In Proceedings of the Twentieth National Conference on Artificial Intelligence, AAAI-2005, Pittsburgh, USA, July 2005. Pages 3--8.

Multiple-Goal Recognition from Low-Level Signals

Xiaoyong Chai and **Qiang Yang**
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong, China
{carnamel, qyang}@cs.ust.hk

Abstract

Researchers and practitioners from both the artificial intelligence and pervasive computing communities have been paying increasing attention to the task of inferring users' high-level goals from low-level sensor readings. A common assumption made by most approaches is that a user either has a single goal in mind, or achieves several goals sequentially. However, in real-world environments, a user often has multiple goals that are concurrently carried out, and a single action can serve as a common step towards multiple goals. In this paper, we formulate the multiple-goal recognition problem and exemplify it in an indoor environment where an RF-based wireless network is available. We propose a goal-recognition algorithm based on a dynamic model set and show how goal models evolve over time based on pre-defined states. Experiments with real data demonstrate that our method can accurately and efficiently recognize multiple interleaving goals in a user's trace.

Introduction

With the recent developments in pervasive computing technology and mobile devices, it is increasingly more feasible to infer a user's movements and goals. Being able to perform goal recognition is critical to many novel applications. A typical example is to help people who suffer from cognitive disorders live safely and independently in their communities (Patterson *et al.* 2003). Plan-recognition systems provide the enabling technology by recognizing a user's goals ahead of time and offering advices in a timely manner.

In the past few years, probabilistic models based on either hidden Markov models (Han & Veloso 1999; Bui, Venkatesh, & West 2002; Bui 2003) and dynamic Bayesian networks (Liao, Fox, & Kautz 2004; Yin, Chai, & Yang 2004) were proposed to bridge the gap between low-level sensor readings and high-level goals. Issues on modeling, learning and inference were addressed with respect to these models. A common assumption made by these approaches is that in a trace of sensory data recording a user's activity, *a single goal* is achieved. Slightly more sophisticated systems can recognize several goals to be achieved sequentially, one after another. However, in real-world environments, a user can work towards multiple goals within a single sequence of actions. Actions for different goals can be interleaved and goals can be achieved concurrently in which a single action serves multiple goals. In these cases, many of the above models cannot be directly applied.

For illustration, suppose that actions are directly observable. Let $\mathbb{G} = \{G_0, G_1, G_2, \cdots, G_m\}$, $\mathbb{A} = \{A_1, A_2, \cdots, A_n\}$ denote the sets of goals and actions in modeling a user's behavior in an environment. $G_k \in \mathbb{G}$ denotes a possible goal (intention) of a user. In particular, G_0 is a goal which we specify to account for *default* behavior. By default, we mean the following two situations: (1) the user takes actions at will in the environment and thus his behavior is random; (2) the user takes actions in service of some other goals that are not modeled, either because these goals are unknown or because they are not of interest. $A_i \in \mathbb{A}$ is an action that the user may take. In general, a single action can be used for multiple purposes. That is, it can be a shared step towards multiple goals.

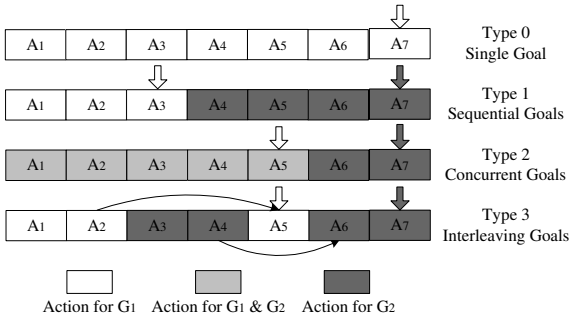

Figure 1: Types of goal composition in an action sequence

Figure 1 illustrates four representative types of goal composition in a single action sequence $< A_1, A_2, \cdots, A_7 >$. An arrow denotes a terminating action of a goal. As shown in the figure, type 0 is the most common case where a single goal G_1 is achieved by the whole action sequence. Two goals are contained in each of the other three types. Type 1 represents a sequential case, where actions for G_2 start when G_1 reaches its final step A_3. In type 2, G_1 and G_2 are pursued concurrently and they share the first five actions. G_1 is achieved by two actions, A_6A_7, ahead of G_2. Type 3 shows another special case where goal transitions between G_1 and G_2 happen frequently. More complicated types of

multiple-goal composition can be obtained from combinations of the above four types. A multiple-goal recognition problem is thus to infer one or more of the user's goals from an ongoing action sequence, and to associate actions to each recognized goal. It is ideal for a multiple-goal recognition system to recognize type 3 of the case above.

Unfortunately, many previous approaches cannot handle type-3 multiple goals. In this paper, we propose a novel algorithm to handle this situation, by inferring a user's multiple high-level goals from low-level sensory data. In our approach, we establish a dynamic model set in which models are instantiated and terminated dynamically. Each model is a finite state machine and functions as a goal recognizer. Multiple-goal behavior is modelled as transitions among some pre-defined states of these models. By distinguishing the state of a model, we can infer whether one of a user's goals is present or not. In our method, we also address the issue of goal abandonment where a user gives up his goals.

The main contribution of this paper is that we identify and solve all four types of multiple-goal recognition problem from low-level signals received from sensors in a wireless environment. A novel method is proposed for inferring a user's multiple goals within a single observed trace. In addition, we demonstrate the effectiveness and efficiency of our method through empirical studies using real data that we have collected in such an environment.

Related Work

Recognizing a user's high-level behavior has traditionally been the focus of plan recognition in the area of artificial intelligence (Kautz & Allen 1986; Charniak & Goldman 1993; Goldman, Geib, & Miller 1999; Blaylock & Allen 2003). However, most of the work was restricted to high-level inference, whereas the challenge of dealing with low-level sensory data has not been addressed.

In recent years, there has been an increasing interest in inferring a user's activities through integrating both high-level behavior and low-level sensor modeling. (Bui, Venkatesh, & West 2002; Bui 2003) introduced abstract hidden Markov models to represent the hierarchical structure of a person's activity in an indoor environment and predict his goal from camera data. In (Liao, Fox, & Kautz 2004), the authors applied dynamic Bayesian networks to estimate a person's locations and transportation modes based on logs of GPS measurements. (Yin, Chai, & Yang 2004) proposed a two-level architecture of Bayesian models to infer a user's goal in a complex indoor wireless environment.

For all these approaches, it is assumed that a user has a single goal (e.g., "going to the work place") in action execution or switch to a next goal (e.g., "returning home") only when the previous one is achieved. Thus, while these models can handle types 0 and 1 in Figure 1, they are inherently incapable of modeling a user's multiple, concurrent or interleaving goals (type 2 or 3). This is because in these models, goals "compete" with each other to explain actions. In other words, although these Bayesian methods can exploit the phenomenon of "explaining away", they cannot distinguish between co-existence and non-existence of multiple goals. For example, these models cannot distinguish the case where evidence for discriminating two goals is insufficient (e.g., each goal has a posterior probability of 1/2), from the case where the two goals are actually being pursued concurrently. In our work, we aim to provide a general framework in which both single-goal and multiple-goal recognition can be modeled. Thus, we have achieved a major advance over previous work.

In (Han & Veloso 1999), the authors presented an HMM-based algorithm that performs automated robot-behavior recognition in robotic soccer. In their work, an augmented version of HMM called behavior HMM (BHMM) is introduced to represent behavior of a robot in a soccer game. A BHMM can recognize a single execution of a behavior, provided it is instantiated around the time when the real behavior starts. BHMMs are thus repeatedly created at regular intervals to meet the starting time. In this scheme, although the single-goal assumption in action execution is relaxed for modeling types $0\sim2$ in Figure 1, their algorithm is still insufficient to handle the cases where goals are achieved in an interleaving or more complex manner.

Our method, inspired by the BHMM approach in building and applying recognition models dynamically, goes beyond their work . We introduce a *suspending state* that a model can enter. We also provide a novel mechanism to manage state transitions for these models. As a result, all four types of multiple-goal composition can be handled. Moreover, in our approach, goal-recognition models are created economically (only when they are needed). Thus, our algorithm improves both the capability and efficiency in inference.

Overview of a Wireless Environment

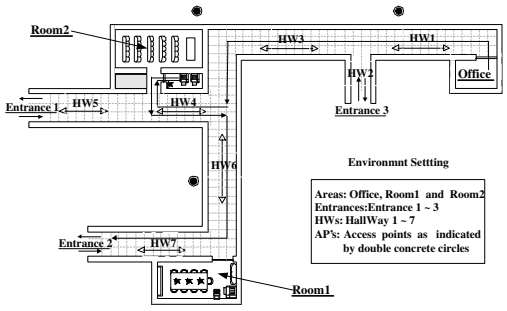

Figure 2: Layout of an office area

A user with a mobile device performs actions, such as *Walk-in-HW2*, *Enter-Office* and *Print*, in the environment (Figure 2). The mobile device periodically (e.g. per second) records strength measurements of signals propagated from the APs. For example, $o =< 48, 83, 57 >$ denotes an observation consisting of strength values of signals from the three APs. We define a user's behavior as a sequence of actions taken to achieve high-level goals (e.g., *"Seminar-in-Room2"* and *"Exit-through-Entrance1"*). A user's behavior trace is represented as a sequence of signal-strength measurements $< o_1, o_2, \cdots, o_t >$ recording his movements.

Proposed Method

In this section, we first briefly introduce the two-level architecture that we adopt to model a user's multiple-goal be-

havior in a wireless environment. Then, we give a detailed description of our proposed recognition method.

Two-level Architecture in Behavior Modeling

The architecture, similar to that used in (Yin, Chai, & Yang 2004), is shown in Figure 3. It consists of two levels. The lower level starts from the sensor layer to the action layer. A dynamic Bayesian network (Murphy 2002) is applied to estimate a user's actions from traces of signal-strength measurements. An estimated action sequence is then passed from the lower level to the upper level. On the upper level is a set of goal models that are created and terminated dynamically. Each model is a finite state machine that corresponds to one of a user's goal. We model a user's behavior as transitions in these finite state machines rather than as competitions among them. Taking the action sequence as input, each model reports whether a goal is present or not. By this means, we enable multiple-goal recognition. The advantage of this two-level representation is that it treats uncertainty in low-level signal dynamics and uncertainty in high-level user-behavior dynamics separately.

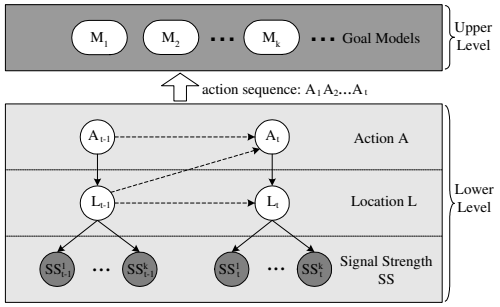

Figure 3: Multiple-goal recognition architecture

Since our main contributions are the design and implementation of the upper level, we omit detailed discussion of the lower level in this paper. Interested readers please refer to (Yin, Chai, & Yang 2004).

Goal Recognition with a Dynamic Model Set

Definitions For each goal $G_k \in \mathbb{G}$, we define a *goal model* M_k as a triple $< \pi_k, Q_k, T_k >$. Here, $\pi_k = P(G_k)$ and $Q_k = \{P(A_i|G_k), A_i \in \mathbb{A}\}$ are the goal and action priors. $T_k = \{P(A_i|A_j, G_k), A_{i,j} \in \mathbb{A})\}$ specifies the action-transition probabilities. $Q_k(A_i) \triangleq P(A_i|G_k)$, and $T_k(A_j, A_i) \triangleq P(A_i|A_j, G_k)$. We call G_0 a *default* goal and others ($G_1 \sim G_m$) *special* goals. Correspondingly, M_0 is a *default*-goal model and others ($M_1 \sim M_m$) are *special*-goal models. Intuitively, we allow a default model to capture all background behavior of the user and keep this model running along the user trace. When the model of a goal G_k, $k \neq 0$, has a higher likelihood score than its corresponding default model, we can infer that this goal is currently being pursued.

To track an agent's movement, we keep a model set $\mathbb{M}$ defined as a set of models $\{M_k^t, 0 \leq k \leq m\}$, where $M_k^t, k \neq 0$ denotes a goal model for G_k created at time t. In addition, we define $\mathbb{S} = \{s_r, s_p, s_t\}$ as the set of states that each goal model can be in, namely:

- **Running State (s_r)** – It indicates that a model is at work and its recognizing goal is actively being pursued. It is also the state which a new model enters after being initialized. Actions processed by a model when it is in state s_r are said to be *accepted* by the model. We use $Acc(M)$ to denote the latest accepted action of the model M; this variable serves as a memory to remember the last action taken towards a goal when the goal is running, or before a goal is temporarily suspended.
- **Suspending State (s_p)** – The state in which a model stops responding to input actions while the model still remains in $\mathbb{M}$. These input actions are said to be *not accepted* by the model and we use $N_{ms}(M)$ to denote the number of actions that are not accepted by the model M since its latest suspension. A model enters s_p when input actions do not meet the model's expectations and the model leaves s_p under certain conditions. As an example, a professor might stop working in office temporarily and go to have a cup of coffee for a while, and then resuming his work.
- **Terminal State (s_t)** – A model is removed from $\mathbb{M}$ upon entering s_t, which denotes a goal's completion. We define $\widehat{\mathbb{A}}_k$ as a set of actions with which goal G_k can terminate. $\widehat{\mathbb{A}}_k$ can be obtained from the history of a user's action execution for G_k in the training data.

The state diagram is shown in Figure 4. Transitions between states are specified below.

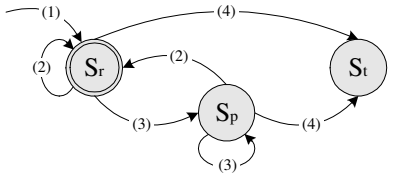

Figure 4: State diagram: (1) instantiate; (2) evolve; (3) suspend; (4) terminate.

Model Instantiation Goal models are instantiated when the model set $\mathbb{M}$ is empty or all the special-goal models in $\mathbb{M}$ are in state s_p. Suppose that the estimated action at the current time t is A_t. To determine whether a special-goal model M_k ($k \neq 0$) should be instantiated or not, its likelihood score $\pi_k Q_k(A_t)$ is compared with that of the default model $\pi_0 Q_0(A_t)$. If $\pi_k Q_k(A_t) \geq \pi_0 Q_0(A_t)$ and no instance of M_k for G_k is in $\mathbb{M}$, a new model M_k^t is instantiated. Otherwise, it is not instantiated. Upon instantiation, its likelihood at time t is initialized as $L_t(M_k^t) = \pi_k Q_k(A_t)$ and its state $S_t(M_k^t) = s_r$. By definition, $Acc(M_k^t) = A_t$. M_k^t is subsequently added into $\mathbb{M}$. If at least one special-goal model is created at time t, a default-goal model M_0^t is also instantiated to track a user's actions. $L_t(M_0^t) = \pi_0 Q_0(A_t)$ and M_0^t are also added into $\mathbb{M}$. The purpose of instantiating a default-goal model along with others is to distinguish a user's *special-goal–oriented* behavior from his *default* behavior, where different default behaviors have different likelihood depending on when it is started. Once instantiated, M_0^t stays in state s_r until terminated.

Model Update Let $M_k^{t_0}$ ($k \neq 0$) denote a special-goal model initialized at time t_0. $M_0^{t_0}$ is thus the default-goal model also instantiated at t_0. Let A_t and A_{t-1} be the actions at current time t and previous time $t-1$, respectively. Models are updated *iff* $A_t \neq A_{t-1}$. Suppose that $A_t \neq A_{t-1}$ in the following discussion. Updating $M_k^{t_0}$ includes updating its state, likelihood and latest accepted action. A detailed description is given in Algorithm 1 and we explain it through the two cases below. $M_0^{t_0}$ remains in state s_r and its likelihood is updated as follows:

$$L_t(M_0^{t_0}) = L_{t-1}(M_0^{t_0}) \times T_0(A_{t-1}, A_t). \quad (1)$$

Algorithm 1 Goal-Model Update (of a special model $M_k^{t_0}$)

Input: $L_{t-1}^{k,t_0} \triangleq L_{t-1}(M_k^{t_0})$ – the likelihood of $M_k^{t_0}$ at time $t-1$, $Acc^{k,t_0} \triangleq Acc(M_k^{t_0})$ – the latest accepted action;
Output: L_t^{k,t_0}, S_t^{k,t_0}, Acc^{k,t_0};
Procedure:

1: Update the model state:
  if $T_k(Acc^{k,t_0}, A_t) < T_0(A_{t-1}, A_t)$ **then**
    $S_t^{k,t_0} \leftarrow s_p$; // enter the suspending state
  else
    if $A_t \in \widehat{\mathbb{A}}_k$ **then**
      $S_t^{k,t_0} \leftarrow s_t$; // enter the terminal state
    else
      $S_t^{k,t_0} \leftarrow s_r$; // enter the running state
    end if
  end if
2: Update the model likelihood:

$$L_t^{k,t_0} = \begin{cases} L_{t-1}^{k,t_0} \times T_k(Acc^{k,t_0}, A_t) & \text{if } S_t^{k,t_0} \in \{s_r, s_t\}, \\ L_{t-1}^{k,t_0} \times T_0(A_{t-1}, A_t) & \text{otherwise;} \end{cases} \quad (2)$$

3: Update the latest accepted action:

$$Acc^{k,t_0} = \begin{cases} A_t & \text{if } S_t^{k,t_0} \in \{s_r, s_t\}, \\ Acc^{k,t_0} & \text{otherwise.} \end{cases} \quad (3)$$

Case 1: Model Suspension As given in Algorithm 1, model $M_k^{t_0}$ ($k \neq 0$) is suspended if the condition:

$$T_k(Acc^{k,t_0}, A_t) < T_0(A_{t-1}, A_t), \quad (4)$$

holds. Here, goal-oriented behavior (T_k) and default behavior (T_0) are compared. This condition means that following the latest accepted action Acc^{k,t_0} and accepting the current action A_t lowers the likelihood of this special-goal model, while the likelihood of the default-goal model increases. In other words, a user less probably continues his plan for goal G_k than exhibits default behavior by taking action A_t. The criterion of Equation 4 is reasonable in that when a user follows his plan, the possibility of his goal-oriented behavior G_k increases. On the other hand, when he deviates from his plan, default behavior becomes more likely instead.

Once suspended, $M_k^{t_0}$ stays in s_p and its likelihood is updated by Equation 2 to keep pace with the update of $M_0^{t_0}$ in Equation 1. The idea is that by suspending $M_k^{t_0}$, we maintain the distance between the likelihood of $M_k^{t_0}$ and that of $M_0^{t_0}$, rather than change it. In addition, its latest accepted action Acc^{k,t_0} remains the same, as given in Equation 3.

Case 2: Model Evolvement and Termination When the condition of Equation 4 fails, A_t is more likely an action towards goal G_k than an action of default behavior. Accordingly, the model likelihood and the last accepted action are updated by Equations 2 and 3. If $A_t \in \widehat{\mathbb{A}}_k$, the terminating-action set of G_k, model $M_k^{t_0}$ enters state s_t. Otherwise, it enters state s_r. When the model enters s_t, goal G_k is reported to have been achieved. G_k can also be predicted earlier when the model is still in s_r, depending on different applications. Furthermore, we can obtain those actions for G_k in the current trace by keeping track of Acc^{k,t_0} update. Upon termination, $M_k^{t_0}$ is removed from the model set $\mathbb{M}$.

Complete Multi-Goal Recognition Algorithm

Algorithm 2 Multi-Goal Recognition with a Dynamic Model Set

Procedure:

1: Initially, $t = 0$ and $\mathbb{M} = \{\}$;
2: **while** a user's activity is in progress **do**
3: Estimate the current action A_t from the signal-strength measurements with the lower action model in Figure 3;
4: **if** $A_t == A_{t-1}$ **then**
5: Continue;
6: **end if**
7: **for** each model $M_k^{t_0} \in \mathbb{M}$ $(0 \leq k \leq m, 0 \leq t_0 < t)$ **do**
8: Update the model by Equation 1 (if $k = 0$) or Algorithm 1 (if $k \neq 0$);
9: **if** $S_t(M_k^{t_0}) == s_t$ **then**
10: Report G_k as a recognized goal which starts at t_0;
11: $\mathbb{M} \leftarrow \mathbb{M} - \{M_k^{t_0}\}$;
12: **end if**
13: **if** $S_t(M_k^{t_0}) == s_p$ **and** $N_{ms}(M_k^{t_0}) > N_m$ **then**
14: $\mathbb{M} \leftarrow \mathbb{M} - \{M_k^{t_0}\}$; // timeout, discard $M_k^{t_0}$ from $\mathbb{M}$
15: **end if**
16: **end for**
17: **for** $0 \leq t_0 < t$ **do**
18: **if** $\nexists k \in \{1, \ldots, m\}$ such that $M_k^{t_0} \in \mathbb{M}$ **then**
19: $\mathbb{M} \leftarrow \mathbb{M} - \{M_0^{t_0}\}$; // remove the default-goal model
20: **end if**
21: **end for**
22: **if** $\mathbb{M} = \{\}$ **or** $S_t(M_k^{t_0}) = s_p, \forall M_k^{t_0} \in \mathbb{M}, 1 \leq k \leq m$ **then**
23: Instantiate a subset of models $\mathbb{M}_t = \{M_k^t\}$;
24: $\mathbb{M} \leftarrow \mathbb{M} \cup \mathbb{M}_t$;
25: **end if**
26: **end while**

The pseudo code of the complete algorithm is presented in Algorithm 2. To recognize goal abandonment, we adopt a timeout threshold N_m (Line 13). N_m specifies the maximum number of actions that a goal model is allowed to skip in state s_p. N_m can be given by domain knowledge. This threshold is used under the assumption that a user will not delay for more than a certain number of actions to achieve a goal. If $N_{ms}(M_k^{t_0}) > N_m$, the user is assumed to have abandoned G_k which starts at t_0 and the model can be thus removed from $\mathbb{M}$. The default-goal model $M_0^{t_0}$ is terminated and removed from $\mathbb{M}$ if all the special-goal models with $k \neq 0$ instantiated at the same time (i.e., time t_0) are terminated (Lines 18∼20).

Experimental Results

We conducted experiments in the office area shown in Figure 2 to evaluate our proposed method. Eight *special* goals of a professor's activity, such as *"Seminar-in-Room1"*, *"Exit-through-Entrance2"* and *"Return-to-Office"*, are modeled. In addition, a *default* goal G_0 is added to account for his default behavior. We collected 850 single-goal traces using the device driver and API we have developed. We synthesized multiple-goal traces from the single-goal ones by simulating a professor's activities in this environment: Segments were pieced together to generate connective traces containing multiple goals. In total, we obtained 750 two-goal traces and 300 three-goal traces that are of goal types 1 to 3.

We compare our algorithm, referred to as ***MG-Recognizer***, with the N-gram-based recognition algorithm (Yin, Chai, & Yang 2004), referred to as ***SG-Recognizer*** and the BHMM-based recognition algorithm (Han & Veloso 1999), referred to as ***BHMM-Recognizer***. *SG-Recognizer* uses an N-gram model to infer the most probable goal of a user from his action sequence. The N-gram model is equivalent to a set of m goal models that are defined in our work. Here, m is the total number of goals and $m = 9$ in our experiments. Since this algorithm was designed also in a wireless domain, we selected it as a representative of single-goal recognition algorithms. In contrast, the *BHMM-Recognizer* is capable of recognizing multiple goals in a single trace by instantiating multiple copies of models at different times, where models are implemented as HMMs that keep running until terminated. Each model reports whether its recognizing goal is present or not. In our experiments, goal-instantiation is activated once every input action. For fairness, all recognizers receive input action sequences from the same low-level action model in Figure 3. Two criteria were used for evaluation:

1. *Recognition Accuracy:* It measures how accurate an algorithm is at recognizing the set of goals in a user's trace, where a trace contains a single or multiple goals.
2. *Inference Efficiency:* Since the three algorithms perform inference using a different number of goal models and each model is computationally cheap, efficiency is measured in terms of the number of models instantiated.

Finally, we used three-fold cross validation. Both action and goal models were trained from the training traces.

A Recognition Example

We use an example to illustrate how these three algorithms perform recognition. In the area shown in Figure 2, a professor started from his office and walked through hallways (HW1, HW3 and HW4) to get some printed material from the printer in Room2. Then, he turned back (HW4) and exited the office area through Entrance2 (through HW6 and HW7). The whole action trace is <*Walk-in-HW1*, *Walk-in-HW3*, *Walk-in-HW4*, *Print*, *Walk-in-HW4*, *Walk-in-HW6*, *Walk-in-HW7*> and this single trace contains two goals, G_1=*"Print-in-Room2"* and G_2=*"Exit-through-Entrance2"*. G_1 and G_2 shared the first two actions (*Walk-in-HW1* and *Walk-in-HW3*). After these two actions, G_1 and G_2 were achieved separately (actions *Walk-in-HW4* and *Print* for G_1; actions *Walk-in-HW6* and *Walk-in-HW7* for G_2).

Figure 5 shows the recognition processes of the three algorithms. For illustration, only a set of three goals $\{G_0, G_1, G_2\}$ was considered. Thus, in each algorithm, three goal models (M_k for $G_k, k = 0, 1, 2$) were instantiated at the starting time. The performance of *SG-Recognizer* is shown in Figure 5(a). As we can see, based on the single-goal assumption, each model competed with one another to explain the whole action trace. As a result, *SG-Recognizer* missed both G_1 and G_2 and recognized the whole trace as the default goal G_0.

In contrast, *BHMM-Recognizer* and *MG-Recognizer* avoid competition among goal models by assuming that multiple goals can coexist. For illustration, in Figure 5(b), we normalized the likelihood of M_1 (M_2) with that of M_0. For M_1 at time t, the normalized likelihood is computed as: $\hat{L}_t(M_1) = \frac{L_t(M_1)}{L_t(M_1)+L_t(M_0)}$. Therefore, when $M_1(M_2)$ terminates, if its likelihood value is greater than $1/2$, we say that $G_1(G_2)$ rather than G_0 is present. As shown in the upper graph of Figure 5(b), *BHMM-Recognizer* was capable of recognizing G_1 at its terminating action *Print*. However, it missed G_2 at the action *Walk-in-HW7* and reported G_0 instead, as shown in the lower graph of the figure. *MG-Recognizer* caught both goals through distinguishing the models' states. For example, as shown in the lower graph of Figure 5(c), M_2 was suspended when the recognizer judged that the user had deviated from his plan for G_2 at the action *Walk-in-HW4*, and M_2 was resumed later at the action *Walk-in-HW6*. The accompanied model-state sequences are also shown in the figure.

Figure 6 compares the three algorithms in terms of the number of goal models instantiated over time. Again, for simplicity, only the three goals (G_0, G_1 and G_2) were involved. *SG-Recognizer* maintained a set of three models during the whole recognition process. Since *BHMM-Recognizer* instantiated a set of models every action, the number increased linearly on the whole. In contrast, *MG-Recognizer* was much more efficient, requiring no more than four models. This is because the suspending state enlarges the expression power of a goal model. Also, goal models are instantiated selectively instead of collectively, compared with *BHMM-Recognizer*.

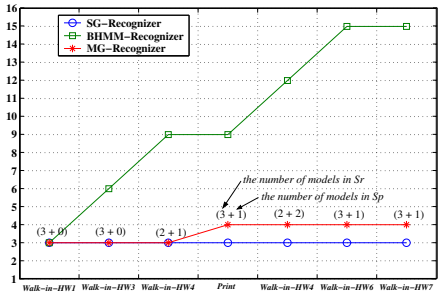

Figure 6: Comparison of the number of goal models

Overall Evaluation

Finally, recognition performance was measured over a total of 850 single-goal traces and 1050 multiple-goal traces us-

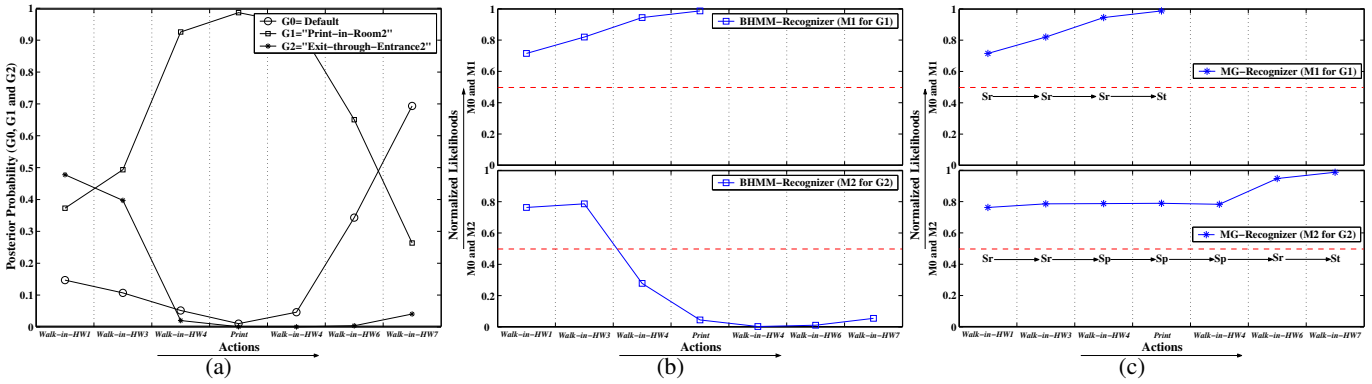

Figure 5: Comparison of the three algorithms in recognizing G_1 and G_2: (a) *SG-Recognizer*; (b) *BHMM-Recognizer* (M_1 for G_1 and M_2 for G_2); (c) *MG-Recognizer* (M_1 for G_1 and M_2 for G_2), together with model-state sequences.

ing a three-fold cross validation. The whole set of nine goals (eight special goals and one default goal that start at time zero) was considered.

Table 1 shows the comparison in recognition accuracy for both single and multiple-goal recognition. *MG-Recognizer* performs the best in multiple-goal recognition: 12.3% higher than *BHMM-Recognizer* and 66.9% higher than *SG-Recognizer*. In single-goal recognition, *MG-Recognizer* is only 3.2% lower than *SG-Recognizer*. Thus, our method offers a solution for general goal recognition.

Table 2 compares the average number of goal models involved during recognition. As we can see, *MG-Recognizer* is much more economical than *BHMM-Recognizer* in both types of recognition: On average, *MG-Recognizer* maintains only about 6.6 running models (in state s_r) and about 4.3 suspending models (in state s_p). Furthermore, compared with *SG-Recognizer*, which requires a minimum of nine models, *MG-Recognizer* incurs only slightly more overhead and requires even less computation if we considered the number of active goals only (in state s_r).

Recognizer	SG-	BHMM-	MG-
Single-Goal	97.8%	95.5%	94.6%
Multiple-Goal	24.5%	79.1%	91.4%

Table 1: Comparison of recognition accuracy

Recognizer	SG-	BHMM-	MG-
Single-Goal	9	20.7	6.5 + 3.7
Multiple-Goal	9	28.7	6.6 + 4.8

Table 2: Comparison of recognition efficiency

Conclusions

In this paper, we proposed a solution to the problem of inferring a user's high-level goals from low-level sensory data. We first formulated the multiple-goal recognition problem in an indoor wireless environment and then proposed a recognition algorithm using a dynamic model set. Experiments with real data demonstrated its accuracy in recognition and its efficiency in inference. In the future, we plan to extend our work in several directions. One extension is to generalize transitions between model states with probability measures. In this paper, we have assumed a deterministic transitional framework. However, we expect that probabilistic transitions will improve the robustness of the algorithm.

Acknowledgments

We would like to thank the support of the Hong Kong ITC Fund ITS/110/02 and the support of Hong Kong University of Science and Technology and a grant from RGC (HKUST 6187/04E).

References

Blaylock, N., and Allen, J. 2003. Corpus-based, statistical goal recognition. In *Proc. of the 18th International Joint Conference on Artificial Intelligence*.

Bui, H. H.; Venkatesh, S.; and West, G. 2002. Policy recognition in the abstract hidden markov model. *Journal of Artificial Intelligence Research* 17.

Bui, H. H. 2003. A general model for online probabilistic plan recognition. In *Proc. of the 18th International Joint Conference on Artificial Intelligence*.

Charniak, E., and Goldman, R. P. 1993. A Bayesian model of plan recognition. *Artificial Intelligence* 64.

Goldman, R. P.; Geib, C. W.; and Miller, C. A. 1999. A new model of plan recognition. In *Proc. of UAI99*.

Han, K., and Veloso, M. 1999. Automated robot behavior recognition applied to robotic soccer. In *Proc. of IJCAI-99 Workshop on Team Behaviors and Plan Recognition*.

Kautz, H., and Allen, J. 1986. Generalized plan recognition. In *Proc. of the 5th National Conference on Artificial Intelligence*.

Liao, L.; Fox, D.; and Kautz, H. 2004. Learning and inferring transportation routines. In *Proc. of the 19th National Conference on Artificial Intelligence*.

Murphy, K. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. dissertation, UC Berkeley, Computer Science Division.

Patterson, D. J.; Liao, L.; Fox, D.; and Kautz, H. 2003. Inferring high-level behavor from low-level sensors. In *Proc. of the 5th International Conference of Ubiquitous Computing*.

Yin, J.; Chai, X.; and Yang, Q. 2004. High-level goal recognition in a wireless lan. In *Proc. of the 19th National Conference on Artificial Intelligence*.