TEBRA: An Automatic Prompting System for Persons with Cognitive Disabilities in Brushing Teeth

Christian Peters, Thomas Hermann and Sven Wachsmuth
CITEC, Bielefeld University, Bielefeld, Germany

Keywords: Cognitive Assistive Technology, Task Analysis, Bayesian Network, Dynamic Timing Model, User Study.

Abstract: We introduce a novel Cognitive Assistive Technology. The TEBRA system assists persons with cognitive disabilities in brushing teeth by prompting the user. We develop the system based on an analysis of the task using qualitative data analysis. We recognize different subtasks applying a hierarchical recognition framework based on a Bayesian Network. The user's progress in the task is monitored using a Finite State Machine and dynamic timing model which allows for different velocities of task execution. We evaluate the TEBRA system in a first study with regular users. We found that the system is able to provide appropriate prompts in terms of timing and modality to assist a user through the complex task of brushing teeth.

1 INTRODUCTION

Cognitive assistive technology aims at developing systems which support persons with cognitive disabilities in the execution of activities of daily living (ADLs). Such persons mostly have problems in accomplishing ADLs on their own and need assistance to perform such tasks successfully. Automatic prompting systems can provide assistance and keep elderly people or persons with cognitive disabilities further in their own homes which leads to an increase of independence of the persons and a relief of caregiver burden.

The execution of ADLs is very complex: ADLs like washing hands, brushing teeth, or preparing meal contain several subtasks which can be combined in a flexible manner. Furthermore, the execution of subtasks differs significantly between persons based on their individual abilities. A key paradigm in the development of automatic prompting systems for complex ADLs is to deliver prompts to the user when necessary in order to foster the independence of the user. A prompt is necessary if the user forgets a step or gets stuck in task execution. Users with cognitive disabilities, but also regular users, show a huge variance in spatial and temporal execution of the task: one user may take the brush with the left hand while another user takes the right hand and performs completely different movements at different velocities. A cognitive assistive technology needs to deal with the spatial and temporal variance to deliver appropriate prompts in terms of timing and modality.

In this paper, we introduce a novel cognitive assistive technology: the TEBRA (**TE**eth **BR**ushing **A**ssistance) system which assists in *brushing teeth* as an important ADL. We develop the TEBRA system based on a systematic analysis of the task. We use Interaction Unit (IU) analysis proposed in (Ryu and Monk, 2009) as a method for qualitative data analysis. The results of IU analysis are utilized for different design decisions. Results are (1) the decomposition of the task into subtasks we aim to recognize, (2) the extraction of environmental conditions associated with subtasks and task progress, (3) preconditions and effect of subtasks.

In order to deal with the variance in spatial execution, we abstract from the recognition of specific movements by tracking objects or the user's hands. Instead, we classify subtasks based on environmental configurations in a hierarchical recognition framework. We cope with the temporal variance in task execution by using a Finite State Machine and a dynamic timing model allowing for different user velocities. We learn the timing parameters for different velocities of users (fast, medium and slow) and switch the parameters dynamically during a trial based on the velocity of the user. We choose appropriate system prompts using a search in an ordering constraint graph (OCG). An OCG models temporal relations between subtasks in terms of preconditions and effects obtained in the IU analysis.

We evaluate the first prototype of the TEBRA system in a study with regular users. Evaluating our system with regular users is feasible in a first study since regular persons show individual ways of task execution which may not coincide with the system's framework of action. The system prompts the user who in turn has to react to the prompts and adapt his/her behavior to successfully execute the task from a system's point of view. Hence, we provoke similar phenomena in terms of prompting and reaction behavior with both regular users and persons with cognitive disabilities. We consider the target group in the development of the TEBRA system because IU analysis is conducted on videos of persons with cognitive disabilities in a residential home setting. We are currently organizing a study with the target group. The aim of the study with regular users is two-fold: firstly, we evaluate the technical correctness of the system with regard to recognition of subtasks, monitoring the user's progress and timing of prompts. Secondly, we determine whether the prompts are appropriate in terms of duration and modality.

The remainder of the paper is structured as follows: section 2 gives an overview of relevant related work. In section 3, we give an overview of the TEBRA system. Section 4 describes IU analysis and the integration of the results in the system design. The main components of the TEBRA system are described in detail in section 5. Section 6 shows the results of the user study, followed by a conclusion in section 7.

2 RELATED WORK

Cognitive assistive technology (CAT) is developed for special user groups like elderly or persons with cognitive disabilities in a number of different ADLs: the COACH system (Hoey et al., 2010) assists persons with dementia in the handwashing task, ARCHIPEL (Bauchet et al., 2008) supports persons with cognitive disabilities in meal preparation. ADLs are complex tasks in terms of spatial execution which makes the recognition of behaviors challenging. Much work is done on recognizing behaviors based on movement trajectories of objects or the user's hands (Moore et al., 1999), (Nguyen et al., 2005), (Pusiol et al., 2008). However, user behaviors with similar appearance are hard to distinguish based on movement trajectories only. In this work, we classify behaviors based on environmental states of objects involved in the brushing task.

Users perform behaviors at different velocities due to individual abilities. In the TEBRA system, we explicitly model timing parameters of user behaviors. We use a Finite State Machine and a dynamic timing model to allow for different velocities of the user's movements. In the *Autominder* system (Pollack et al., 2003), persons with memory impairments are assisted in scheduling daily activities. *Autominder* models durations of events and reasons on temporal constraints to provide appropriate reminders. The PEAT system (Najjar et al., 2009) schedules user's activities by applying reactive planning to restructure the schedule when events take more time than expected.

Monitoring the user's progress in the task is a key aspect to provide appropriate prompting. The COACH system uses a Partially Observable Markov Decision Process (POMDP). A belief state models the user's abilities and monitors the progress in the task. Whenever the belief state changes significantly due to sensor observations, the new belief state is tested over a fixed period of time whether it persists.

The *Activity Compass* assists disoriented users finding a destination by using a hierarchical Markov model to track the location of the user (Patterson et al., 2004). The specification process of probabilistic models like POMDPs is very hard in terms of determining the probabilities of dependent variables in the model. In the TEBRA system, we monitor the user's progress by utilizing a set of environmental variables which we deterministically update based on the occurrence of user behaviors. We find appropriate prompts using a search procedure on an ordering constraint graph (OCG). An OCG models temporal relations between subtasks in terms of a partial ordering of subtasks. We don't model every possible way of executing the task as done in the ARCHIPEL system (Giroux et al., 2008) which uses a full hierarchical representation of the task. Instead, we use the OCG to model the constraints under which the execution of user behaviors is appropriate.

Most CAT systems are modelled using commonsense knowledge without further analyzing the task. Here, we apply a structured approach of retrieving relevant information on which we develop the TEBRA system. We use Interaction Unit (IU) analysis proposed in (Ryu and Monk, 2009) as a method for qualitative data analysis to obtain relevant information about the brushing task. IU analysis was used in a similar context in (Hoey et al., 2011) in order to facilitate the specification process of an automatic prompting system using a POMDP.

3 TEBRA OVERVIEW

Figure 1 depicts an overview of the TEBRA system. We built a washstand setup which we equipped with sensor technology. The sensor data is passed to a hi-

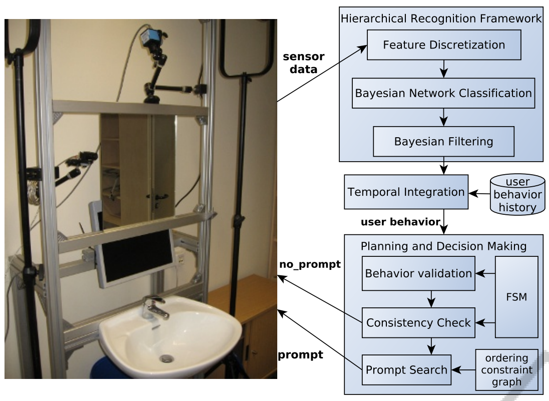

Figure 1: Overview of the TEBRA system.

erarchical recognition framework which computes the current most probable user behavior. The main problem in the recognition component is the huge spatial variance in task execution. We tackle the problem by abstracting from tracking the user's hands or objects. Instead, we infer the user's behavior based on an environmental configuration which is expressed by states of objects manipulated during a behavior: we preprocess the sensor data into discrete features representing object states which are fed into a Bayesian Network classification scheme. A Bayesian filtering step outputs a belief (conditional probability distribution) over the user behaviors.

The most probable user behavior is calculated each time new sensor data is obtained. A temporal integration mechanism accumulates the user behaviors over time and provides the duration of the behavior in seconds.

A key aspect is the huge variance in temporal execution of the task. One user, for example, is able to perform a subtask very quickly while another user takes much longer. We use a Finite State Machine (FSM) to model timing of behaviors. The states of the FSM model different phases during a user behavior: we validate a user behavior and perform a consistency check on the validated behavior with respect to the user's progress in task execution. If the validated behavior is consistent, the system won't prompt, but will instead start a new iteration cycle receiving sensor data. If the validated behavior is inconsistent, the system will search for a consistent behavior to prompt: we use an online search procedure in an ordering constraint graph. An ordering constraint graph models partial orderings between user behaviors in the brushing task and is computed offline using a partial order planner. The consistent behavior found is displayed to the user by an audio-visual prompt delivered via a TFT display at the setup.

Most CAT systems are designed using common-sense knowledge. In the design of the TEBRA system, we focus on an iterative design process using a method of qualitative data analysis which is described in the following section.

4 DESIGN PROCESS

Developing a CAT system for an everyday task like brushing teeth is a challenging problem. The task consists of several subtasks which can be combined in a flexible way during task execution. The analysis of the task and subtasks as well as the possible ways of task execution are important steps in the development of an assistance system. We apply Interaction Unit (IU) analysis proposed in (Ryu and Monk, 2009). IU analysis models interaction by describing the conjunction of cognitive and environmental pre- and postconditions for individual actions. We apply IU analysis on 23 videos recorded at our cooperation partner *Haus Bersaba*, a residential home where persons with cognitive disabilities such as Alzheimer's Disease, Autistic Spectrum Disorder, Learning Disabilities, etc. permanently live. Each video shows a single trial of a user brushing teeth while being observed and supported by a caregiver.

We are interested in three aspects of IU analysis: (1) decomposition of the task into subtasks which we will call *user behaviors* in the following, (2) environmental conditions associated with user behaviors and task progress, (3) preconditions and effects of behaviors. Table 1 shows the results of the IU analysis for brushing teeth. The brushing task is decomposed into seven user behaviors as described in column *UB*: *paste_on_brush*, *fill_mug*, *rinse_mouth*, *brush_teeth*, *clean_mug*, *clean_brush* and *use_towel*. Each user behavior is further subdivided into single steps described in column *UB steps*. *rinse_mouth* for example consists of three steps: mug is moved to the face, the user rinses the mouth and moves the mug away from the face. Column *Current Environment* shows the environmental configuration as a precondition of single user behavior steps. Performing the step changes the environmental state, for example in the first step of *paste_on_brush*: the toothpaste tube is on the counter and taking the tube changes the toothpaste location to 'in hand'.

We utilize the environmental configuration obtained in IU analysis. We extract environmental states in terms of discrete variables as depicted in Table 2. We distinguish between *behavior* and *progress* variables: we apply *behavior* variables to recognize user behaviors in a hierarchical recognition framework. The *progress* variables are hard to observe directly due to reasons of robustness: for example, it is very error-prone to visually detect whether the *brush_condition* is dirty or clean. A specialized sensor at the brushing head is not desirable due to hygienic reasons. We utilize *progress* variables to monitor the user's progress during the task.

We abstract from the recognition of single behavior steps as given in column *UB steps* in Table 1. Instead, we infer the user's behavior based on the behavioral variables which express states of objects manipulated during a behavior. From column *Current Environment*, we extract five *behavior* variables describing important object states: *mug_position*, *towel_position*, *paste_movement*, *brush_movement* and *tap_condition*. The upper part of Table 2 shows the five variables and their according discrete values. For *brush_movement*, we have the states no, yes_sink and yes_face. The latter ones are important to discriminate between the user behaviors *paste_on_brush* and *brush_teeth* based on the movement of the brush. The values of the variables *mug_position* and *towel_position* are the different regions identified in column *Current Environment* where the mug and towel appear during task execution. *No_hyp* is used if no hypothesis about the mug/towel position is available.

Table 1: Results of the IU analysis for brushing teeth. Column "UB" describes the different subtasks involved in the brushing task. Column "UB steps" lists the ideal steps to execute the according subtask. Column "Current Environment" shows the environmental configuration in terms of states of objects involved in a particular step. TT = toothpaste tube.

UB	Current Environment	UB steps
paste_on_brush	TT on counter	take TT from counter
	TT closed in hand	alter TT to open
	brush on counter	take brush from counter
	brush and TT in hand	spread paste on brush
	TT is open	alter TT to closed
	TT closed in hand	give TT to counter
	TT on counter, brush in hand	
fill_mug	mug empty	give mug to tap
	mug at tap, tap off	alter tap to on
	mug at tap, tap on	alter tap to off
	mug filled	
rinse_mouth	mug filled	give mug to face
	mug at face	rinse
	mug else	give mug to counter
	mug counter	
brush_teeth	brush with paste in hand	give brush to face
	brush at face	brush all teeth
	brush at face, teeth clean	take brush from face
	brush not at face	
clean_mug	mug dirty at counter	give mug to tap
	mug dirty at tap, tap off	alter tap to on
	mug dirty at tap, tap on	clean mug
	mug clean at tap, tap on	alter tap to off
	mug clean at tap, tap off	give mug to counter
	mug clean at counter	
clean_brush	brush dirty	give brush to tap
	brush dirty at tap, tap off	alter tap to on
	brush dirty at tap, tap on	clean brush
	brush clean at tap, tap on	alter tap to off
	brush clean at tap, tap off	give brush to counter
	brush clean at counter	
use_towel	towel at hook, mouth wet	give towel to face
	towel at face, mouth wet	dry mouth
	towel at face, mouth dry	give towel to hook
	towel at hook	

Table 2: *Behavior* and *progress* variables extracted from the environmental configuration in Table 1.

Type	State Variable	Values
behavior	mug_position	counter, tap, face, else, no_hyp
	towel_position	hook, face, else, no_hyp
	paste_movement	no, yes
	brush_movement	no, yes_sink, yes_face
	tap_condition	off, on
progress	mug_content	empty, water
	mug_condition	dirty, clean
	mouth_condition	dry, wet, foam
	brush_content	no_paste, paste
	brush_condition	dirty, clean
	teeth_condition	dirty, clean

We utilize *progress* variables to monitor the user's progress in the task. At each time in task execution, the user's progress is modelled by the set of six *progress* variables which we will denote *progress state space* in the following. The lower part of Table 2 shows the variables of the progress state space and their according discrete values.

The occurrence of a user behavior during the execution of the task leads to an update of the progress state space: we define necessary preconditions and effects of user behaviors in terms of *progress* variables. When a user behavior occurs, we check whether the preconditions are met and, if so, update the progress state space with the effects of the current behavior. Table 3 shows the preconditions and effects for user behaviors in terms of *progress* variables extracted during IU analysis. We distinguish between *rinse_mouth_wet* and *rinse_mouth_clean* because the steps have different meanings during task execution: the IU analysis is based on videos recorded at *Haus Bersaba*. The videos showed that wetting the mouth with water using the mug (before brushing the teeth) is a common step as part of the regular daily routine. This step is described as *rinse_mouth_wet*

Table 3: Preconditions and effects of user behaviors extracted from the environmental configuration in Table 1.

User Behavior	Preconditions	Effects

paste_on_brush	brush_content=no_paste
	teeth_condition=dirty
		brush_content=paste
		brush_condition=dirty

fill_mug	mug_content=empty	mug_content=water

clean_mug	mug_content=empty
	mug_condition=dirty
	teeth_condition=clean
		mug_condition=clean

rinse_mouth_clean	mug_content=water
	mouth_condition=foam
	teeth_condition=clean
		mug_condition=dirty
		mouth_condition=wet
		mug_content=empty

rinse_mouth_wet	mug_content=water
	mouth_condition=dry
		mug_condition=dirty
		mouth_condition=wet

brush_teeth	brush_content=paste
	teeth_condition=dirty
	mouth_condition=wet
		teeth_condition=clean
		brush_content=no_paste
		mouth_condition=foam
		brush_condition=dirty

clean_brush	brush_condition=dirty
	teeth_condition=clean
		brush_condition=clean
		brush_content=no_paste

use_towel	mouth_condition=wet
	teeth_condition=clean
		mouth_condition=dry

whereas cleaning the mouth after the brushing task is *rinse_mouth_clean*.

The results of the IU analysis described in this section are integrated into the TEBRA system: the decomposition of the task into user behaviors and the behavioral variables are utilized in the hierarchical recognition framework. The progress state space and the preconditions and effects of user behaviors are integrated into the planning and decision making framework. Both components are described in more detail in the following section.

5 SYSTEM COMPONENTS

We built a washstand setup as depicted on the left in Figure 1 which we equipped with a set of sensors for environmental perception. We use a combination of unobtrusive sensors installed in the environment and tools. We don't attach any wearable sensors to the user directly, because we don't want to disturb in task execution. The washstand setup is equipped with two 2D cameras observing the scene from an overhead and a frontal perspective. A flow sensor installed at the water pipe measures whether the water flow is on or off. The toothbrush is equipped with a 9-DOF[1] sensor module including accelerometer, gyroscope and magnetometer in 3 axis each. We extract a set of features from the sensors which we feed into the hierarchical recognition framework described in the following subsection.

[1]Degree of freedom.

5.1 Hierarchical Recognition Framework

The IU analysis decomposes a task into different user behaviors. Each user behavior is further subdivided into single steps which are described in terms of environmental states. Hence, the IU analysis structures the task into a hierarchy of user behaviors and combines semantic information about the user behavior with environmental states. In our approach, we make use of the hierarchical structure obtained in the IU analysis: we use a two-layered framework for user behavior recognition modeling the hierarchical structure as shown in Figure 2. A detailed description of the

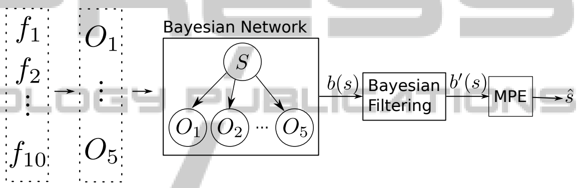

Figure 2: Overview of the hierarchical recognition framework. See text for a detailed description.

recognition framework can be found in (Peters et al., 2012). We extract ten features ($f_1 ... f_{10}$) from the sensory information: from the camera images, we extract the position of the mug, towel and paste using a detector based on color distribution which provides the center positions of the object's bounding boxes. The flow sensor returns a binary feature indicating water flow. For brush movement detection, it is sufficient to extract only the 3-dimensional gyroscope data which is the change in orientation in x, y and z direction.
The features $f_1 ... f_{10}$ are discretized into an intermediate representation of state space variables $O_1 ... O_5$ given in the upper part of Table 2. $O_1 ... O_5$ are obtained from the IU analysis and encode the environmental configuration of user behaviors in terms of behavior variables. $O_1 ... O_5$ which we will call *observation variables* in the following, correspond to the variables *mug_position*, *towel_position*, *paste_movement*, *brush_movement* and *tap_condition*. The position of the mug and towel as well as the movement of the toothpaste are calculated by comparing the bounding box hypotheses of the object detectors to pre-defined image regions like *counter* or *tap*. The movement of the brush is computed using the gyroscope data obtained by the sensor module in the toothbrush. The condition of the tap is set according to the flow sensor. In our framework, we aim to recognize the user be-

haviors obtained from the IU analysis in Table 1. We subsume the user behaviors *fill_mug* and *clean_mug* to a common user behavior *rinse_mug* because the relevant observation variables as well as the according states are nearly identical for both user behaviors. In a regular trial, user behaviors don't follow exactly on each other, but mostly alternate with transition behaviors, for example the user's hand approaches or leaves a manipulated object. We consider these transition behaviors by adding a user behavior *nothing* which we treat as any other user behavior in our recognition model. In this work, we use a Bayesian Network (BN) to classify user behaviors based on the observations $O_1...O_5$. A BN is ideally suited to model the structural relations between user behaviors denoted by the random variable S and relevant observation variables $O_1...O_5$. We use a BN with a *NaiveBayes* structure as depicted in Figure 2. Each observation variable O_i is conditionally independent given the user behavior:

$$P(O_1,...,O_5,S) = P(S) \cdot \prod_{i=1}^{5} P(O_i|S) \qquad (1)$$

The BN with *NaiveBayes* structure has the ability to deal with small training sets since the probability of each O_i depends only on the user behavior S. This is important in our work, because some user behaviors like *clean_brush* are rare compared to other behaviors and the acquisition of data in our scenario is very hard. The result of the BN classification scheme is a belief $b(s)$, a probability distribution over user behaviors. The BN is prone to faulty observations which happen occasionally in the discretization of features into observation variables. Faulty observations lead to rapid changes in the belief b from one time step to the next. This is not desirable in our scenario, because transitions between user behaviors are rather smooth due to the nature of the task. Hence, we extend our framework with a transition model which takes into account the belief of the preceding time step. This results in a Bayesian filtering approach similar to the forward algorithm in a Hidden Markov Model. The belief b is updated to a consecutive belief b' using

$$b'(s') = \frac{O(s',o) \cdot \sum_{s \in S} T(s',s) \cdot b(s)}{C} \qquad (2)$$

with the normalization term $C = \sum_{s' \in S} O(s',o) \cdot \sum_{s \in S} T(s',s) \cdot b(s)$. $O(s',o) = \prod_{i=1}^{5} P(O_i|s')$ is the probability of making observation o when the user behavior is s'. The observation model $O(s',o)$ and transition model $T(s',s)$ are learned on manually annotated training data using Maximum Likelihood (ML) estimation. The transition model in equation 2 leads to smooth state transitions between user behaviors because single faulty observations can't rapidly change the entire belief from one time step to the next. The most probable explanation from b' - the user behavior s' with the highest probability - is fed into the temporal integration mechanism.

5.2 Planning and Decision Making

The planning and decision making framework determines whether to prompt the user during task execution and chooses a prompt as appropriate. The main paradigm in our system is that a prompt should only be given to the user when necessary in order to foster the independence of the user. A prompt is necessary when the user gets stuck in task execution or performs a step which is not appropriate at that time. In order to check whether a user behavior is appropriate, we maintain the user's progress in the task utilizing the progress state space described in section 4. We update the progress state space based on the occurrence of user behaviors: when a behavior is recognized by the system, the progress state space is deterministically updated with the effects of the user behavior. For example, if *paste_on_brush* is recognized, the variables *brush_content* and *brush_condition* of the progress state space are set to *paste* and *dirty*, respectively, according to the effects in Table 3. All other variables remain unchanged. Monitoring the user's progress in the task is a very challenging problem due to the huge temporal variance in behavior execution. For example, one user may successfully perform *paste_on_brush* much slower compared to another user. Furthermore, the execution time of a single person may vary from day to day, especially for persons with cognitive disabilities. We cope with the huge intra- and inter-personal variance in a generalized timing model as described in the following subsection.

5.2.1 Timing Model

The hierarchical recognition framework works in a frame-wise manner: the sensors in our setup are synchronized to a rate of 15 Hz. For each set of sensor data, the hierarchical recognition framework calculates the most probable user behavior *ub*. The planning framework works on a real-time scale: durations of user behaviors are measured in seconds. Hence, we apply a temporal integration mechanism between recognition and planning framework: we maintain a local history, a list of user behaviors to which the most probable behavior *ub* calculated in the recognition framework is added. For each *ub*, we calculate the occurrence o of *ub* in the history. As long as $o \geq 0.8$, the duration of the current behavior is measured in seconds. If $o < 0.8$, we reset the current behavior time to 0. With a threshold of 0.8, we allow for misperceptions in the hierarchical recognition framework without resetting the current behavior duration in case of single sensor errors. The temporal integration mechanism provides the duration of the current user behavior in seconds which is fed into a Finite State Machine (FSM). The FSM depicted in

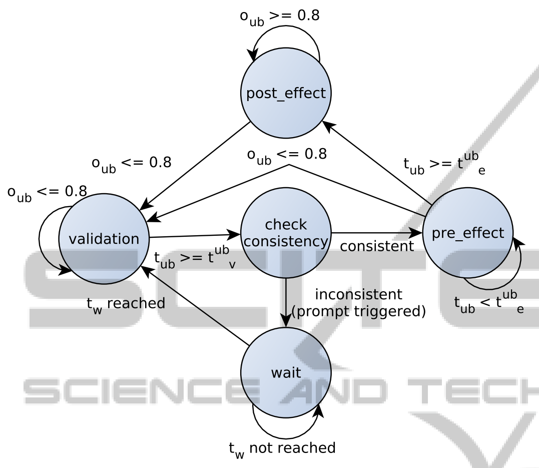

Figure 3: Finite State Machine used to model the different phases in user behavior timing.

Figure 3 models the timing behavior in our planning framework. A FSM is suitable to model the different phases during a behavior: we validate a user behavior over a certain period of time (*validation* state) before we check the consistency (*check consistency* state). We refer to this time as validation time t_v^{ub} based on the average duration of the behavior. We compare the duration t_{ub} of the current behavior *ub* to t_v^{ub}. If $t_{ub} \geq t_v^{ub}$, the FSM transits to state *consistency_check*. The consistency of *ub* is checked by comparing the preconditions of *ub* with the progress state space. If the preconditions are not fulfilled, *ub* is inconsistent which means that *ub* is not an appropriate behavior at that time. Hence, a prompt is delivered to the user. We describe the selection process of a prompt in detail in section 5.2.2. If the preconditions are fulfilled, *ub* is consistent. If the consistency check occurs too early, the user might feel patronized by the system. If the check is too late, the behavior effects might have been erroneously occurred already although the behavior is inconsistent. If *ub* is consistent, the FSM transits to state *pre_effect*. The effects of a user behavior occur after a minimum duration of the behavior which we call effect time t_e^{ub}. State *pre_effect* denotes that the duration of *ub* is too short for the effects to occur because t_e^{ub} is not reached. If $t_e^{ub} \geq t_{ub}$, we update the progress state space by applying the effects of *ub*. The FSM transits to state *post_effect*. For any user behavior (except for *nothing*), a timeout t_t^{ub} may occur in the *post_effect* state. A timeout t_t^{ub} denotes that the user might be stuck in task execution. If the duration $t_{ub} \geq t_t^{ub}$, a timeout prompt will be selected and delivered to the user. After a prompt, the FSM transits to a state *wait* for a fixed time $t_w = 5$s in order to wait for the user to receive the prompt and react properly.

We maintain a different set of timing parameters t_v^{ub} t_e^{ub} and t_t^{ub} for each user behavior in order to cope with the huge variance in temporal execution of individual behaviors. For example, the duration of *rinse_mouth* is usually much shorter compared to *brush_teeth*. Hence, the effect time t_e^{ub} and timeout t_t^{ub} of the behaviors are completely different. The validation time t_v^{ub} can be set higher for longer behaviors to avoid a misdetection of the behavior due to perception errors. In addition to the different durations of user behaviors, we cope with different velocities of users by maintaining a set of timing parameters for three different user velocities: $t_{v_i}^{ub}$ $t_{e_i}^{ub}$ and $t_{t_i}^{ub}$ where $i = \{f, m, s\}$ corresponding to *fast*, *medium* and *slow* execution velocity chosen manually by the authors. The parameters are estimated using an unsupervised learning mechanism: we apply a k-means algorithm to user behavior durations which we recorded in 18 test trials. We cluster the durations of each user behavior into $k = 3$ classes corresponding to *fast*, *medium* and *slow* execution velocity. We fit a Gaussian distribution $\mathcal{N}_i^{ub}(\mu, \sigma^2)$ over the members of each cluster. We calculate the timing parameters $t_{v_i}^{ub}$ $t_{e_i}^{ub}$ and $t_{t_i}^{ub}$ using the inverse cumulative distribution function (invCDF) of $\mathcal{N}_i^{ub}(\mu, \sigma^2)$. For a given probability p, invCDF returns the duration at which the cumulative distribution function (CDF) is p. Exemplarily, we depict the CDF of *rinse_mouth_wet* for velocity *fast* in Figure 4: We calculate the validation time $t_{v_i}^{ub}$ based on the average length of *ub* in velocity model *i*:

$$t_{v_i}^{ub} = 0.3 \cdot \mu_i^{ub} \tag{3}$$

where μ_i^{ub} is the mean duration of the user behavior *ub* and velocity model *i*. $t_{v_i}^{ub}$ denotes that we validate behavior *ub* in velocity model *i* for a duration of 0.3 times the mean duration of *ub* in *i*. The effect time and timeout are set with respect to invCDF:

$$t_{e_i}^{ub} = \text{invCDF}(0.3) \tag{4}$$

$$t_{t_i}^{ub} = 2.5 \cdot \text{invCDF}(0.9) \tag{5}$$

with $\text{invCDF}(x) = \mu_i^{ub} + \sigma_i^{ub} \cdot (-1) \cdot \sqrt{2} \cdot \text{erfcInv}(2x)$ and erfcInv is the inverse complementary error function. $t_{e_i}^{ub}$ denotes that the effects of behavior *ub* in velocity model *i* occur after a duration of invCDF(0.3).

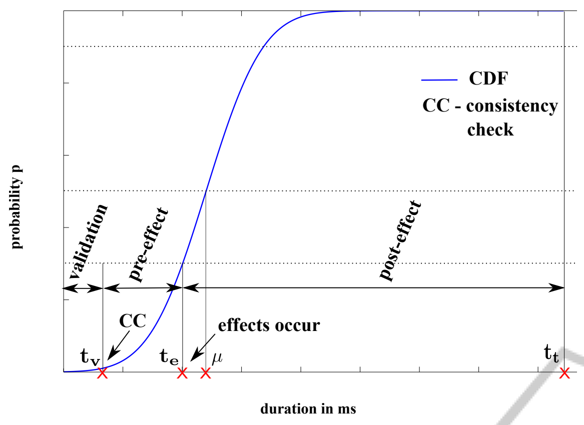

Figure 4: Cumulative distribution function of *rinse_mouth_wet* for velocity *fast* showing the different timing parameters and according phases of the FSM.

We chose a small parameter $p_{v,e} = 0.3$ in the calculation of the effect time of a behavior due to the following reason: missing a successful behavior and an update of the progress state space due to a large effect time is not desirable because it leads to an incorrect progress state space. Setting the effects of a behavior earlier than necessary is not critical since we already validated the user behavior. The description of $t_{t_i}^{ub}$ is analogue to $t_{e_i}^{ub}$. We have a total of $N = 72$ timing parameters: for each of the eight user behaviors given in Table 3, we have three velocities with three timing parameters each. We calculate the parameters based on two meta-parameters manually set to $p_{v,e} = 0.3$ and $p_t = 0.9$ used in equations 3, 4 and 5.

In order to cope with the intra-personal variance in temporal execution of user behaviors, we use the learned parameters in a dynamic timing model: when a user behavior *ub* switches to a successor behavior *ub*$'$, we determine the duration of *ub*. We categorize the duration into one of the velocity classes *fast*, *medium* and *slow*. We maintain a discrete probability distribution over the velocity classes. We determine the current user velocity by applying a winner-takes-all method on the probability distribution which chooses the velocity occurring most frequently during the trial so far. The timing parameters of the FSM are set according to the user's velocity.

The FSM models the timing behavior of the TEBRA system. The selection mechanism of an appropriate prompt is described in the following subsection.

5.2.2 Prompt Selection

The prompt selection mechanism determines the content and modality of prompts given to the user. The content of a prompt is a hint on a user behavior and denotes an appropriate next step in task execution. The modality describes the sensory channel on which the prompt is delivered to the user. In the TEBRA system, we apply two types of visual prompts accompanying a verbal command: (1) pictograms and (2) real-time videos. The real-time videos are prerecorded showing a human actor performing the desired behavior. In a preliminary study using a Wizard-of-Oz paradigm with persons with cognitive disabilities (Peters et al., 2011), some users didn't react to audio prompts and were not able to proceed in task execution. Hence, we implemented an escalation hierarchy to provide prompts with different levels of information to the user: the type of prompt preferred by the system is a pictogram prompt. If the user doesn't react to the prompt, we escalate in the prompting hierarchy and deliver a video prompt where the behavior is depicted in more detail. We conducted interviews with the caregivers at *Haus Bersaba* about the prompting modalities. The caregivers named pictogram and video prompts paired with a verbal command as the preferred modalities. Amongst the choices were pure audio prompts, visual prompts showing objects to use next and cartoon-like prompts. We have a total of 15 prompts: for each user behavior, we have a pictogram and a video prompt plus an additional pictogram showing that the user has reached the final state of the task.

The prompt selection mechanism is triggered in two cases: firstly, when a timeout of a user behavior occurs. Secondly, when the current user behavior is inconsistent with the progress state space. Inconsistent means that there are open preconditions of the current user behavior which are not fulfilled in the progress state space. In both situations, an appropriate prompt to the user addresses the following requirements: the prompt needs to (1) be consistent with the current progress state space, (2) push forward the user's progress in the task, and (3) provide at least one open precondition of the current behavior as an effect. The latter requirement arises because we want to assist the user in his way of executing the task as far as possible. Hence, we aim to find a prompt which supplies the open precondition and allows the user to re-perform the desired behavior after correctly performing the prompted behavior. For example, assume the user has successfully performed *brush_teeth* and performs *use_towel* afterwards. *use_towel* is inconsistent because precondition *mouth_cond=wet* is not fulfilled (*mouth_cond=foam*). In this situation, two prompts are appropriate: *clean_brush* and *rinse_mouth_clean*.

The prompt selection mechanism would then decide for *rinse_mouth_clean* because it provides the open precondition *mouth_cond=wet* as an effect. If the user performs *rinse_mouth_clean*, he/she can go on performing *use_towel* as desired which would not be the case with *clean_brush*.

Our prompt selection mechanism performs a search on an ordering constraint graph (OCG) to find an appropriate prompt. An OCG is a visualization of a set of ordering constraints which are calculated using a partial order planner: given an initial state I, a goal state G and a set of STRIPS-like actions A with preconditions and effects, the partial order planner calculates a plan to transit from the initial to the goal state. A plan consists of a set of actions A, a set of ordering constraints O (action a before b), a set of causal links C (action a provides condition x for b) and a set of variable bindings B (variable $v = c$ where c is a constant).

In this work, we use a partial order planner to obtain an OCG for the task of brushing teeth. The user behaviors and according preconditions and effects identified in the IU analysis as given in Table 3 form the set of possible actions A. The initial and goal states are extracted from the IU analysis in Table 1 in terms of progress variables: $G = [mug\_content = empty, mug\_condition = clean, mouth\_condition = dry, brush\_content = no\_paste, brush\_condition = clean, teeth\_condition = clean]$ The initial state differs only in the variable $teeth\_condition = dirty$. From the ordering constraints O and the causal links C, we manually construct a directed OCG as depicted in Figure 5. An arrow in the OCG describes that the

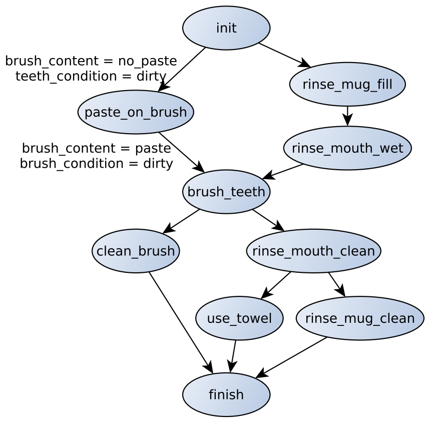

Figure 5: Ordering constraint graph depicting partial orderings of user behaviors in the brushing task. We depict the preconditions and effects of *paste_on_brush*, exemplarily.

source behavior provides necessary preconditions for the target behavior. For example, *rinse_mug_fill* provides the effect $mug\_content = water$ which is a

Algorithm 1: Select appropriate prompt.

```
function SELECTPROMPT(ub)
    op ← GETOPENPRECONDITIONS(ub)
    for all op do
        prompt ← PROCESSPRECONDITION(ub,op[i])
        add prompt to valid prompts
    end for
    if |valid| ≥ 2 then
        prompt ← GETCLOSESTTOGOAL(valid)
        return prompt
    else
        return valid[0]
    end if
end function
```

Algorithm 2: Process precondition.

```
function PROCESSPRECONDITION(ub,op)
    supplyUB ← FINDSUPPLYUB(ub, op)
    if supplyUB is empty then
        prompt ← FINDCONSISTENTPREDECESSOR()
        goal
        return prompt
    else
        CHECKCONSISTENCY(supplyUB)
        if supplyUB is consistent then
            return supplyUB
        else
            return SELECTPROMPT(supplyUB)
        end if
    end if
end function
```

precondition of *rinse_mouth_wet*. The OCG depicts no strict execution plan of the task which the user has to follow, but models the ordering between behaviors in the overall task: the behavior sequence $rinse\_mug\_fill, paste\_on\_brush, rinse\_mug\_fill$ e.g. is consistent with respect to the partial ordering given in Figure 5.

We search for an appropriate prompt in the OCG as described in algorithm 1: We determine the open preconditions of the inconsistent user behavior *ub*. We process each open precondition as described in algorithm 2: we search for a user behavior *ub*′ which is a predecessor of *ub* in OCG and provides the open precondition. If *ub*′ exists, we check the consistency with regard to the progress state space. If *ub*′ is consistent, *ub*′ is an appropriate prompt. If *ub*′ is also inconsistent due to open preconditions, we recursively call *selectPrompt* with $supplyUB$ in order to find a behavior resolving the open preconditions of $supplyUB$. By recursively calling *selectPrompt*, we resolve chains of open preconditions over several user behaviors by iterating backwards through the OCG. If no predecessor of *ub* is found providing the open precondition, we search for a consistent behavior by iterating backwards through the OCG starting at the *goal* node. By starting at the goal node, we ensure to find a consistent behavior which is most closely to the desired goal state. In case of a timeout, the prompt selection mechanism directly searches for a consistent user behavior starting at the *goal* node.

We evaluate the technical correctness of the planning framework as well as the hierarchical recognition framework in a user study described in the following section.

6 USER STUDY AND RESULTS

We evaluate the first prototype of the TEBRA system in a study with a total of 26 trials. Each trial is a single brushing task performed by a regular user. A study with regular users is feasible since regular persons show individual ways in the execution of the task which may not coincide with the system's framework of action. The system prompts the user who has to adapt to the prompts to successfully execute the task from a system's point of view. Hence, we provoke similar phenomena in terms of prompting and reaction behavior in a study with regular users compared to a study with persons with cognitive disabilities. However, we aim to conduct a study with persons with cognitive disabilities in the future. The goal of the study described here is two-fold: we aim to evaluate (1) the technical correctness of the system and (2) the user's reaction to system prompts. The reaction of regular users to system prompts is a measure whether the prompts are semantically reasonable to a minimum degree: if regular users have problems understanding the prompts, they might most likely be inappropriate for persons with cognitive disabilities. The 26 trials were performed by 13 users. Each user performed a single trial in each of two different scenarios: in the *free* scenario, users received the instruction to brush teeth as they would regularly do. The system generates prompts if necessary according to the user's task execution. In the *collaborative* scenario, the user is instructed to perform the brushing task in collaboration with the system: the user ought to follow the prompts whenever they are appropriate.

6.1 Technical Correctness and System Improvement

The technical correctness of the TEBRA system is highly dependent on the performance of the hierarchical recognition framework and the planning and decision making framework. In the hierarchical recognition framework, we can't distinguish between *rinse_mouth_clean* and *rinse_mouth_wet* because the behavioral variables are nearly identical for both behaviors. In the planning framework, we need to distinguish between *rinse_mouth_clean* and *rinse_mouth_wet* since the behaviors are different in terms of preconditions and effects in Table 3. *Rinse_mouth_wet* describes taking water before brushing the teeth and *rinse_mouth_clean* is performed after the brushing step. We apply a simple heuristic: when *rinse_mouth* is classified by the hierarchical recognition framework, it will be set to *rinse_mouth_wet* if *brush_teeth* has already been recognized in a trial. Otherwise, *rinse_mouth* will be set to *rinse_mouth_clean*. We apply the same heuristic for *rinse_mug_fill* (before *brush_teeth*) and *rinse_mug_clean* (after *brush_teeth*) which are subsumed to a common behavior *rinse_mug* in the recognition framework. Table 4 shows the classification rates of the user behaviors: The classification

Table 4: Classification rates of user behaviors. RMgC - rinse_mug_clean, RMgF - rinse_mug_fill, UT - use_towel, PB - paste_on_brush, RMC - rinse_mouth_clean, RMW - rinse_mouth_wet, BT - brush_teeth, CB - clean_brush, N - nothing.

	RMW	RMC	RMgF	RMgC	BT	PB	CB	UT	N
RMW	**79.5**	0.0	0.0	0.0	0.0	0.6	0.0	0.0	19.9
RMC	32.5	**42.0**	2.4	1.6	0.0	0.1	9.9	0.0	11.5
RMgF	1.9	0.8	**86.0**	4.4	0.0	0.0	3.9	0.0	3.0
RMgC	0.0	6.2	0.0	**78.8**	0.0	0.0	0.5	8.0	6.6
BT	0.6	0.0	0.1	0.0	**70.0**	25.6	0.3	1.0	2.4
PB	0.0	0.0	0.0	0.0	0.5	**99.2**	0.0	0.0	0.3
CB	0.8	0.0	0.2	1.8	1.9	6.6	**85.4**	0.1	3.2
UT	0.0	0.0	0.0	0.2	0.1	0.6	0.0	**87.1**	12.1
N	3.9	2.4	1.3	1.4	2.8	22.8	2.0	9.8	**53.7**

rates of *rinse_mug_fill*, *paste_on_brush*, *clean_brush* and *use_towel* are very good with 86%, 99.2%, 85.4% and 87.1%, respectively. However, the rate of *rinse_mouth_clean* is very low with 42%. The heuristic is highly dependent on the recognition of *brush_teeth* which has a classification rate of 70%. The recognition of *brush_teeth* is challenging: the gyroscope in the brush sensor module measures the change in orientation. The changes are integrated over time to obtain the absolute orientation on which the behavior variable *brush_movement* is set. In the integration process, small errors are accumulated. For a behavior like *brush_teeth* which usually has a long duration compared to other behaviors, the accumulation of errors leads to misclassifications: *brush_teeth* was mixed up with *paste_on_brush* in 25.6% of the cases. However, the average classification rate of 75.7% over all user behaviors is a very good result with regard to the huge variance in task execution.

Table 5: Coll - collaborative scenario, #P - number of prompts, avg P - average nr of prompts, SC - semantically correct prompts, C - correct reaction to a prompt, CSC - correct reaction to a semantically correct prompt, dur - minimum(maximum) duration, FSR - final state reached.

	#P	avg P	SC(%)	C(%)	CSC(%)	dur	FSR(%)
free	87	6.7	59	10	10	63(184)	8
coll	117	9	66	75	85	142(292)	100

6.2 System Performance

The performance of the TEBRA system in the *collaborative* scenario is excellent as depicted in column *FSR(%)* in Table 5: Each of the 13 users reached the final state in this scenario where users ought to follow the prompts when appropriate. In the *free* scenario, only a single user reached the final state: regular users have an individual way of executing the task which may not coincide with the system's framework of action. In order to reach the final state, users have to adapt to the system prompts. In the *free* scenario, users were not explicitly encouraged to react to prompts, but were instructed to brush their teeth as they would regularly do. Since all users were capable of brushing their teeth independently, all users except one didn't follow system prompts which leads to the rate of 8%.

However, the excellent results in the *collaborative* scenario shows that the system is able to assist a user in trials which differ significantly in duration: the minimum (maximum) duration in seconds are 63(184) seconds in the *free* and 142(292) in the *collaborative* scenario. The trials not only differ in the overall durations, but also in the durations of single user behaviors. We cope with the different behavior durations using the dynamic timing model as described in section 5.2.2. Exemplarily, we show the advantage of the dynamic timing model in two situations: in the first situation, the timing model switched from user velocity *slow* to *fast*. A subsequent user behavior *rinse_mouth_wet* was correctly recognized using the timing parameters of *fast*. However, with the timing parameters of *slow*, *rinse_mouth_wet* would not have been recognized because the duration would have been too short for the effects to occur. The recognition of *rinse_mouth_wet* would have been missed by the system. In a second situation, the timing model switched from user velocity *fast* to *slow*. In the subsequent user behavior *brush_teeth*, a perception error occurred and *brush_teeth* was erroneously recognized as *paste_on_brush* for a duration t. With the timing parameters of velocity *fast* or *medium*, the system would have erroneously recognized *paste_on_brush* as a new behavior due to the decreased validation time. With velocity *slow*, the duration t was too short for *paste_on_brush* to be recognized. The timing model avoided a perception error which would have led to false prompts in further task execution.

6.3 Appropriate Prompting

An important measure of the performance of our system is the number of prompts which are semantically correct. Semantically correct means that the type of prompt is appropriate with regard to the user's progress in the task so far: we determine the semantical correctness by using a ground truth annotation of the behaviors in the task which was done by the first author of the paper. In the *collaborative* scenario, 66% of the total number of 117 prompts were semantically correct as depicted in Table 5. The 34% of semantically incorrect prompts contain follow-up prompts arising from perception errors: when a user behavior was successfully performed, but not recognized correctly, the system delivers follow-up prompts which are semantically incorrect. If we only regard the first prompts of the system in each trial, the number of semantically correct prompts increases to 92% which is a very good rate based on the complexity of the task.

The user's reaction to prompts measures whether the prompts are meaningful to the user. The reaction of the user is correct when he/she updates the behavior to what was prompted by the system. We found 75% of correct reactions in the *collaborative* scenario including semantically correct and incorrect prompts. Correct reactions to semantically correct prompts are much higher with 85%. A major challenge in enhancing the TEBRA system will be the increase of semantically correct prompts which depends highly on the performance of the hierarchical recognition framework. The system performance is very good measured in terms of correct reactions to prompts which are appropriate for the user. Additionally to the appropriateness of prompts in terms of timing and meaning, we asked the users to judge the modality of the prompts in terms of prompt duration and understandability in a questionnaire. Prompt duration and understandability were evaluated with a score of 5.5 and 6.1 of 7, respectively. A value of 1 denotes insufficient and 7 denotes perfectly good. This result underlines that the prompts are understandable at least to regular persons. We will evaluate in a future study whether persons with cognitive disabilities are also able to follow the prompts. We are optimistic, because the prompting modalities were selected as a result of interviews with caregivers of *Haus Bersaba* who found the modalities of prompting to be appropriate for the target group.

The results show that the TEBRA system is able to deal with the huge variance in spatial and temporal execution of the task: the system mostly gives semantically correct prompts and is able to assist users through the entire task of brushing teeth.

7 CONCLUSIONS

In this paper, we describe a novel cognitive assistive technology: the TEBRA system aims to assist users with cognitive disabilities in the complex task of brushing teeth. We use a structured approach based on IU analysis by utilizing the results in the development of the TEBRA system. We tackle the huge variance in spatial and temporal execution of the task: in the hierarchical recognition framework, we abstract from tracking objects or the user's hands. Instead, we infer behaviors based on the environmental state of objects. We deal with the temporal variance in task execution in our planning framework: we apply a Finite State Machine and a dynamic timing model to allow for different velocities of users. We showed in a study with regular users that the TEBRA system is able to monitor the user's progress in the task and provide appropriate prompts to the user if necessary. The user's reactions show that prompting modalities are meaningful to a minimum degree with regard to regular users. Due to the structured approach using IU analysis, the TEBRA system can be easily extended to different tasks.

Future work includes technical enhancements of the system: we aim to recognize user behaviors more robustly by improving the classification mechanisms. We will enhance the dynamic timing model by learning the parameters on a larger set of training data. We aim to evaluate the TEBRA system with persons with cognitive disabilities in the near future.

ACKNOWLEDGEMENTS

The authors would like to thank the inhabitants and caregivers of Haus Bersaba for their cooperation. This work was funded by the German Research Foundation (DFG), Excellence Cluster 277 "Cognitive Interaction Technology". Special thanks to Simon Schulz from the CITEC Central Lab Facilities for integrating the sensor module into the toothbrush.

REFERENCES

Bauchet, J., Giroux, S., Pigot, H., Lussier-Desrochers, D., and Lachapelle, Y. (2008). Pervasive assistance in smart homes for people with intellectual disabilities: a case study on meal preparation. *Assistive Robotics and Mechatronics*, 9(4):42–54.

Giroux, S., Bauchet, J., Pigot, H., Lussier-Desrochers, D., and Lachappelle, Y. (2008). Pervasive behavior tracking for cognitive assistance. In *PETRA'08*, pages 1–7.

Hoey, J., Ploetz, T., Jackson, D., Monk, A., Pham, C., and Olivier, P. (2011). Rapid specification and automated generation of prompting systems to assist people with dementia. *Pervasive and Mobile Computing*, 7(3):299 – 318.

Hoey, J., Poupart, P., Bertoldi, A. v., Craig, T., Boutilier, C., and Mihailidis, A. (2010). Automated handwashing assistance for persons with dementia using video and a partially observable markov decision process. *Computer Vision and Image Understanding*, 114:503–519.

Moore, D., Essa, I., and Hayes, M. (1999). Object Spaces: Context Management for Human Activity Recognition. In *AVBPA'99*.

Najjar, M., Courtemanche, F., Hamam, H., Dion, A., and Bauchet, J. (2009). Intelligent recognition of activities of daily living for assisting memory and/or cognitively impaired elders in smart homes. *Ambient Computing and Intelligence*, 1(4):46–62.

Nguyen, N., Phung, D., Venkatesh, S., and Bui, H. (2005). Learning and detecting activities from movement trajectories using the hierarchical hidden markov model. In *CVPR'05*.

Patterson, D. J., Liao, L., Gajos, K., Collier, M., Livic, N., Olson, K., Wang, S., Fox, D., and Kautz, H. A. (2004). Opportunity knocks: A system to provide cognitive assistance with transportation services. In *Ubicomp'04*, pages 433–450.

Peters, C., Hermann, T., and Wachsmuth, S. (2011). Prototyping of an automatic prompting system for a residential home. In *RESNA/ICTA 2011 Conf. Proc. (online)*.

Peters, C., Hermann, T., and Wachsmuth, S. (2012). User behavior recognition for an automatic prompting system - a structured approach based on task analysis. In *ICPRAM'12*, pages 162–171.

Pollack, M. E., Brown, L. E., Colbry, D., McCarthy, C. E., Orosz, C., Peintner, B., Ramakrishnan, S., and Tsamardinos, I. (2003). Autominder: an intelligent cognitive orthotic system for people with memory impairment. *Robotics and Autonomous Systems*, 44(3–4):273–282.

Pusiol, G., Patino, L., Bremond, F., Thonnat, M., and Suresh, S. (2008). Optimizing Trajectories Clustering for Activity Recognition. In *MLVMA'08, Int. Workshop on Machine Learning for Vision-based Motion Analysis*.

Ryu, H. and Monk, A. (2009). Interaction Unit Analysis: A New Interaction Design Framework. *Human-Computer Interaction*, 24(4).