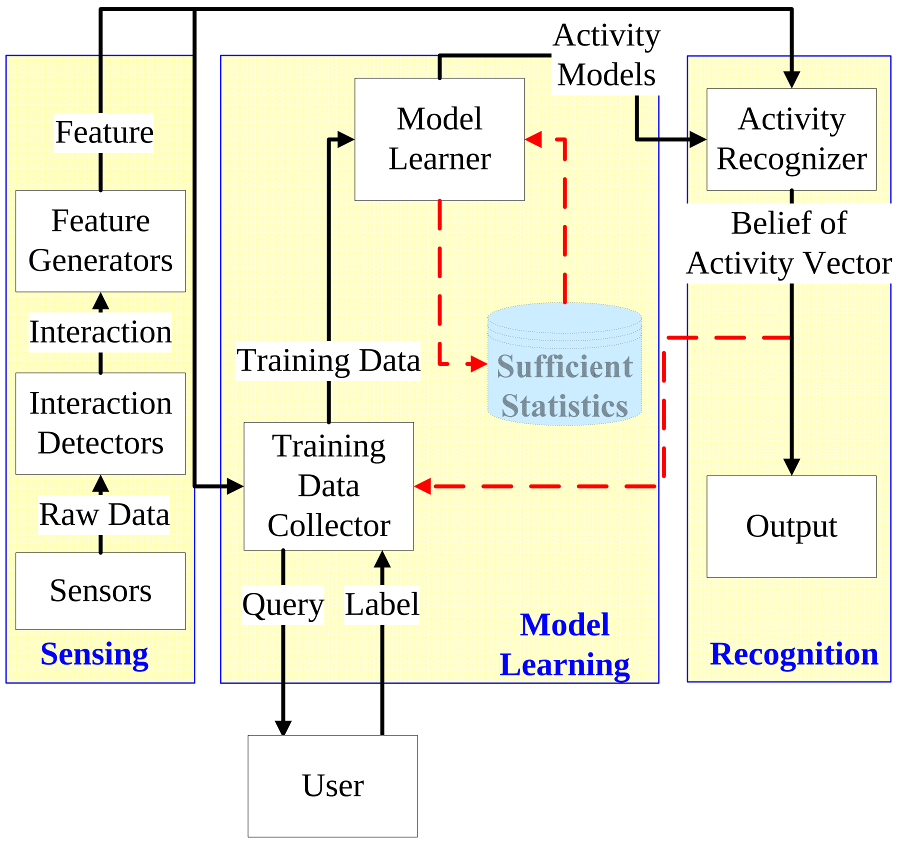


Fig. 3-1 System overview of activity recognition system for static environment (does not include the red dotted lines and the limpid part)

deployed in the environment for collecting information about how the objects are used in the environment.

2. The model learning component: The flexible and expansible activity modeling strategy makes the system can adapt to different situation with less human supervision.
3. The recognition component: The efficient on-line activity recognition algorithm can be executed to estimate users' on-going activities.

The sensing part includes multi-modal environment sensors that are deployed around the environment to sense the status of the objects in the environment and interaction detectors can detect the patterns of state change of the sensors. The feature generators capture the relationships among interactions. The model learning part learns the correlations between the observed features and activity labels from collected training data. The recognition part infers on-going activities from current observed features using the learned activity model and outputs the estimated belief of activities to further activity-aware applications.

During the learning stage, the system collects a batch of observed features calculated from the sensor measurements, and then it off-line requests the users to give the activity labels. The model learner creates models of activities based on the relationship between features and activity labels. In the recognition stage, the system estimates the belief of activity status from observed features and learned activity models. The following sections will explain the components of the recognition system in more detail.

3.2 Environment Sensors and Interaction Detectors

We define activities as sequences of interactions between inhabitants and objects in the environment where an interaction is a description of how the object is used and is a basic component of an activity. We deploy various types of sensors around the environment to sense the status of the objects. Sensors are mounted on the objects and connected to a circuit board (hereafter called Taroko) which can preprocess the measurement data and send the results wirelessly to a remote system.

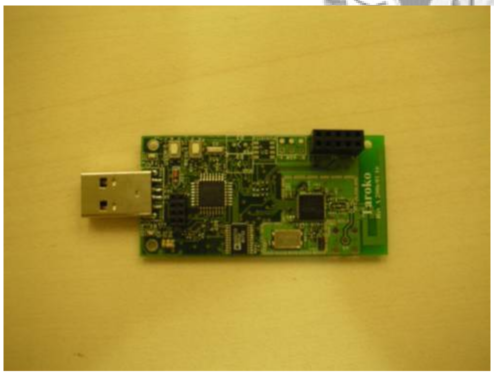
3.2.1 Sensor Deployment

The goal of sensor system is to design suitable sensors for different types of objects, to deploy a sufficient number of sensors in the environment, to left unattended, and to collect synchronized data. In order to achieve this goal, the sensor needs to have the following characteristics:

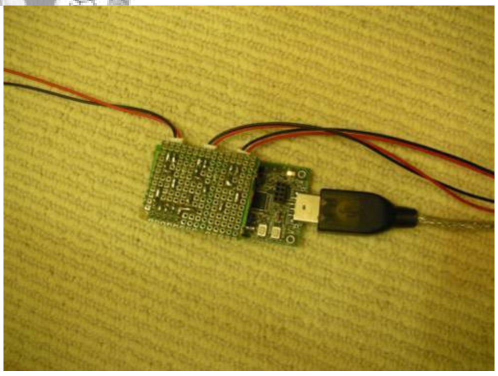
1. It is low-cost so that we can afford to deploy a sufficient number of sensors in the environment.
2. In can send the measurement data wirelessly so that it can be deployed everywhere.
3. Its size is small so that it can be installed and hidden easily.
4. It has low power consumption so that it can be powered by a small battery and left unattended for a long period.
5. It has high reliability.

6. Its module can be replaced by various types of sensors so that it is easily to customize.

Fig. 3-2 shows the circuit board of a Taroko and how various types of sensors connect to it. The board can connect at most eight sensors and is powered through a USB (Universal Serial Bus) interface (The USB can connect to a battery or a power regulator). The board includes a microprocessor (along with a programmable flash memory) that can samples and preprocesses the sensor measurement and sends the information wirelessly. Various types of sensors can be directly powered from the board and controlled by the microprocessor. In order to decrease the power consumption and the network interference, the board sends the information in an event-based manner. Fig. 3-3 is examples of how sensors and the corresponding circuit board deployed in the environment.



(a)



(b)

Fig. 3-2 (a) The circuit board (Taroko) with a programmable microprocessor that can control the sensors connected to it and preprocess the measurements of the sensors and send the information wirelessly. (b) The circuit board module can connect various types of sensors (at most eight sensors) and powered through USB interface (The USB can connect to a battery or a socket).

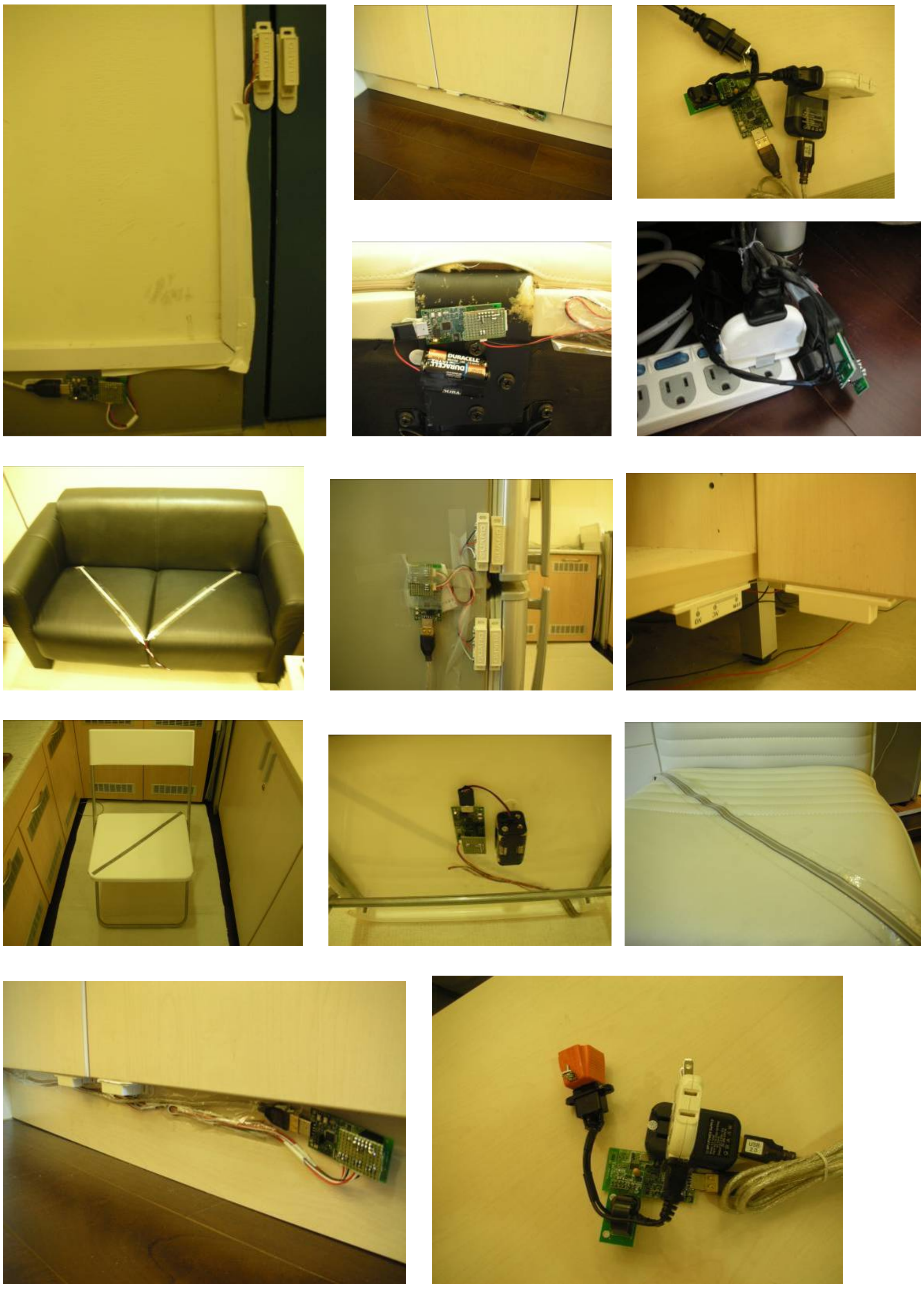


Fig. 3-3 Examples of how sensors and the corresponding circuit board deployed in the environment.

3.2.2 Interaction Detectors

An interaction is a description of how the object is used. The interaction detector is a binary function where input is a sequence of measurement and where output is a binary value that determines whether the specific interaction pattern happens or not. We can define various types of interactions for each type of sensor based on domain knowledge. For example, Fig. 3-4 illustrates the raw data of a pressure sensor and the corresponding responses from interaction detectors. The pressure sensor can measure how much force is on it, and we can use it to collect the location information of inhabitants by the change of pressure value. The pressure sensors can be installed in the floor, sofa, chair, bed, etc. Currently, we have defined two types of interactions on the pressure sensor, "Pressing" and "Pressing Still", which means there is a significant

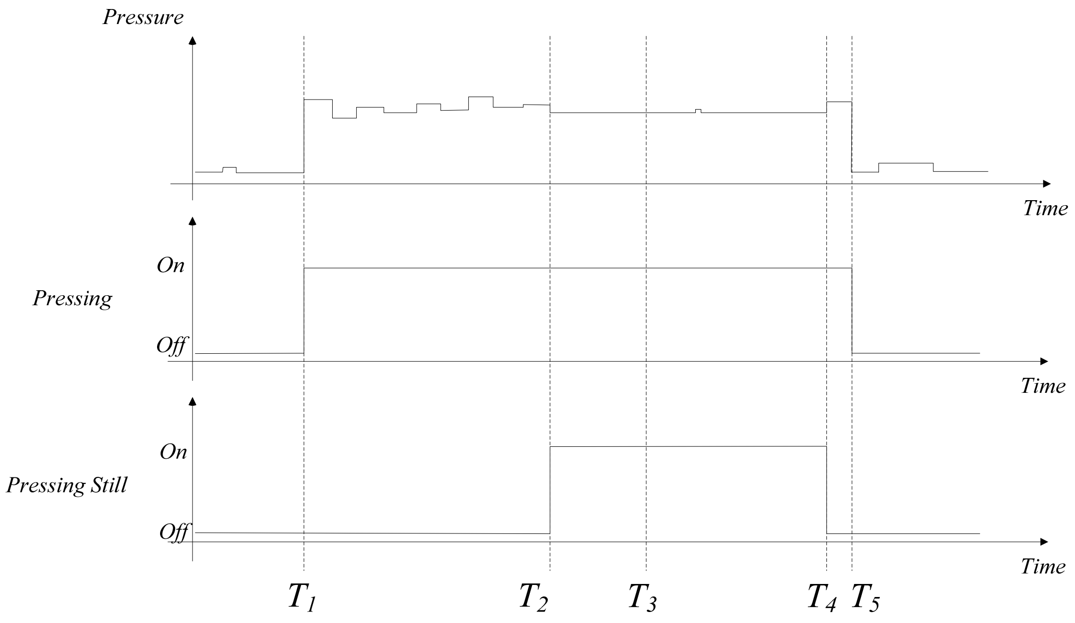


Fig. 3-4 An example of raw data from a pressure sensor and the corresponding interaction detectors

pressure on it (may can be interpreted as someone sitting on it) and there is a significant pressure on it and no change for a while (may can be interpreted as someone sitting on it without moving for a while) respectively. The interaction detector of “Pressing” is a function that determines whether the filtered pressure measurement is greater than a threshold or not. The interaction detector of “Pressing Still” is a function that determines whether the filtered pressure measurement is greater than a threshold and the variance is less than a threshold or not.

The functions of interaction detectors can be programmed and run in the microprocessor on the circuit board. As the measurement data are received, the interaction detectors will calculate the function described above, and the circuit board will wirelessly output an event to the system only when function output changes. The event format is shown in Table 3-1. The event ID is a unique value. The interaction ID consists of the sensor ID and the interaction type. For example, “P1_Pressing” means the ”Pressing” interaction on pressure sensor P1, and “C1_Use” means the “Use”

Event ID	Interaction ID	State	Time
1	P1_Pressing	On	2008/05/21 19:18:26
2	C1_Use	On	2008/05/21 19:18:39
3	P1_PressingStill	On	2008/05/21 19:31:55
4	P1_PressingStill	Off	2008/05/21 20:22:47
5	C1_Use	Off	2008/05/21 20:36:33

Table 3-1 An example of interaction detecting events

interaction on current sensor C1 (current sensor is installed on the electrical appliance to measure the current usage). The state attribute records whether the corresponding interaction occurs or not (On/Off), and the time attribute records the time instant of the occurrence of the event. Note that although detection of the interactions might have some delay (caused by the processing data window), but the time attribute of the event is not affected. For example, in Fig. 3-4, The “Pressing Still” interaction started at time T_2, but detection has some delay (detected at time T_3) cause by the data window of determining whether the variance is less than a threshold or not, however, the time attribute of the event which sent to recognition system is still T_2.

Many types of interaction detectors can be defined for each kind of sensor based on the domain knowledge. We assume that the designers of sensors are responsible designing the corresponding interaction detectors.

3.3 Activity Modeling

The proposed environment sensing sub-system and interaction detectors provide information about what happens in the environment. The goal of the activity recognition system is to interpret the sequences of interactions into activity labels. In order to accomplish this goal, we have to model the relationships between the interaction sequences and activity labels, and to recognize the activities from newly incoming interaction sequences.

The following reasons motivate the design of activity modeling in this work:

- **Multitasking**

It is natural that the resident often performs multiple activities simultaneously although some of them only engage little attention. The recognition system must have the ability that recognizes multiple activities at the same time.

- **Flexibility to control the complexity of the activity model**

Different kinds of activity have different level of complexity. Even the same kind of activity may have different characteristics with different levels of complexity under different environments or associated with different behavior. If we use a complex model to capture the characteristics of activities, although the model would have high representation power and could recognize more complex activities, but it also needs a large number of training data; or it may have bias, and the recognized activity may be too specific. The high computational complexity may also cause the system bad performance. On the other hand, if the model is too simple to capture the characteristics of activities, then the system may not have enough ability to recognize some kinds of activities. Because we can not anticipate the users' situation, it is hard to decide the complexity of the model in advance. Thus, it is very important for the recognition system to be able to tune the complexity of activity models for different types of activities flexibly under various situations from learning procedure.

- **Semi-supervised learning**

The environment and its furnishings have highly variable layouts, and individuals can perform activities in many ways. It is impossible to know all kinds of situations in advance and thus the system has to learn the activity models from residents' daily life. At the learning stage, the user has to give the desired output (activity labels) of the input (detected interactions). However, labeling the training data for model learning is a tedious and error-prone job. Semi-supervised learning is a learning mechanism that uses labeled and unlabeled data simultaneously. It allows activities to be represented by various interactions from different environments and individuals. Furthermore, semi-supervised learning mechanism can train the model using a less number of labeled data and a large number of unlabeled data, and thus reducing the training effort from end users.

- **Probabilistic reasoning**

Probability reasoning is a good way to deal with the uncertainty from the ambiguous and noisy observations from multiple sensors. Furthermore, the probabilistic representation of the recognition output can provide more information than yes/no to the subsequent applications that need the activity information. It also provides a way to estimate the expected error rate of recognition results.

- **Model-based learning**

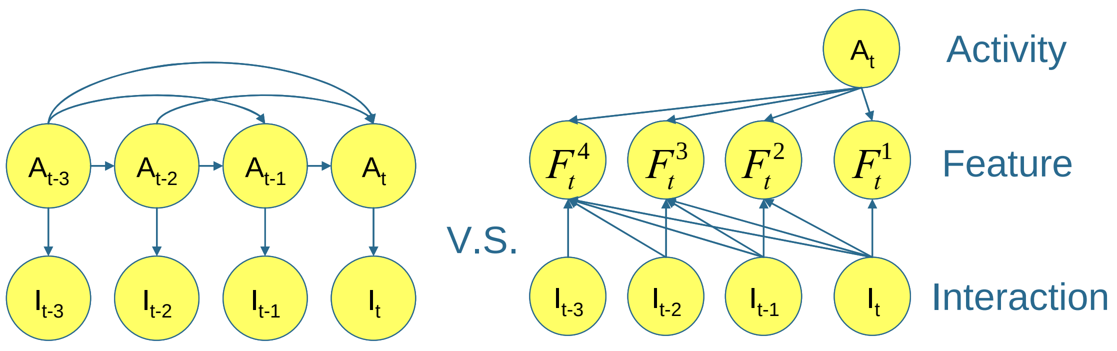
Model-based learning uses the training data to construct models to represent the distribution of data. Once the learning procedure finishes, the training data can be discarded. This can reduce the system memory required to store a large number of data and also relieve the users' privacy concerns.

- **Real-time performance**

Some subsequent applications may need the real-time activity information. This requires a trade-off between models, features, and the computational complexity.

- **Online learning**

Because of the environment changes and other reasons (which will be discussed in the next chapter), the recognition system needs to continuously update its knowledge about the relationships between interactions and activities. In such a situation, the training data is continuously received, and the learning algorithm needs to efficiently fuse the old and new coming data to update the activity model, instead of totally retraining the model.



(a) (b)

Fig. 3-5 Model (a) models the long-term dependencies in the model level, and model (b) models the long-term dependencies in the feature level.

In order to recognize multi-tasking activities, our recognition system aims directly at recognizing what activities are occurring in the environment by tracking the states of activities (hereafter called activity vectors) instead of each individual. As shown in Fig. 3-5, model (a) models the dependencies in the model level, if it need to be adjust to different situation, the whole needs to be retrained; model (b) models the dependencies in feature level, it can flexibly consider features with different level of dependency, thus flexibly control the complexity of the model. The recognition system uses various types of features instead of complex model to capture the different level of complexity of the characteristics of the activity structure. By doing this, the recognition system can flexibly control the complexity of the activity model by selecting suitable features with different representation power in the learning stage rather than changing the entire model. The recognition system uses Dynamic Bayesian Networks (DBNs) with an efficient on-line and semi-supervised learning strategy to model the activity. Because the recognition system selects only the features that have the most influence on the recognition performance in the learning stage, so the system can on-line recognize multiple activities with less computational effort (and shorter delay caused by the data window of the interaction detectors). The following two sub-sections will describe the cooperation between feature generation and activity model in more detail.

3.3.1 Feature Generation

In order to make the recognition system able to control the complexity of the activity model more flexibly, we propose to represent the model complexity in the feature level instead of in the model level. We define features with various representation powers and then select the appropriate features during learning. There are three types of features:

- **Single-interaction feature**

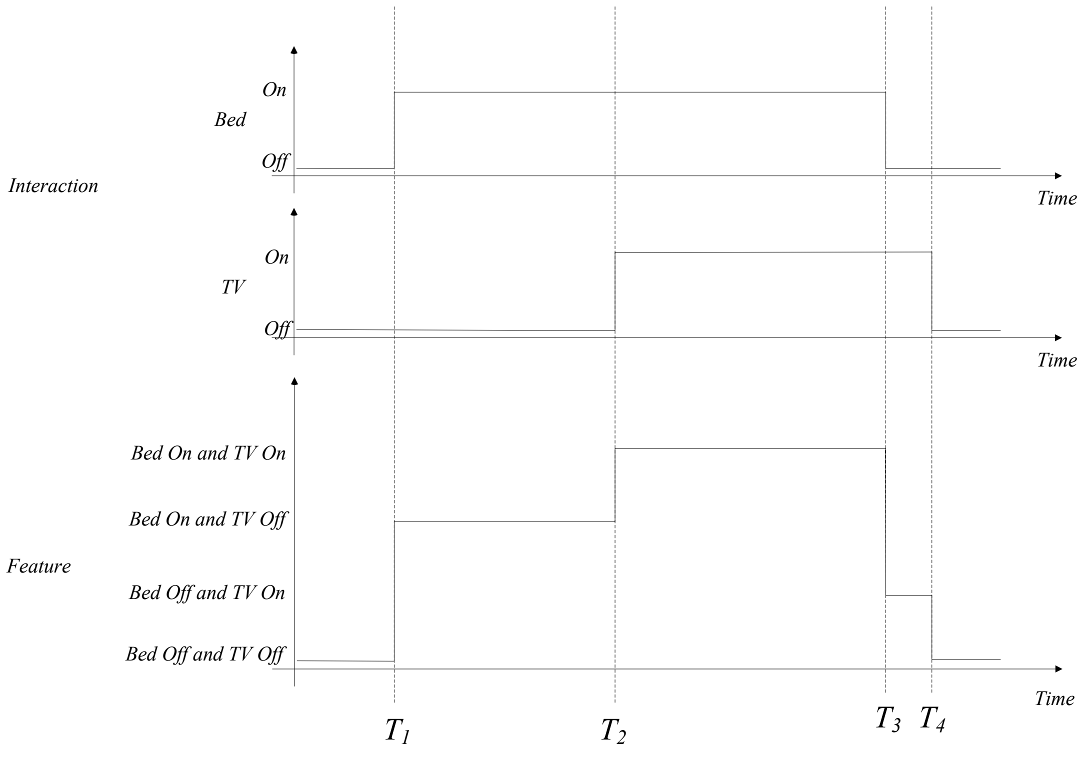


Fig. 3-6 An example of interactions and the corresponding multiple-interaction feature

It is the most basic type of feature, and captures the status of a single interaction.

- **Multiple-interaction feature**

In some cases, the occurrences of interactions are not independent. For example: “lying on the bed” may mean “sleeping”, but “lying on the bed” and “turning on the TV” may mean “watching TV”. This type of feature captures the relationships among multiple interactions. Fig. 3-6 shows an example of a two-interaction feature.

- **N-gram feature**

One of the challenges of activity recognition that is discussed in Section 1.2 is to capture the temporal information and encode it into the model. This type of feature captures the temporal information among interactions, for example: “go out” and “come back” may include the same interactions but with different orders. Fig. 3-7 shows an example of bi-gram features.

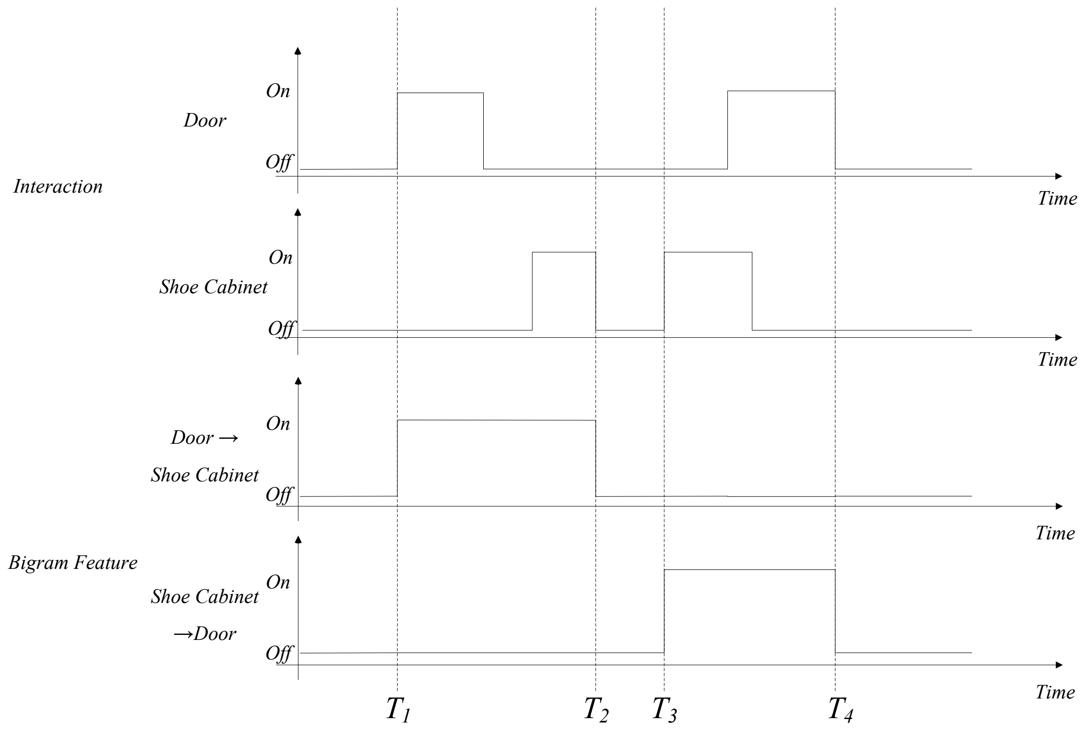


Fig. 3-7 An example of interactions and the corresponding bi-gram features

3.3.2 Activity Model

As the feature generation part generates various types of features that capture the temporal characteristics of activity and the relationship among interactions, the activity model correlate the features and activity labels. The activity model is depicted in Fig. 3-8 as a Dynamic Bayesian Network (DBN) with first order Markov chain assumption. Each time instant *t* is the timing of the event that any feature state changed. As shown in Fig. 3-8 (a), $O_t = \{f_t^{ID_1}, f_t^{ID_2}, ..., f_t^{ID_L}\}$ is the state vector of all features at time *t*, which is the observation of this model, $A_t = \{A_t^1, A_t^2, ..., A_t^N\}$ is the activity vector at time *t* that we want to estimate. At each time instant, the activity vector A_t is estimated from the observation O_t, and the temporal dependency (from A_t to A_{t+1}) is used to filter the recognition result for reducing the effect of noise. Fig. 3-8 (b) shows that our activity model selects useful features for every activity types. $O_t^i = \{f_t^{ID_1}, f_t^{ID_2}, ..., f_t^{ID_{L(i)}}\}$ is the state vector of selected features of activity type A^i, and hence $O_t^i \subseteq O_t$. For simplification, we assume that the occurring of each type of activities is independent; and thus the state transition probability distribution of each activity type is independent:

$$P(A_{t+1} \mid A_t, O_{t+1}) = \prod_{i=1}^{N} P(A_{t+1}^i \mid A_t^i) P(A_{t+1}^i \mid O_{t+1}^i) \tag{3.1}$$

Fig. 3-8 (c) shows the observation distribution of each type of activity. We assume each observed feature $f_t^{ID_j}$ of each activity type is independent, and thus the observation distribution of *i*th type activity at time instant *t* is defined as:

$$P(A_t^i \mid O_t^i) = \prod_{j=1}^{L(i)} P(A_t^i \mid f_t^{ID_j}) \tag{3.2}$$

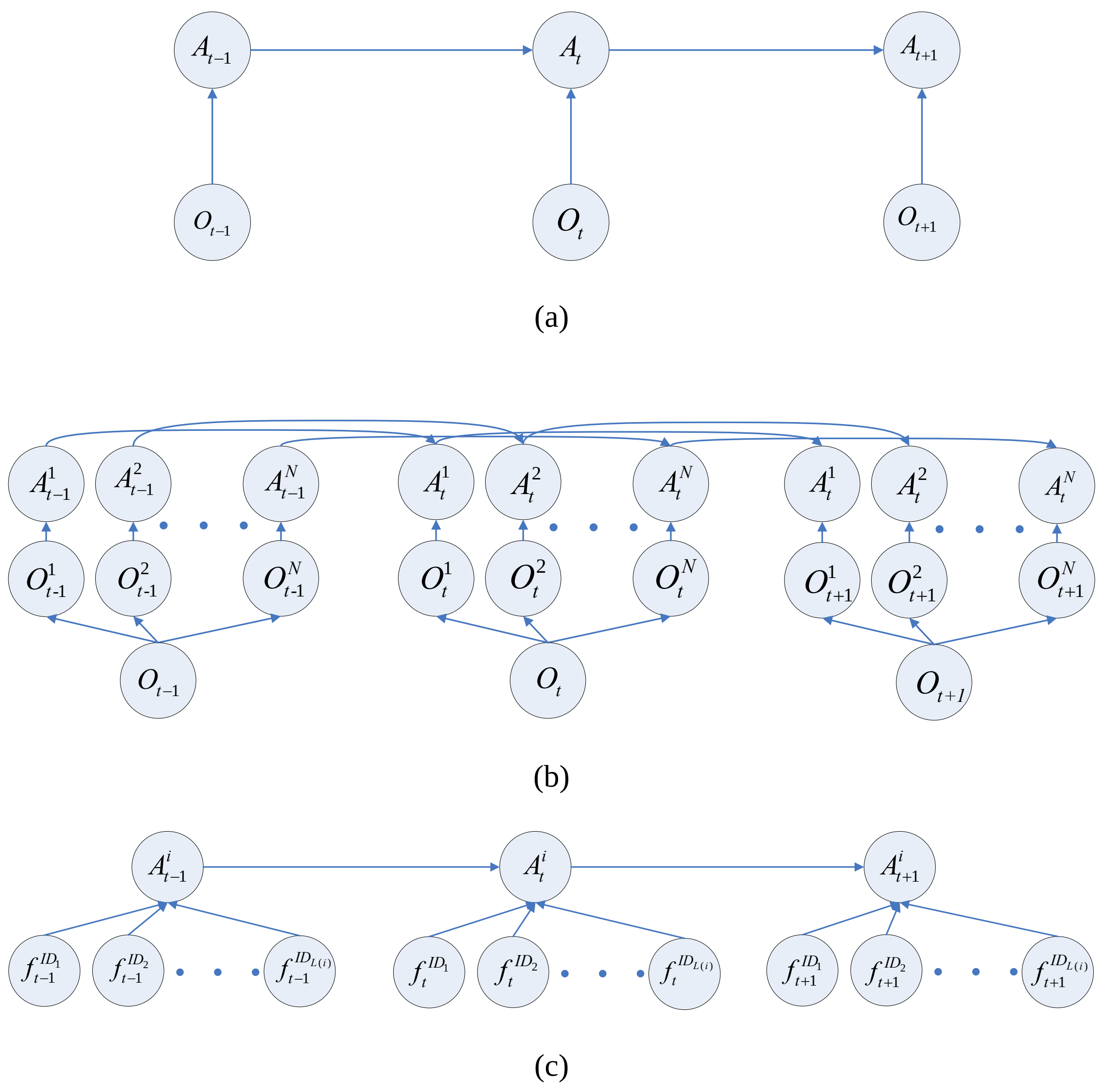


Fig. 3-8 Our activity model of a Dynamic Bayesian Network (DBN) with first order Markov chain assumption. Each time instant *t* is the timing of the event that any feature state changed. (a) $O_t = \{f_t^{ID_1}, f_t^{ID_2}, ..., f_t^{ID_L}\}$ is the state vector of all features at time *t*, $A_t = \{A_t^1, A_t^2, ..., A_t^N\}$ is the activity vector at time t that we want to estimate. (b) $O_t^i = \{f_t^{ID_1}, f_t^{ID_2}, ..., f_t^{ID_{L(i)}}\}$ is the state vector of selected features of activity type A^i, $O_t^i \subseteq O_t$, and we assume each type of activities are independent. (c) The observation distribution of i'th type activity: we assume each observed feature $f_t^{ID_j}$ is independent.

The advantages of this activity model are as follow:

- **Probabilistic representation**

The probabilistic representation has the ability to capture the uncertainty, and it can be used to calculate expected error for making optimal decisions.

- **Combines prior knowledge and observations**

It combines the prior distribution (recognition result of last time instant) and the observation distribution. This reduces the effect of observation noise.

- **Simple recognition algorithm**

The independent assumptions of the observation and long-term dependency (we extract the dependency into feature level) greatly simplify the on-line recognition procedure. This makes the system be applicable to real-time recognition.

- **Online learning with fast update**

The model parameters can be on-line updated (just accumulate the counting). This offers a possibility for adapting the activity model to various variations over time.

- **Multi-label classification**

The recognition system tracks the state of each activity over time, and thus each data instance can have multiple activity labels. This makes the system able to handle the multitasking situation.

- **Flexibly using features with different complexity**

The system can use various features with different representation power to capture the characteristic of an activity. This allows the recognition system to be able to flexibly tune the complexity of the model for different types of activity under various situations by only changing the connections of features, not the entire model.

3.4 Model Learning

In this section, we will discuss how the system selects important features for each type of activity and learns the parameters of activity model from labeled data. For simplification, we assume that the user only can edit activity label at the time instant when the event occurs, and the activity labels are unchanged during two successive events (the same as our recognition system).

Fig. 3-9 shows the learning procedure of the activity model. First, in training data collection, the system collects the observed features (preprocessed data) for a time period, and then requests the corresponding activity labels from the users. Second, the system calculates the sufficient statistics of the collected training data which will de used in feature selection and parameter estimation. Third, the system constructs the activity model by selecting useful features of every activity types. Fourth, the system estimates the parameters of the activity model. In the following three subsections, we will detail the three components of the learning procedure: feature selection, parameter estimation, and the final one is how the sufficient statistics (which are used in feature selection and parameter estimation) are calculated.

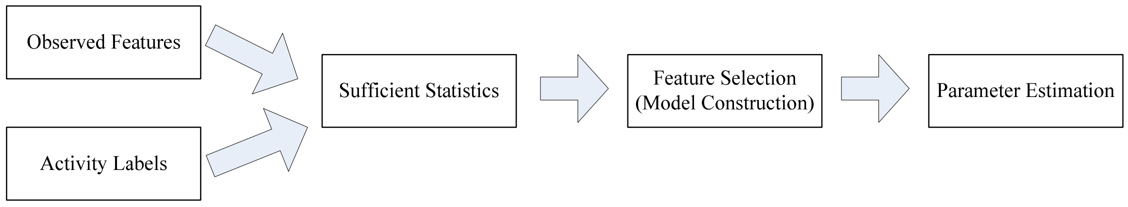


Fig. 3-9 The learning procedure of the activity model.

3.4.1 Feature Selection

As the feature generators generate various features with different representation power, the system selects the features that have the most influence on the recognition performance during learning stage. In feature selection, the system selects useful features for every activity type by computing the corresponding weight. Because each state of a feature has different influence (for example, if we observe usage of the microwave, we may believe there is a “preparing food” activity happening; however, if we do not observe the usage of the microwave, because there exist other means for “preparing food”, then this observation will not influence the belief for “preparing food” activity), thus they have different weights. First, we define the weight of a feature j with state f^j to activity type i as follows:

$$w_i(F^j = f^j) = \sum_{a^i \in A^i} P(A^i{=}a^i, F^j{=}f^j) \left| \log \frac{P(A^i{=}a^i, F^j{=}f^j)}{P(A^i{=}a^i)P(F^j{=}f^j)} \right| \tag{3.3}$$

where the probabilities are calculated as follows:

$$P(A^i{=}a^i, F^j{=}f^j) = \frac{S(A^i{=}a^i, F^j{=}f^j)}{\sum_{a \in A^i, f \in F^j} S(A^i{=}a, F^j{=}f)} \tag{3.4}$$

$$P(A^i{=}a^i) = \frac{S(A^i{=}a^i)}{\sum_{a \in A^i} S(a)} \tag{3.5}$$

$$P(F^j{=}f^j) = \frac{S(F^j{=}f^j)}{\sum_{f \in F^j} S(f)} \tag{3.6}$$

where the S function is the sufficient statistics calculated from training data set (we will describe how the calculation is done in Subsection 3.4.3). This weight measures the degree of dependency between $F^j = f^j$ and A^i, where greater value means greater dependency, and thus higher influence.

After computing the weights of every feature value on every activity type, the system selects the useful features by clustering all the weights into two groups, selected and non-selected.

3.4.2 Parameter Estimation

After feature selection, the structure of the activity model is constructed, and the next step is to estimate the parameters. There are three types of parameters in the activity model needed to be estimated: initial probability π, transition probability α, and observation probability β. We use maximum likelihood estimation (MLE) to estimate theses parameters:

$$\begin{aligned} &M = \{G, \theta\} \\ &\theta = \{\pi, \alpha, \beta\} = \arg\max_{\pi,\alpha,\beta} P(D \mid \pi, \alpha, \beta) \end{aligned} \tag{3.7}$$

where G is structure of the activity model, θ is the parameters we want to estimate, and D is the training data set. The training data set $D = \{O_t, A_t\}_{t=1:T}$ consists of observed feature O_t and corresponding activity labels A_t at each event time instant t, and T is the total length of the training data set D.

Given $\pi = \{\pi_1, \pi_2, ..., \pi_N\} = \{P(A_1^1), P(A_1^2), ..., P(A_1^N)\}$; the MLE of the initial probability distribution $\pi_i = P(A_1^i)$ of the *i*th activity type is calculated as follows:

$$P(A_1^i{=}a^i) = \frac{S(A^i{=}a^i\ )}{\sum_{a \in A^i} S(A^i{=}a)} \tag{3.8}$$

$\alpha = \{\alpha_1, \alpha_2, ..., \alpha_N\} = \{P(A_{t+1}^1 \mid A_t^1), P(A_{t+1}^2 \mid A_t^2), ..., P(A_{t+1}^N \mid A_t^N)\}$; the MLE of the transition probability distribution $\alpha_i = P(A_{t+1}^i \mid A_t^i)$ is calculated as follows:

$$P(A_{t+1}^i = a' \mid A_t^i = a) = \frac{S(A_{t+1}^i = a', A_t^i = a)}{\sum_{a^i \in A_{t+1}^i} S(A_{t+1}^i = a^i, A_t^i = a)} \tag{3.9}$$

$\beta = \{\beta_1, \beta_2, ..., \beta_N\}$, $\beta_i = \{P(A^i \mid F^{ID_1} = f^{ID_1}), ..., P(A^i \mid F^{ID_{L(i)}} = f^{ID_{L(i)}})\}$, where $L(i)$ is the total number of selected features of *i*th activity type. The MLE of the observation probability distribution $P(A^i \mid F^{ID_j} = f^{ID_j})$ is calculated as follows:

$$P(A^i = a^i \mid F^{ID_j} = f^{ID_j}) = \frac{S(A^i = a^i, F^{ID_j} = f^{ID_j})}{\sum_{a \in A^i} S(A^i = a, F^{ID_j} = f^{ID_j})} \tag{3.10}$$

3.4.3 Used Sufficient Statistics in Learning Procedure

In the feature selection and parameter estimation, the probabilities are computed from the sufficient statistics of training data. In here, because our recognition system is event based, except $S(A_{t+1}^i = a', A_t^i = a)$ is defined as the frequency counting, others are defined as the accumulated total time period of the event rather than instance number.

3.5 Activity Recognition

As shown in Fig. 3-10, in the recognition step, the system continuously calculates the feature values, and then the recognition algorithm estimates the state of each activity type using Bayes filter:

$$\begin{aligned} &P(A_t = a_t \mid O_{1:t}) \\ &= P(A_t = a_t \mid O_t) \cdot \sum_{a' \in A} \left\{ P(A_t = a_t \mid A_{t-1} = a') \cdot P(A_{t-1} = a' \mid O_{1:t-1}) \right\} \end{aligned} \tag{3.11}$$

Because we assume the occurrence of each activity type is independent, the computation of each activity type can be separated. The computation of the *i*th type of activity is:

$$
\begin{aligned}
&P(A_t^i = a_t^i \mid O_{1:t}^i) \\
&= P(A_t^i = a_t^i \mid O_t^i) \cdot \sum_{a' \in A^i} \left\{ P(A_t^i = a_t^i \mid A_{t-1}^i = a') \cdot P(A_{t-1}^i = a' \mid O_{t-1}^i) \right\} \qquad (3.12) \\
&= \prod_{j=1}^{L(i)} P(A_t^i = a_t^i \mid F_t^{ID_j} = f_t^{ID_j}) \cdot \sum_{a' \in A^i} \left\{ P(A_t^i = a_t^i \mid A_{t-1}^i = a') \cdot P(A_{t-1}^i = a' \mid O_{t-1}^i) \right\}
\end{aligned}
$$

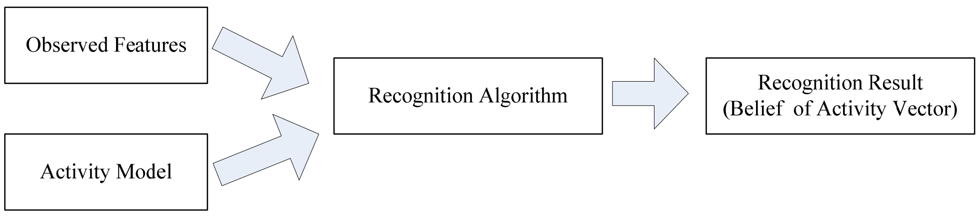


Fig. 3-10 The recognition procedure of the activity model.

Chapter 4

Activity Recognition System in a Dynamic Environment

In Chapter 3, we detail how the proposed activity recognition system works in a static environment. In this chapter, we will describe how the proposed recognition system deals with the environment changes.

4.1 Overview

In the real environment such as home, it is dynamic in nature. Deployment of various devices may change over time, and we categorize them into the following:

- **Changes of sensor deployment**

Adding or removing objects/sensors in the environment will directly change the measurement space from sensors, and thus affect the meaning of the corresponding interactions and features. The activity recognition system should learn the knowledge about the new sensors and the corresponding interactions and features, and remove the influence of removed sensors and corresponding interactions and features. In addition, the events of adding/removing objects may cause changes of the inhabitant behavior to perform activities. For example, after adding a TV in front of the bed, lying on the bed may not just for resting, since it is possible for watching TV. The activity recognition system needs to re-correlate the features and activities.

- **Changes of Object deployment**

This affects users' interactions or behaviors with respect to objects; namely, the relationship between sensor measurements from the objects and activities may change. For example, moving a sofa in front of a TV from the living room to a study room, and sitting on the same sofa does not necessarily mean watching TV any longer; or after the contents of a cabinet are changed, opening the same cabinet does not necessarily mean the same purpose any longer.

Because of the change of objects and sensors, how an activity is performed in the training phase may be significantly different from that in the application phase after a time period. For instance, a user uses a broom to do cleaning activity, and sensors sense the interactions; however, after a few days, the user buys a new vacuum cleaner and uses it to clean the house. The training data for the two cleaning activities will be very different from each other. As a result, activity recognition based on a static activity model may gradually become obsolete and inaccurate for a dynamically changed environment. In addition, in order to recognize more complex activities, we can upgrade the ability of the system to capture more complex characteristics of an activity:

- **Define new types of interaction**

In our activity recognition system, interaction is the basic element of an activity, and it is based on a specific pattern of sensor measurements. We can define new types of interaction based on an existing kind of sensor to capture another characteristic of an activity. For example, we can mount an accelerometer sensor on an object to detect the movement of the object. In the beginning, we may only define an interaction based on the accelerometer sensor as follows: whether the object is used or not can be detected by telling whether the accumulation of acceleration is greater than a threshold or not. Afterwards, we find that the accelerometer sensor can also be used to detect a specific motion pattern of movement (such as shaking or falling) involved in other complex activities (for example, the motion pattern of using a broom to sweep the floor), which thus defines new types of interaction.

- **Add features with higher representation power**

The feature generators generate various features with different level of

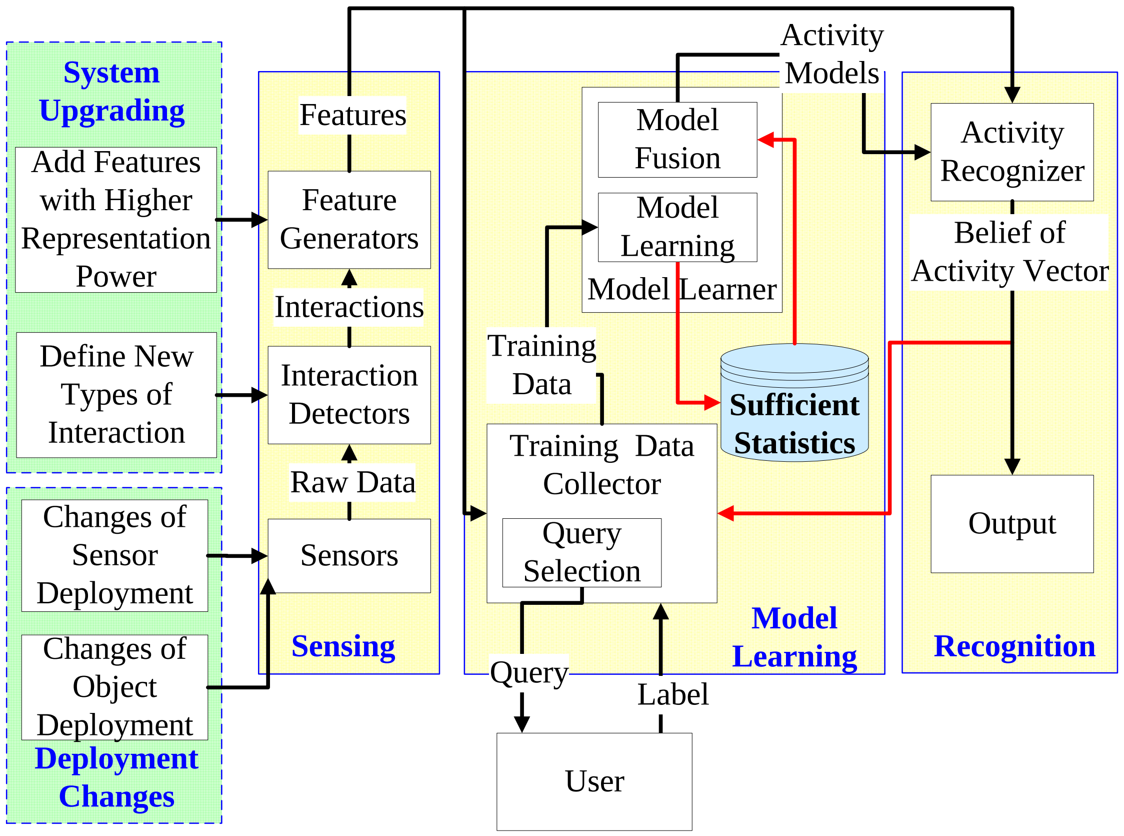


Fig. 4-1 System overview of self-reconfigurable activity recognition system

representation power. However, the complexities of features are bounded within a threshold (for examples, in multiple-interaction features, we may only consider the number of level less than three; or in *n*-gram features, we may only consider bi-gram), so that the ability of the recognition system to capture complex activity is bounded. We can raise the complexity threshold to capture more complex characteristics of activities.

Each time after any deployment changes or system upgrades, the activity model needs to be retrained from newly collected training data. However, collecting training data and labeling them is a tedious and an error-prone job. This motivates us to minimize the number of labeling instances for retraining; and therefore it becomes important for the system to keep much knowledge from the prior activity model in response to the deployment change such that the activity model can automatically adapt (hereafter called self-reconfigurable) to the change, rather than totally retrain the model. By achieving this self-reconfigurable algorithm, we can reduce the training effort and shorten the overall training periods. In addition, in order to further improve the performance of merely self-reconfiguring, the system takes advantages of active learning to select potentially good training data for querying correct labels from users, thus reducing the overall retraining effort.

As show in Fig. 4-1, we keep the influence of the deployment changes and the system upgrades in the sensing component. It only affects the meaning and space of the observed features, not the entire model. The activity recognition system just needs to update the feature selection and parameters of the activity model. The activity model is updated from newly collected training data, includes observed features and the activity labels. Because the originally collected training data are unlabeled, we estimate the expected values of the labels using the previously learned activity model (the red line which feeds the output of the recognition system to the input of the training data collector). Because the new incoming training data may not be rich enough, the retrained activity model may have bias in such situation, and thus it is very important to fuse global and local activity model. In here, "global model" means it is trained from all training data (collected from beginning to present), and "local model" means it is trained from newly collected training data. The system will store sufficient statistics of the global activity model for the purpose of fusion. In addition, the recognition system will select useful examples to query the user in order to labels instances with more useful value, which will improve the training performance more efficiently (we will describe in Section 4.3).

4.2 Self-reconfiguring

The goal of self-reconfiguring is to on-line update the knowledge of activities in

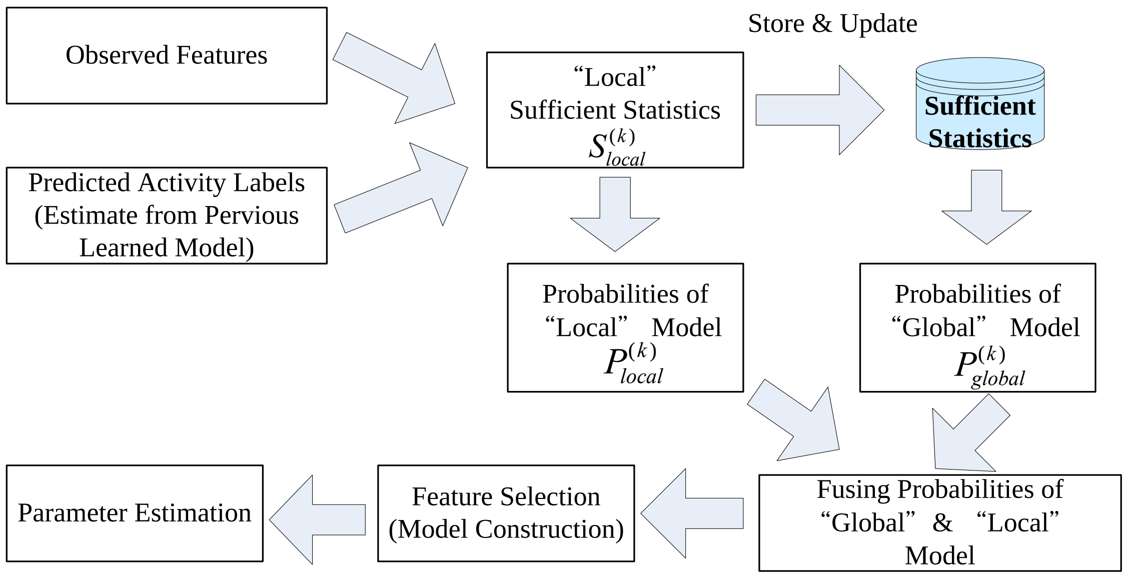

Observed Features
Store & Update
"Local" Sufficient Statistics $S^{(k)}_{local}$
Sufficient Statistics
Predicted Activity Labels (Estimate from Pervious Learned Model)
Probabilities of "Local" Model $P^{(k)}_{local}$
Probabilities of "Global" Model $P^{(k)}_{global}$
Parameter Estimation
Feature Selection (Model Construction)
Fusing Probabilities of "Global" & "Local" Model

Fig. 4-2 The learning procedure of reconfiguring the activity model.

the activity model, including feature selection and model parameters. Fig. 4-2 shows the learning procedure of reconfiguring the activity model. First, the system collects the training data, including observed features and the corresponding predicted activity labels (being estimated from the previously learned model). Second, the system computes sufficient statistics $S_{local}^{(k)}$, which are computed from the newly collected training data and will be used in model construction. Third, the system stores the new computed sufficient statistics $S_{local}^{(k)}$ into database and updates $S_{global}^{(k)}$, which is the sufficient statistics of the global model. Fourth, the system computes the probabilities of the global model $P_{global}^{(k)}$ and the local model $P_{local}^{(k)}$ (the computation is the same as those described in Subsections 3.4.1 and 3.4.2), and then fuses them. Finally, the system selects the useful features and estimates the model parameters using the fused probabilities.

At the *k*th time update of the activity model by the system, the training data set $D^{(k)} = \{O_t^{(k)}, B^{(k)}(A_t)\}_{t=1:T^{(k)}}$ includes observed features $O^{(k)} = \{O_1^{(k)}, O_2^{(k)}, ..., O_{T^{(k)}}^{(k)}\}$ and the belief of the predicted activity vector $B^{(k)}(A) = \{B^{(k)}(A_1), B^{(k)}(A_2), ..., B^{(k)}(A_{T^{(k)}})\}$, where $T^{(k)}$ is the length of the training data set. The system computes the expected value of the activity label from the previously learned activity model $M^{(k-1)}$ using Bayes filter:

$$\begin{aligned} B^{(k)}(A_t^i) &= P^{(k)}(A_t^i \mid O_{1:t}^i, M^{(k-1)}) \\ &= P^{(k)}(A_t^i \mid O_t^i, M^{(k-1)}) \sum_{a' \in A^i} \left\{ P^{(k)}(A_t^i \mid A_{t-1}^i = a', M^{(k-1)}) P^{(k)}(A_{t-1}^i = a' \mid O_{1:t-1}^i, M^{(k-1)}) \right\} \end{aligned} \tag{4.1}$$

where $B^{(k)}(A_t^i)$ is the state belief of *i*th type of activity at time instant *t* in the training data set $D^{(k)}$.

The system computes the expected local sufficient statistics $S_{local}^{(k)}$, which includes

$S^{(k)}_{local}(A^i{=}a^i, F^j{=}f^j)$, $S^{(k)}_{local}(A^i{=}a^i)$, $S^{(k)}_{local}(F^j{=}f^j)$, and $S^{(k)}_{local}(A^i_{t+1}=a', A^i_t=a)$, from $O^{(k)}$ and $B^{(k)}(A)$ by multiplying the belief of activity vector:

$$S^{(k)}_{local}(A^i{=}a^i, F^j{=}f^j)=\sum_{t=1}^{T^{(k)}}\{B^{(k)}(A^i_t{=}a^i)S^{(k)}_{event}(A^i_t{=}a^i, F^j_t{=}f^j)\} \tag{4.2}$$

$$S^{(k)}_{local}(A^i{=}a^i)=\sum_{t=1}^{T^{(k)}}\{B^{(k)}(A^i_t{=}a^i)S^{(k)}_{event}(A^i_t{=}a^i)\} \tag{4.3}$$

$$S^{(k)}_{local}(F^j{=}f^j)=\sum_{t=1}^{T^{(k)}}S^{(k)}_{event}(F^j_t{=}f^j) \tag{4.4}$$

$$S^{(k)}_{local}(A^i_{t+1}=a', A^i_t=a)=\sum_{t=1}^{T^{(k)}}\{B^{(k)}(A^i_{t+1}{=}a')\bullet B^{(k)}(A^i_t{=}a)\} \tag{4.5}$$

Where the function $S^{(k)}_{event}$ is defined as the time period of the event.

After computing the expected local sufficient statistics, the system updates the global sufficient statistics as follows:

$$S^{(k)}_{global}=S^{(k-1)}_{global}+S^{(k)}_{local} \tag{4.6}$$

Then, the system computes the probabilities, $P^{(k)}_{global}$ and $P^{(k)}_{local}$, for model construction (the same as those described in Subsections 3.4.1 and 3.4.2), which are computed from the “global” and “local” sufficient statistics respectively. Next, the system fuses the probabilities of local and global models according to the following fprmula:

$$P^{(k)}=w(T^{(k)})\bullet P^{(k)}_{local}+(1-w(T^{(k)}))\bullet P^{(k)}_{global} \tag{4.7}$$

where $w(T^{(k)})$ is used to control the fused probabilities $P^{(k)}$ so that they can be closer to local model’s or to global model’s, defined as:

$$w(T^{(k)})=1-e^{-\eta T^{(k)}} \tag{4.8}$$

where η is the learning rate. If $T^{(k)}$ is longer, the value of $w(T^{(k)})$ is higher, and thus the fused probabilities $P^{(k)}$ are closer to local model’s. Finally, the system selects useful features and estimates the parameters from the fused probabilities $P^{(k)}$ (the computation is the same as those described in Subsections 3.4.1 and 3.4.2).

4.3 Active Learning for Activity Label Requirement

As shown in Fig. 4-3, in the learning procedure of self-reconfiguring, the training data set are unlabeled. The user can label the unlabeled training data for improving the performance of the system (the label will directly change the activity belief). In here, we employ an active learning strategy which will select training instances for requesting labels with priority. The priority of a training example $D_t^{(k)} = \{O_t^{(k)}, B^{(k)}(A_t)\}$ to activity type A^i is:

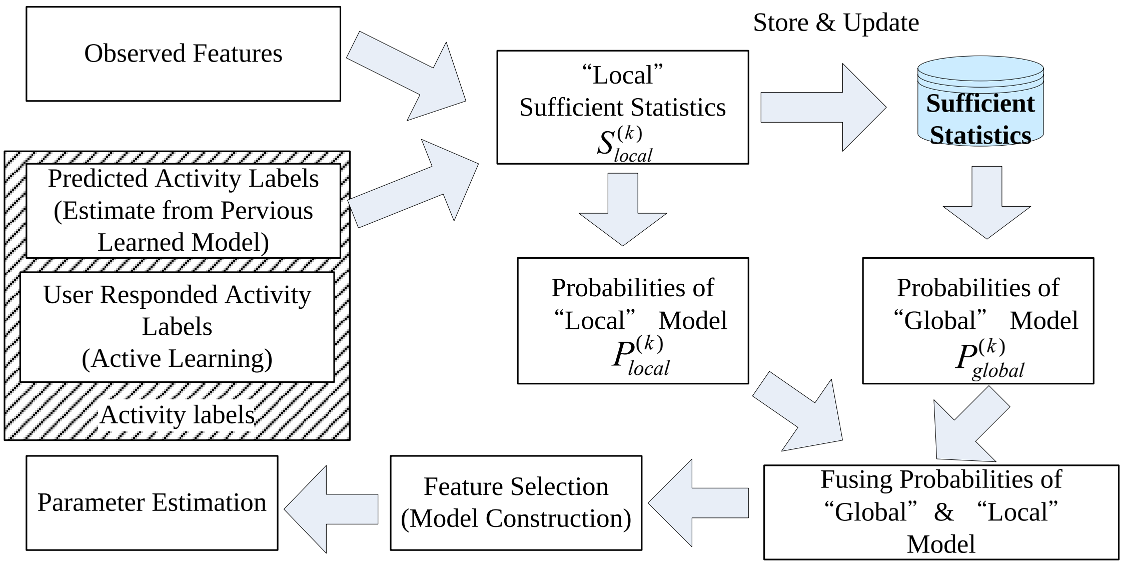

Observed Features
Store & Update
"Local" Sufficient Statistics $S_{local}^{(k)}$
Sufficient Statistics
Predicted Activity Labels (Estimate from Pervious Learned Model)
User Responded Activity Labels (Active Learning)
Activity labels
Probabilities of "Local" Model $P_{local}^{(k)}$
Probabilities of "Global" Model $P_{global}^{(k)}$
Parameter Estimation
Feature Selection (Model Construction)
Fusing Probabilities of "Global" & "Local" Model

Fig. 4-3 The self-reconfiguring procedure with active learning (selecting hard-to-predicted instances to request the corresponding activity labels).

$$E_i(D_t^{(k)}) = \frac{-\sum_{a^i \in A^i} B^{(k)}(A_t^i = a^i)\log B^{(k)}(A_t^i = a^i)}{\log N(A^i)} + \frac{L^{(k)}(F_{unknown})}{L^{(k)}(i) + L^{(k)}(F_{unknown})} \qquad (4.9)$$

where the first term of the equation is normalized entropy of the activity belief, and $N(A^i)$ is the number of states of activity type A^i; the second term of the equation is the ratio of the number of selected features $L^{(k)}(i)$ to the number of unknown (never seen) features $L^{(k)}(F_{unknown})$. Higher value of $E_i(D_t^{(k)})$ means the data may be more uncertain. User can label the training data with priority, so that the label number can be reduced effectively.

Chapter 5

System Evaluation

We have realized the proposed activity recognition system in a home environment, and design some experiments to evaluate it. In this chapter, we will introduce the experiment environment, and evaluation metric, and finally we will present the experiment result and provide a discussion.

5.1 Experiment Environment

Fig. 5-1 is the overview of the experiment environment: NTU Attentive Home Lab, and there are some photographs of the environment as demonstrated in Fig. 5-2 . The properties of the deployed sensors are listed in Table 5-1. Note that the four cameras deployed on the corners are used to collect ground-truth data. The evaluation data set is collected in the Lab from several volunteers. One of the volunteers lives in the lab for several days, and others are visitors. In most of the time, the first volunteer is alone in the lab, and he can perform activities arbitrarily (arbitrary ordering, multitasking, interrupted, even null activities). Sometimes, there are more than one resident in the environment, and they also can perform activities arbitrarily.

Sensor Type	Value Type	Purpose
Pressure Mat	Weight on it (Positive real number)	Location
Current Sensor	Current usage (Positive real number)	Object usage
Pressure Sensor	Weight on it (Positive real number)	Location
Reed Switch	Open / Close (Binary value)	Object usage
Camera	Pictures	Only used to collect ground-truth

Table 5-1 Properties of sensors deployed in the NTU Attentive Home Lab

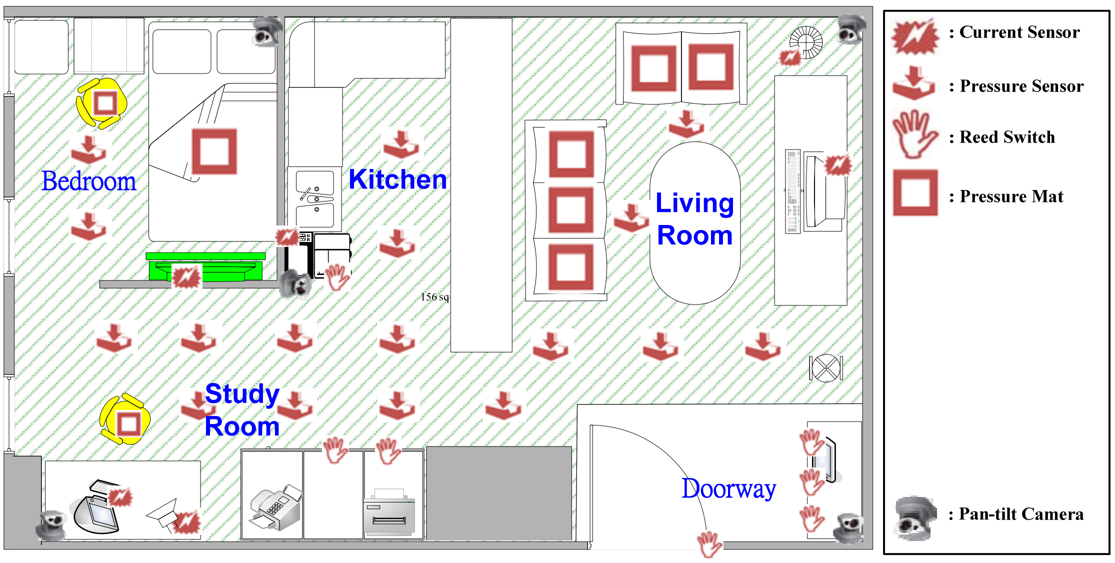


Fig. 5-1 Overview of the experiment environment: NTU Attentive Home Lab. There are two environment changes, first, we add a TV in the bedroom (painted with green), and second, we swap the chairs in the bedroom and study room (painted with yellow).

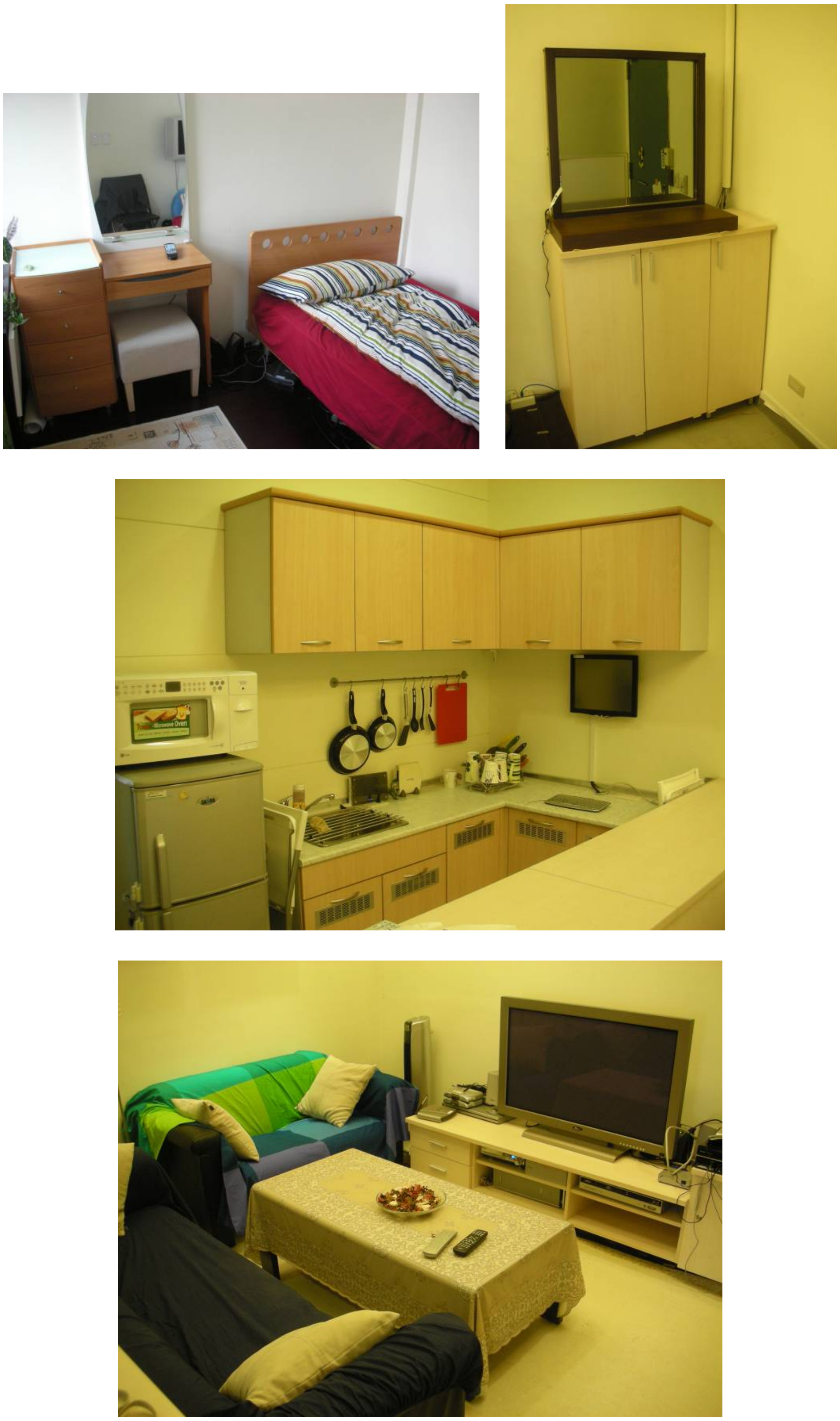


Fig. 5-2 Some photographs of the experiment environment.

In order to evaluate the abilities of the recognition system dealing with the environment changes, we design two changes in the experiment environment. As shown in Fig. 5-1, first, we add a TV in the bedroom (painted with green), and second, we swap the chair in the bedroom with the one in the study room (painted with yellow).

In our activity recognition system, the output is the states of each type of activities. Table 5-2 shows the list of nine types of activities aimed at in this experiment. At each time instant, the recognition system outputs the status of these nine types of activities. "On" means the activity is happening, "Off" means the activity is not happening, and "Unfocused" means the activity is occurring but causes the user's attention.

5.2 Evaluation Description

There are four stages in the evaluation:

（1） Initial training stage: We collected two days of training data $D^{(0)}$ to train

Activity	Status	Activity	Status
Watching TV in the Living	On / Off / Unfocused	Watching TV in the Bedroom	On / Off / Unfocused
Cleaning	On / Off	Studying	On / Off
Preparing Food	On / Off	Working on PC	On / Off / Unfocused
Go out	On / Off	Come back	On / Off
Sleeping	On / Off	Take a Drink	On / Off

Table 5-2 The list of types of activities aimed in this experiment. "On" means the activity is happening, "Off" means the activity is not happening, and "Unfocused" means the activity is occurring but cause the user's attention.

the initial activity model $M^{(0)}$.

（2） Before environment changes: We collected two days of testing data $D^{(1)}$, which the environment is the same as in the initial training stage, to evaluate the performance of $M^{(0)}$.

（3） After environment changes: We collected 1.5 days of testing data $D^{(2)}$, which the environment is changed, to evaluate the performance of $M^{(0)}$.

（4） After reconfiguring the activity model: The system reconfigured the initial activity model $M^{(0)}$ to $M^{(1)}$ via dataset $D^{(1)}$ and $D^{(2)}$. Then, we collected two days of testing data $D^{(3)}$ in the changed environment to evaluate the performance of $M^{(1)}$.

5.3 Evaluation Metric

To evaluate the result of the activity recognition is very difficult. This is because that the beginning and the end of activities are very fuzzy, so that the ground-truth values are not very definite. The observers label the same activity may have large variations. Even more, activities may occur sequentially, in parallel, alternate, and overlapping.

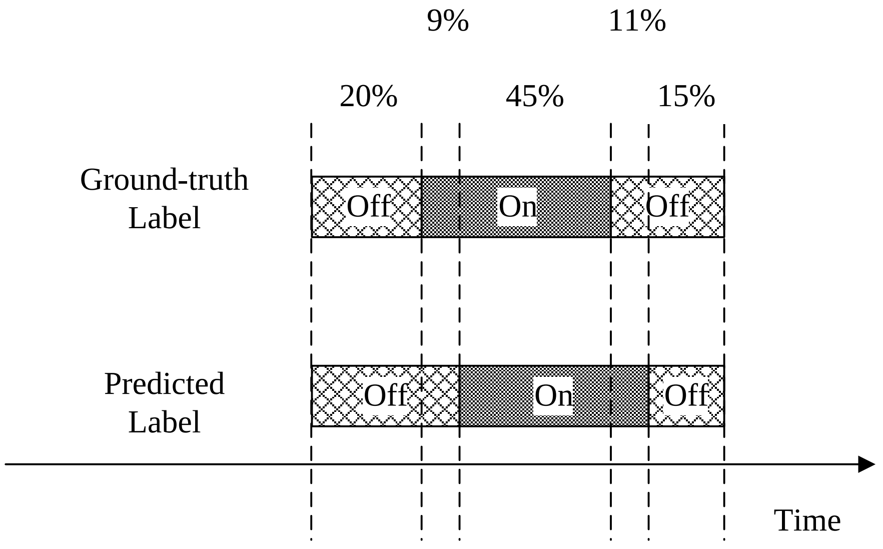
We employ two methods to evaluate the accuracy of the activity recognition system, which consider different features of the recognition system that could be important for different applications:

- **The percentages of time period that correctly classified**

This criterion measures the amount of time that the state of activity is correctly classified during the duration of the label. Fig. 5-3 shows an example and corresponding confusion matrix to exemplify the evaluation using this method: (1) 20 % of time correctly classified as “Off”; (2) 9 % of time “On” but miss classified as “Off”; (3) 45 % of time correctly classified as “On”; (4) 11 % of time “Off” but miss classified as “On”; (5) 15 % of time correctly classified as “Off”.

There are two measures for evaluating the quality of the recognition results which are defined as follows, recall:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{5.1}$$



Ground-truth / Predicted	On	Off
On	45 %	11 %
Off	9 %	20 % + 15 % = 35 %

Fig. 5-3 An example of the “The percentages of time period that correctly classified” method used to evaluate the activity recognition system. This measures the amount of time that the state of activity is correctly classified during the duration of the label.

Precision:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Postive} \tag{5.2}$$

- **How many times that correctly classify the activities**

This criterion measures the number of operations required to transform the predicted label string into the ground-truth label string (matched if the labels have overlap time period, ignore the time length, only care about counts). Fig. 5-4 shows examples of the evaluation using this method: (1) the number of operation is zero; (2) the number of operation is one, because the predicted result inserts an "Off" state among the "On" state, needs one deletion operation; (3) the number of operation is two, delete the "On" state among "Off" and insert the "On" state (because the "On" state in the predicted result and the "On" state in the ground-truth do not have overlap, they are treated as not matched).

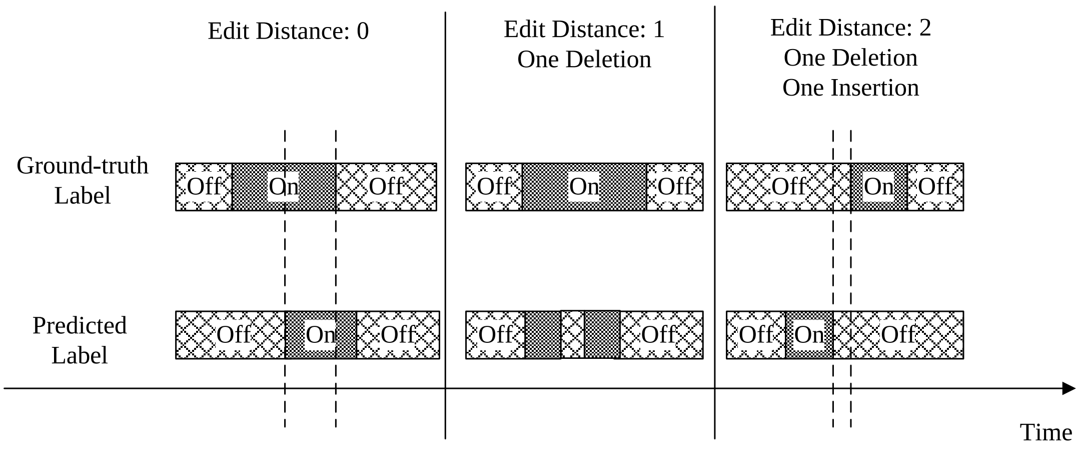


Fig. 5-4 Examples of "How many times that correctly classify the activities?" method used to evaluate the activity recognition system. This measures the number of operations required to transform the predicted label string into the ground-truth label string (matched if the labels have overlap time period, ignore the time length, only care about counts).

The selected evaluation method depends on the applications that need the recognition result. For example, "how many times that correctly classify the activities?" method is more important to applications that intended to notify elderly people when they forget to perform routine activities of daily living such as take the medicine. Conversely, a system designed to detect abnormities of activities over time may require statistics of how long activities occurred in a daily basis.

5.4 Experimental Result and Discussion

Table 5-3 shows the recognition performance before any environment changes. "Watching TV in the bedroom" activity is not available because there is no TV in the bedroom yet. We list the discussion of each type of activities in follows:

（1） Watching TV: It is difficult to distinguish "On" and "Unfocused" because it is hard to know the residents are really watching the TV or not. We do not have sensors that provide sufficient information to distinguish them clearly.

（2） Cleaning: Because the tools for cleaning are in the same cabinet, and the cabinet does not contain other things, each time between the users successively open the cabinet must be doing this type of activity, thus the recognition performance is very high.

（3） Preparing food: The bad performance of recall measure is caused by the reason that users do not continuously using tool for preparing food but recorded as preparing food activity.

（4） Go out / Come back: These two types of activities have similar interactions but with different orders. They can be distinguished by the moving direction of the residents (n-gram feature).

（5） Sleeping: The pressure sensors measure the bed is used or not by the pressure value. However, sometimes the distribution of the pressure is too unbalanced and causes the false negative detection.

（6） Studying: Sometimes the resident may stand up for stretching the body

Activity	Recall	Precision	Edit Distance
Watching TV in the Living	0.86 / 0.99 (On / Unfocused)	0.98 / 0.88 (On / Unfocused)	6 (11)
Watching TV in the Bedroom	N/A	N/A	N/A
Cleaning	1	1	0 (4)
Preparing Food	0.65	0.95	6 (2)
Go Out	1	1	0 (14)
Com Back	1	0.82	2 (14)
Sleeping	0.97	0.99	28 (6)
Studying	0.99	0.95	16 (22)
Working on PC	0.99 / 0.97 (On / Unfocused)	0.96 / 0.99 (On / Unfocused)	14 (28)

Table 5-3 The recognition performance before any environment changes. The length of the data set is about two days.

while studying for a long time, but not record in ground-truth. Thus causes some false positive detection.

（7） Working on PC: Similar as “studying” activity, thus the “On” state and the “Unfocussed” state are not very clear in ground-truth.

Some types of activities occur in very low frequency, and most of time their states are “Off” (means not happen). The lack of training data may cause the biased activity model; the lack of testing data may cause the high variance of measured performance.

Table 5-4 shows the recognition performance after environment changes: (1) adding a TV in the bedroom (painted with green in Fig. 5-1), (2) swap the chairs in the bedroom and study room (painted with yellow in Fig. 5-1). The performance of recognizing “sleeping” activity is decreased, because after adding the TV in the bedroom, lying on the bed not only means sleeping any longer. The “watching TV in the bedroom” activity is still not available because the recognition system does not have any knowledge about it. Also the performances of recognizing “working on PC” and “studying” activity are decreased after swapping the chair. Surprised, the performance of recognizing “watching TV in the living room” also decreased; this is because the resident changes his behavior: sitting on another chair for watching TV.

Table 5-5 shows the recognition performance after reconfiguration. The length of training data set for reconfiguring is about 3.5 days (2 days of the beginning testing data and 1.5 days of data recorded after environment changes). The length of testing data set is about 2 days. The result shows that the system has recovered the outdated knowledge of activity models.

Activity	Recall	Precision	Edit Distance
Watching TV in the Living	0.74 / 0.98 (On / Unfocused)	0.98 / 0.77 (On / Unfocused)	7 (17)
Watching TV in the Bedroom	0	N/A	2 (2)
Cleaning	0.20	1	2 (4)
Preparing Food	0.63	0.95	19 (7)
Go Out	0.79	1	2 (22)
Com Back	1	1	0 (22)
Sleeping	0.98	0.84	46 (4)
Studying	0.23	0.33	84 (12)
Working on PC	0.43 / 0.81 (On / Unfocused)	0.21 / 0.77 (On / Unfocused)	27 (60)

Table 5-4 The recognition performance after environment changes: (1) adding a TV in the bedroom (painted with green in Fig. 5-1), (2) swap the chairs in the bedroom and study room (painted with yellow in Fig. 5-1). The length of this data set is about 1.5 days.

Activity	Recall	Precision	Edit Distance
Watching TV in the Living	0.88 / 0.98 (On / Unfocused)	0.98 / 0.87 (On / Unfocused)	9 (21)
Watching TV in the Bedroom	0.72 / 0.97 (On / Unfocused)	0.73 / 0.86 (On / Unfocused)	4 (13)
Cleaning	0.56	1	4 (8)
Preparing Food	0.72	0.93	13 (12)
Go Out	0.81	0.87	4 (36)
Com Back	0.92	0.89	6 (36)
Sleeping	0.99	0.96	37 (8)
Studying	0.76	0.77	11 (15)
Working on PC	0.71 / 0.84 (On / Unfocused)	0.81 / 0.90 (On / Unfocused)	18 (56)

Table 5-5 The recognition performance after reconfiguration. The length of training data set for reconfiguring is about 3.5 days. The length of testing data set is about 2 days.

5.5 Fall Detection Application

In this thesis, we focused on recognizing daily activities. However, there are some types of abnormal activities such as fall also needed to be detected. Because the data amounts of abnormal activities are very rare, it is impractical to train the corresponding activity models. In this subsection, we predefine the activity model of fall based on our knowledge, and use it to detect fall accidents.

Base on the deployment of the environment (show in Fig. 5-1), we use the cameras and pressure sensors in the floor to detect fall accidents. The basic idea is:

1. The camera detects is there a person lying on the floor.
2. The pressure sensors in the floor measure the pressure values of the corresponding area that camera detected. If there is a person on the floor, the mean of the pressure values will be high; If the person is lying on the floor, the variance of the pressure values will be low (because the center of gravity is more stable when lying). Therefore, if the mean of the pressure values is high and the variance is low, then it is very possible a fall.

Fall Detection	Recall	Precision
Using camera only	0.88	0.65
Using floor only	0.94	0.92
Using camera and floor	0.92	0.93

Table 5-6 The recognition performance of the fall detection

3. Combining the result from camera and floor to decrease the false positive detection.

Table 5-6 shows the recognition performance of the fall detection. The false positive rate is decreased (or precision is increased) after combining the detecting result of camera and floor.

Chapter 6

Conclusion

6.1 Summary

This work shows that how to recognize daily activities in the home sitting via ubiquitous sensors, and how to adapt the activity model to deal with the environment changes via an active learning assisted semi-supervised learning strategy.

A flexible activity modeling approach has been proposed for making the recognition system can easily adapt to different situations. This approach flexibly incorporates various features with different level of representation power. This makes the recognition system can automatically tune the trade-off between complexity and representation power by selecting good features that best classifying the training data.

Unlike prior work that assume the environment is static, which the recognition system has to learn only once; in this thesis, we consider the situation when environment changes, making the proposed recognition system capable of self-reconfiguring to various situations. In addition, the active learning strategy help the recognition system requesting activity labels only when real need, and thus reducing the training effort for users.

Although the preliminary results were based on small datasets collected over several-days period of time under a multi-resident environment, techniques have been developed that could be applied to various environment to study human behavior.

6.2 Future Work

6.2.1 Improving Environment Sensors

- **Capable for identifying residents**

  If the sensor can distinguish the user identification who activating it, the complexity of data association problem in the multi-resident environment could be reduced. This makes the recognition system capable for dealing with the multi-resident problem and modeling the interactions between residents for recognizing more complex activities.

- **Incorporating more rich types of sensors**

  The most important information for recognizing activities are location and object usage and different types of objects may need different types of sensors to get these two kinds of information (for example, we can not use a pressure sensor or a switch sensor to detect a object is moved or not). It is important to design various types of sensors that can tape on various types of objects to sense various characteristics of these two kinds of information.

- **Making the whole sensor module more compact and in a single-component**

  In order to making the installation of the sensors easier and non-intrusive (do not influence the use of the object), it is important to making the whole sensor module in a single-component (do not have any out connected part or wires). Accelerometer sensor is a good option.

6.2.2 Reducing the Learning Effort

- **Reducing the initial training effort**

In our activity recognition system, it needs complete training data for training the initial activity model. If the recognition system rudimentary clusters the initial training data set and then requests the label of each cluster, it would greatly reduce the number of labels of the initial training data set.

- **Designing more rich types of queries**

In our activity recognition system, a request for labeling data is like “Are you sleeping during time a to time b?” or “What are you doing during time a to time b?”, it is limited. If the recognition system allows more rich types of queries appropriately, the user can label the data more flexible and closer to the ground-truth.

- **Requesting the label at the right time**

In order to avoid the interruption of user performing activities, our recognition system offline requests label rather than on-line. However, the user may not remember the answer very clear (in our system, we assume the user gives labels are correct). If the system can request the activity label at the right time such as when user just finish an activity (determine from the activity recognition result), then we can online request the activity label with less interruption.

6.2.3 Improving the Self-reconfigurable Activity Recognition System

- **Incorporating probabilities representation into interaction and feature**

In our activity modeling, we use various features with different representation power to flexibly control the model complexity. However, the drawback is making the activity model not robust to the sensor noise (because the state of feature is binary, this makes the system does not model the uncertainty of sensor noisy, only models the uncertainty of ambiguity in model level). We can incorporate probabilities representation into interaction and feature, thus modeling the uncertainty of sensor noisy.

- **Fusing more models which learned from various interval of training data**

As mentioned in Section 1.2 , one of the challenges of activity recognition is periodic variations. In our recognition system, we only fuse two kinds of periods of models, local and global (more detail in Section 4.2). If we fuse more models which learned from various types of periods of data, the system may be able to deal with the periodic variation problem.

- **Self discovering complex features**

In our recognition system, the feature generators exhaustively generate all kinds of features; however, their representation power is bounded in a threshold. If the recognition system can self discovering complex features from statistics (such as in [14]) rather than exhaustively generate, we may release this limitation.

- **Considering the noise of activity label**

In our recognition system, we assume that the user response activity labels are correct; however, in real scenario, the users may report incorrect labels. If the recognition system considers the noise of the user reported labels (such as in [27]), it will more robust in real applications.

REFERENCE

[1] H. H. Bui, Venkatesh, S., West, G, "Policy recognition in the abstract hidden Markov model," *Journal of Artificial Intelligence Research 17,* pp. 451 - 499, 2002.

[2] H. H. Bui, "A general model for online probabilistic plan recognition," in *International Joint Conferences on Artificial Intelligence*, 2003, pp. 1309-1318.

[3] T. V. Duong, T. V. Duong, H. H. Bui, D. Q. Phung, and S. A. V. S. Venkatesh, "Activity recognition and abnormality detection with the switching hidden semi-Markov model," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (CVPR'05)*, 2005, pp. 838-845 vol. 1.

[4] S. V. H. H. Bui, and G. West "Layered dynamic probabilistic networks for spatio-temporal modelling," *Intelligent Data Analysis,* vol. 3(5), pp. 339-361, 1999.

[5] D. F. L. Liao, and H. Kautz., "Learning and inferring transportation routines," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2004.

[6] D. J. Patterson, D. J. Patterson, D. Fox, H. Kautz, and M. A. P. M. Philipose, "Fine-grained activity recognition by aggregating abstract object usage," in *Proceedings of Ninth IEEE International Symposium on Wearable Computers, 2005.* , 2005, pp. 44-51.

[7] D. Wilson, "Simultaneous tracking & activity recognition (STAR) using many anonymous, binary sensors," in *Proceedings of The 3rd International Conference on Pervasive Computing (Pervasive 05)*, Munich, Germany, 2005.

[8] N. T. Nguyen, N. T. Nguyen, D. Q. Phung, S. Venkatesh, and H. A. B. H. Bui, "Learning and detecting activities from movement trajectories using the hierarchical hidden Markov model," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (CVPR'05)*, 2005, pp. 955-960 vol. 2.

[9] J. Wu, J. Wu, A. Osuntogun, T. Choudhury, M. A. P. M. Philipose, and J. M. A. R. J. M. Rehg, "A scalable approach to activity recognition based on object use," in *IEEE 11th International Conference on Computer Vision (ICCV)*, 2007, pp. 1-8.

[10] P. William, P. Matthai, B. J. A., and H. A. Kautz, "Learning large scale common sense models of everyday life," in *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, 2007, pp. 465-470.

[11] M. Perkowitz, M. Philipose, D. J. Patterson, and K. P. Fishkin, "Mining models of human activities from the web," in *Proceedings of the Thirteenth International World Wide Web Conference (WWW 2004)*, 2004, pp. 573-582.

[12] T. X. a. S. Gong., "Beyond tracking: modelling activity and understanding behaviour," *International Journal of Computer Vision (IJCV),* vol. 67(1), pp. 21-51, 2006.

[13] R. Hamid, R. Hamid, A. Johnson, S. Batta, A. A. B. A. Bobick, C. A. I. C. Isbell, and G. A. C. G. Coleman, "Detection and explanation of anomalous activities: representing activities as bags of event n-grams," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2005, pp. 1031-1038 vol. 1.

[14] R. Hamid, R. Hamid, S. Maddi, A. Bobick, and I. A. E. I. Essa, "Structure from statistics - unsupervised activity analysis using Suffix Trees," in *IEEE 11th International Conference on Computer Vision*, 2007, pp. 1-8.

[15] G. Shaogang and X. Tao, "Recognition of group activities using dynamic probabilistic networks," in *Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003, pp. 742-749 vol.2.

[16] X. Wang, X. Ma, and E. Grimson, "Unsupervised activity perception by Hierarchical Bayesian Models," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, 2007, pp. 1-8.

[17] H. Zhong, Visontai, M., Shi, J., and J. S. a. M. V. H. Zhong, "Detecting unusual activity in video," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)*. vol. 2 Washington, DC, 2004, pp. 819-826.

[18] S. Deerwester, S. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science (JASIS),* vol. 41, pp. 391-407, 1990.

[19] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research,* pp. 3:993-1022, 2003.

[20] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical dirichlet process," *Journal of the American Statistical Association,* 2006.

[21] F. B. Peter, V. d. Peter, L. M. Robert, J. D. P. Vincent, and C. L. Jenifer, "Class-based n-gram models of natural language," *Comput. Linguist.,* vol. 18, pp. 467-479, 1992.

[22] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," *American Association for Artificial Intelligence,* 2005.

[23] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, pp. 747-757, 2000.

[24] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford, "A hybrid discriminative/generative approach for modeling human activities," in *Proceedings of the Nineteenth International Joint Conference on Artificial*

Intelligence, Edinburgh, Scotland, 2005.

[25] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM,* vol. 18, pp. 613-620, 1975.

[26] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society,* vol. 39, pp. 1-38, 1977.

[27] S. Jianqiang and G. D. Thomas, "Active EM to reduce noise in activity recognition," in *Proceedings of the 12th international conference on Intelligent user interfaces* Honolulu, Hawaii, USA: ACM, 2007.

[28] A. C. David, G. Zoubin, and I. J. Michael, "Active learning with statistical models," Massachusetts Institute of Technology 1995.

[29] M. Andrew and N. Kamal, "Employing EM and Pool-Based Active Learning for Text Classification," in *Proceedings of the Fifteenth International Conference on Machine Learning*: Morgan Kaufmann Publishers Inc., 1998.

[30] M. Ion, M. Steven, and A. K. Craig, "Active + semi-supervised learning = robust multi-view learning," in *Proceedings of the Nineteenth International Conference on Machine Learning*: Morgan Kaufmann Publishers Inc., 2002.

[31] S. Tong and D. Koller., "Active learning for parameter estimation in bayesian networks," *NIPS,* pp. 647-653, 2000.

[32] K. Murphy, "Dynamic Bayesian Networks: Representation, Inference and Learning," in *Computer Science Division*. vol. Ph.D.: University of California, Berkeley, 2002.

[33] O. Pourret, P. Naim, and B. Marcot, *Bayesian Networks: A Practical Guide to Applications*. Chichester, UK: Wiley, 2008.

[34] Z. Ghahramani and M. Jordan, "Factorial hidden Markov models," *Machine Learning,* vol. 29, pp. 245–273, 1997.

[35] L. Saul and M. Jordan., "Boltzmann chains and hidden Markov models," *NIPS-7,* 1995.

[36] S. Fine, Y. Singer, and N. Tishby., "The hierarchical hidden Markov model: analysis and applications," *Machine Learning,* vol. 32, 1998.

[37] M. N. Radford and E. H. Geoffrey, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Learning in graphical models*: MIT Press, 1999, pp. 355-368.