CogWatch:

Jean-Baptiste, Emile M.D ; Rotshtein, Pia; Russell, Martin

DOI:
10.1016/j.jides.2016.10.003

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (Harvard):
Jean-Baptiste, EMD, Rotshtein, P & Russell, M 2016, 'CogWatch: Automatic prompting system for stroke survivors during activities of daily living', *Journal of Innovation in Digital Ecosystems*, vol. 3, no. 2, pp. 45-56. https://doi.org/10.1016/j.jides.2016.10.003

HOSTED BY
ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/jides

CogWatch: Automatic prompting system for stroke survivors during activities of daily living☆

Emilie M.D. Jean-Baptiste[a,*], *Pia Rotshtein*[b], *Martin Russell*[a]

[a] *School of Engineering, University of Birmingham, UK*
[b] *School of Psychology, University of Birmingham, UK*

HIGHLIGHTS

- The contribution of this paper is to provide further knowledge about how an artificial intelligent rehabilitation system such as CogWatch can be conceptualized.
- The process through which CogWatch can learn how to properly provide guidance to stroke survivors during the tea-making task will be explained.
- The algorithms developed for action planning and human error recognition will be evaluated and compared with other techniques.

ARTICLE INFO

Article history:
Published online 1 December 2016

Keywords:
POMDP
Planning under uncertainty
Error recognition under uncertainty
Assistive technology

ABSTRACT

This paper presents CogWatch - an automatic prompting system designed to guide stroke survivors during activities of daily living, such as tea-making. In order to provide guidance during such activities, CogWatch needs to plan which optimal action should be done by users at each step of the task, and detect potential errors in their behavior. This paper focuses on the CogWatch Task Manager, which contains the modules responsible for action planning and human's error detection under uncertainty. We first give an overview of the global assistive system where the Task Manager is implemented, and explain how it can interact with a user during the tea-making task. We then analyze how novel algorithms allow the Task Manager to increase the system's performance.

1. Introduction

Each year, there are more than 100,000 new stroke cases in the UK [1], with over half of all stroke survivors depending on others to carry out Activities of Daily Living (ADL); for example, cooking, grooming, teeth-brushing or making a drink. Stroke survivors face difficulties due to the loss of physical and cognitive functions caused by Apraxia or Action Disorganization Syndrome (AADS) [2–4]. In [5], it is estimated that such cognitive deficits affect 46% of stroke survivors during ADL. For example, when preparing a cup of tea, individuals with AADS may forget to pour water into the kettle then switch it on, skip steps, or misuse objects with possible safety implications. These errors can relate to

Peer review under responsibility of Qassim University.
☆ Fully documented templates are available in the elsarticle package on CTAN.
* *Corresponding author.*
E-mail address: emj198@alumni.bham.ac.uk (E.M.D. Jean-Baptiste).

http://dx.doi.org/10.1016/j.jides.2016.10.003

the defective use of tools and objects [6], the inability to correctly select tools for a task [7], or the inability to complete sequences of actions [8]. These neurological deficits impact on stroke survivors, relatives, caregivers and society as a whole.

Rehabilitation interventions can mitigate the effects of AADS and improve stroke survivors' conditions [9,10]. During such interventions, skilled clinicians guide stroke survivors through ADL by observing their behavior, provide appropriate assistance and prompt them when necessary. However, rehabilitation is costly in both economic and human terms. In the UK, the annual cost of stroke to society is estimated to be £8.9 billion, with about half linked to indirect costs of on-going support [11]. Beyond this economical aspect, the loss of independence affecting stroke survivors' personal privacy also needs to be highlighted. Indeed, caregivers may go to stroke survivors' homes to deliver rehabilitation and recovery care. Some stroke survivors may perceive this as an invasion of their personal space, and be unwilling to accept this over-reliance on caregivers as a long-term solution [12].

Therefore, there is a need for technology that can provide assistance automatically; to reduce AADS related disabilities, and help stroke survivors regain self-sufficiency and independence while keeping their dignity.

2. Related work

In the field of assistive and rehabilitation technology, several Artificial Intelligent systems have been designed to increase independent completion of ADL by individuals with cognitive deficits. Extensive literature reviews focusing on assistive technology for cognition have been published and updated [13,14]. In 2011, 63% of the studies reviewed by Gillespie et al. [14] focused on assistive systems designed to provide reminding and prompting interventions to users. This supports the conclusion of Hart et al. [15] that clinicians saw more potential for devices in the areas of learning/memory, planning/organization and initiation.

The interest in this area led to the development of complex systems such as COACH [16], which provides instructional cueing to guide users during hand-washing. The COACH Markov Decision Process (MDP) based planning system is described in [16,17]. Its goal is to provide appropriate cues to the user during the task. In a new COACH system, a Partially Observable MDP (POMDP)-based planning system [18,19] was implemented to accommodate uncertainty in the system's inputs. The POMDP-based system was evaluated via simulations of hand-washing [18], and compared with heuristic policies and the MDP. Results showed that the POMDP-based planning system performed best, but not significantly better than the heuristic policies. More recently, Peters et al. [20,21] developed the TEBRA system to support mildly impaired people during teeth-brushing. A user study was performed with 7 participants suffering from cognitive deficits. The results showed that users made significantly more independent steps when they had access to the system's prompts [21].

The CogWatch system [22–24] was designed to guide stroke survivors during tea-making. The first CogWatch prototype used a MDP-based planning system [25], which was evaluated with 12 stroke survivors. In a first scenario, stroke survivors completed the tasks by themselves, while in a second scenario they had access to the system's prompts. Results showed that stroke survivors made fewer errors when they were guided by the system [24]. Subsequently CogWatch was enhanced with a POMDP-based planning system to cope with incorrect interpretations of users' behavior. In a planning system (or Task Manager), the module responsible for action planning is the Action Policy Module (APM). The second main component of the Task Manager is the Error Recognition Module (ERM), which is responsible for detecting users' errors during a task. Similarly to the POMDP-based COACH system, the POMDP-based CogWatch system was evaluated via simulation [26]. Results showed that the novel Nearest Neighbor Search (NNS) technique developed by CogWatch's authors (i.e., "SciMK") systematically improved the planning system's performance under uncertainty. However, the POMDP-based ERM did not perform significantly better than the MDP-based ERM [26].

This paper focusses on the CogWatch Task Manager; its POMDP-based APM and ERM. The main contribution is to provide further understanding of how an intelligent rehabilitation system such as CogWatch can be conceptualized. More precisely, SciMK is compared with other techniques in a novel system configuration, and another algorithm for error recognition under uncertainty is proposed and evaluated. The results presented in this paper extend our previous work [26] by demonstrating that "SciMK" outperforms 6 alternative techniques and by verifying its ability to improve planning system performance. In addition, this paper focuses on the techniques used to train the ERM, and their impact on its ability to correctly detect user errors.

For clarity, a summary of the main abbreviations used in this paper is given in Table 1.

Table 1 – Nomenclature.

AADS	Apraxia or action disorganization syndrome
ADL	Activity of daily living
AI	Artificial Intelligence
CW	CogWatch
SimU	Simulated User
ARS	Action Recognition System
TM	Task Manager
APM	Action Policy Module
ERM	Error Recognition Module
MC	Monte Carlo
MDP	Markov Decision Process
NNS	Nearest Neighbor Search
POMDP	Partially Observable Markov Decision Process
BT	Black tea
WT	White tea
BTS	Black tea with sugar
WTS	White tea with sugar

3. The CogWatch system

The workflow in the CogWatch system is as follows (Fig. 1):

1. The user is given a task to complete.

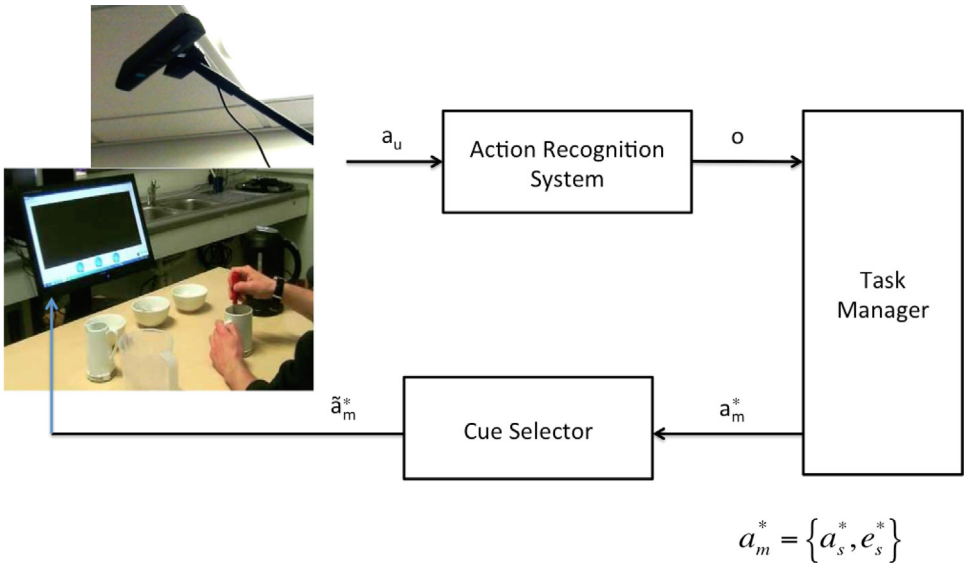

Fig. 1 – CogWatch system's architecture.

2. The user moves objects and performs actions a_u related to the given task.
3. Sensors in the environment and smart objects monitor the user's actions.
4. The sensors' outputs are analyzed by an Action Recognition System (ARS) which outputs its interpretation/observation o of the user's actions.
5. A Task Manager plans which optimal action a_s^* the user should do, detects if an error has been made e_s^*, then passes the information to a cue selector.
6. The cue selector chooses in which form the Task Manager's output a_m^* should be delivered to the user.

3.1. *Monitoring module*

An assistive system to remind a user what to do during a task needs monitor the user's environment and behavior. In CogWatch, the user is observed via sensors embedded in objects in the environment, and via the camera part of Kinect™ which tracks user hands coordinates [27]. Data from the sensors and Kinect™ are outputs by the monitoring module and recognised as actions by the ARS.

3.2. *Action Recognition System*

The purpose of the ARS is to infer the user's current action from the monitoring module's outputs. Action recognition is challenging due to the spatial variance in the execution of the task, and variability in how users move objects to perform the same action. In CogWatch, action recognition uses action-dependent Hidden Markov Models (HMMs) [28,27]. There is no guarantee that the interpretation o of the user's action output by the ARS is correct, and wrong information may be sent to the Task Manager.

3.3. *Task Manager*

In the prototype CogWatch system, the Task Manager was based on a MDP. However, such a system is poorly equipped to recover from ARS errors. Hence, since ARS error are inevitable, the most recent versions of the Task Manager are implemented as a POMDP. In contrast to the MDP, the POMDP can support decision planning in the case where the system cannot directly observe the *state* of the user, where the state of the user corresponds to the sequence of actions that he or she has completed so far [26].

Fig. 2 shows the structure of the POMDP-based Task Manager. It comprises two main modules: the APM and ERM (Section 2). Under uncertainty, the user's state s is unknown. Instead the user's behavior is characterised as a *belief state* b which is a probability distribution over the user states. The POMDP combines an output o from the ARS, the current belief state b and its knowledge of the probabilities of different types of ARS error to create a new belief state b'. Thus, after each ARS observation o, the belief states b_s and b_e for the APM and ERM, respectively, are updated to create new belief states b'_s and b'_e. In the APM the POMDP-based Task Manager uses b'_s to select the best next action a_s^* that the user should follow to successfully finish or continue a task. Similarly, in the ERM b'_e is used to select an output e_s^*, which is either True or False depending on whether or not the user has just made an error. The outputs a_s^* and e_s^* are then passed in the form of a prompt a_m^* to the cue selector.

More information about how the POMDP-based Task Manager updates belief states can be found in [26].

3.4. *Cue selector*

The Cue Selector automatically chooses the appropriate cue to be sent to the user during the task. Cues can be vocal commands, videos showing an actor performing the action the user should mirror to continue the task, a picture giving a hint about the next best step to take, or a written message displayed on a screen stating what should be done. The Cue Selector decides the form in which the cue $\tilde{a}_m^*$ is shared. However, the core information in the cue comes directly

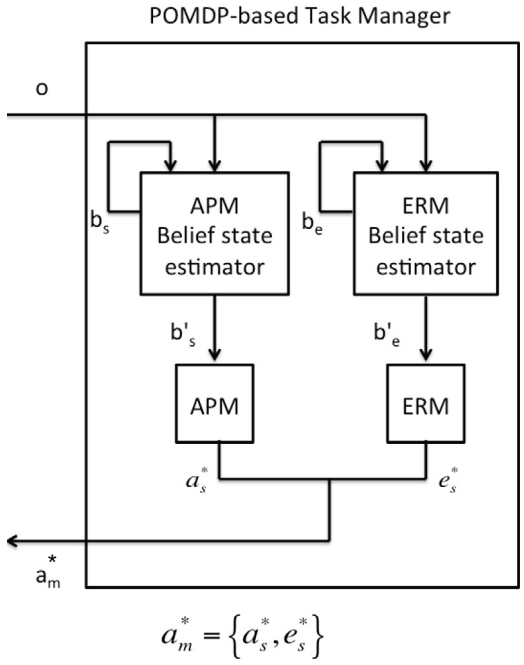

Fig. 2 – CogWatch's POMDP-based Task Manager.

from the Task Manager. When the appropriate form of cue is selected, it is presented via a screen and speakers at the location of the task [22,25,24].

4. Task definition

The CogWatch system targets issues that arise in the preparation of a hot drink such as tea: Black tea (BT), Black tea with sugar (BTS), White tea (WT) and White tea with sugar (WTS). Note that in our case, the difference between a black tea and a white tea is that the latter contains milk.

4.1. *Actions tree*

It is shown in [29], that an ADL can be divided in multiple levels of action steps. In CogWatch, the tea-making task is defined using a set A of top level actions: "Fill kettle", "Boil water", "Pour kettle" (i.e., pour boiling water into the cup), "Add teabag", "Add sugar", "Add milk", "Remove teabag", "Stir". Describing the task in this enables different types of errors in the user's behavior to be identified. Schwartz et al. [29] distinguished between six error types.

4.2. *Error types*

In the CogWatch tea-making task, the following errors are accounted for:

1. An **Addition Error** occurs if the user performs an action form another task. For example, when making "Black tea" the user adds sugar.
2. A **Perseveration Error** occurs if the user repeats an action already performed during the task.
3. An **Anticipation Error** occurs if the user performs an action too soon relative to the current action history. For example, stirring an empty cup.
4. A **Perplexity Error** occurs if the user fails to perform an action during a specified time period *T*. A counter is reset after each observation received, and the Task Manager considers that the user needs assistance if no relevant action is received after *T* seconds.
5. An **Omission Error** occurs if the user considers that tea preparation is finished but the Task Manager detects that the task is incomplete. For example a tea-bag has not been put in the cup.
6. A **Fatal Error** occurs if the user performs any action that can potentially cause injuries (for example, toying with boiling water), or when one of the above errors is repeated too many times.

5. User simulation

While the ARS's challenge is to cope with variability in the sensor outputs and the way that actions are performed, the Task Manager's challenge is to accommodate variability in task completion and any erroneous ARS outputs. Ideally, the system should be evaluated by letting stroke survivors interact with it. However, user simulation is an alternative established way to evaluate POMDP techniques [30,31] and is known to provide a useful basis for comparing systems [32]. It offers a wider coverage of user space, and the possibility to analyze the main elements in the Task Manager in a controlled way. To train or evaluate the Task Manager in this way, we need to synthesize user actions identical to the Task Manager inputs (including ARS errors), and simulate the same variability in task completion as real users. Our simulated user (*SimU*) is based on bigram probabilities estimated from stroke survivors' data. It is explained in detail in [22].

6. Training the APM via simulation

To fulfill its goal, the POMDP-based APM needs to find an optimal policy π^*, which is a mapping from each belief state b_S in a belief subspace B_S to an optimal action $a_S^* \in A$. A grid-based approximation method was implemented, as proposed by Young et al. in [32], using the *SimU* and a Monte Carlo Algorithm, as explained in [26]. As discussed in [26], during evaluation the POMDP-based APM may encounter a belief state that was not seen during training, and for which no optimal action exists. In such a case, a technique inspired by the "cache plan" method discussed in [33] was implemented. This technique consists of finding a neighbor of a given belief state and imitating its behavior [34,26]. Hence, during evaluation, selecting the optimal policy for a belief state can be seen as a classification problem. Consequently the quality of the optimal actions selected by the POMDP-based APM depends on the Nearest Neighbor Search (NNS) technique used during evaluation. Indeed, as highlighted in [35–37], the accuracy of k-nearest neighbor classification depends on the metric applied, and on the dimensionality of the points taken into account.

7. Training the ERM via simulation

The task of the ERM is to find an optimal policy ε^*, which associates each belief state b_e in a belief subspace B_e with an optimal label e_s^*. In [26] the approach to error detection under uncertainty was unsuccessful and the ERM in the POMDP-based system failed to outperform the one implemented in the MDP-based system. In this section a different approach to user error detection under uncertainty in the ERM is proposed (see Algorithm, Fig. 3). The algorithm works as follows:

- The *SimU*'s initial state s_0 is empty (no action has been performed), and the initial belief state is b_0, with $b_0 = b(s_0) = 1$.
- In each virtual tea-making trial, the *SimU* passes actions to the virtual ARS [26] until it considers that the task is complete. When the ARS receives an action it outputs an observation, the belief state is updated, and the current true state s of the *SimU* is noted.
- Following the current ERM policy, the new belief state b is labeled as e, which is *True* or *False*. The algorithm then verifies whether the true user state s is erroneous or not. If the ERM label for b is correct it incurs a negative cost (-1000), otherwise it incurs a positive cost ($+1000$).
- After each suggestion, $Q(b, e)$ is updated, where $Q(b, e)$ is the cost of a trial starting in belief state b, the ERM selecting output e, and thereafter proceeding according to the current policy ε, until the trial ends and a final belief state is reached.
- At the end of each trial, the current policy is updated, considering that the label incurring the lowest cost for each belief state is the optimal one. The whole process is repeated until convergence.

8. Evaluation via simulation

8.1. Evaluation of the APM

To evaluate the APM, the impact of its best next actions a_s^* on the simulated user's [22] success rate at varying ARS error rate, and with different Nearest Neighbor Search (NNS) techniques, is analysed. The cue selector is configured to pass the APM's prompts directly to the *SimU*. The *SimU* was 100% compliant: each time the *SimU* received a_s^*, it continued the task by selecting $a_u = a_s^*$. The virtual ARS is defined by an action confusion matrix. To analyze the impact of ARS error rate on APM performance, three confusions matrices were considered (see Tables 2–4). In these confusion matrices, rows correspond to the user's true actions and columns correspond to ARS outputs. The actions are abbreviated as follows: 'FillK' (Fill kettle), 'BoilW' (Boil water), 'AddTB' (Add teabag), 'PourK' (Pour kettle), 'AddSu' (Add sugar), 'AddMi' (Add milk), 'StirT' (Stir), 'RemTB' (Remove teabag), 'PourW' (Pour water from jug to cup), 'ToyBW' (Toy with boiling water), 'NoAct' ($\emptyset$ (no action performed or detected)).

Intuitively, as the ARS error rate increases the TM will have more difficulty outputting correct prompts, leading to more user errors. However, the type of the task and its difficulty and the methods implemented by TM may also impact user performance. The aim of this evaluation is to gain a better understanding of how ARS error rate, task complexity and the NNS technique used in the POMDP affect the task completion rate for a fully compliant user interacting with the system.

8.2. Evaluation of the ERM

To evaluate the ERM, we analyzed its ability to correctly detect user errors at 30% ARS error rate during preparation of "Black tea". We also investigated the impact of the simulated user's behavior during training on ERM performance. The POMDP-based ERM trained using the algorithm from Section 7 was also compared with the performance of the ERM implemented in the MDP-based system.

9. Results

9.1. Performance of the APM

Tables 5–7 show the *SimU*'s performance when interacting with the MDP-based APM, the POMDP-based APM (applying different NNS techniques), and when being 0% compliant to the APM outputs, for respectively "Black tea", "Black tea with sugar", "White tea with sugar". Note that for each task and each NNS technique, the *SimU* performed 300 trials. When the *SimU* is configured to be 0% compliant, it performs the tasks by itself and ignores the system's prompts. Thus its performance is independent from the ARS error rate. The *SimU*'s success rate at 0% compliance is a way to measure the usefulness of the system. Indeed, in Table 5, one can see that at 20% ARS error rate, it is more advantageous for the *SimU* to perform the task by itself rather than comply 100% of the time with the MDP-based APM. In the same table, one can also see that the NNS techniques implemented in the POMDP-based APM have a similar impact on the *SimU* success rate, except for SciMK and the correlation distance, which allowed the *SimU* to succeed "Black tea" 100% of the time at varying ARS error rates. During "Black tea with sugar", Table 6, one can note that the choice of the NNS technique influenced the ability of the POMDP-based APM to select the appropriate prompt for the *SimU*. For example, at 30% ARS error rate, when the POMDP-based APM used the Euclidean metric to select neighbors, it succeeded to make the *SimU* correctly complete its task 55% of the time. By contrast, when SVM or SciMK were implemented, the *SimU* respectively succeeded its task 74% and 83% of the time. The same phenomenon occurred during "White tea with sugar" (Table 7), where SciMK outperformed the other methods. From a general point of view, these results highlight the fact that the POMDP-based APM's performance does not only depend on the ARS error rate, but also on the NNS techniques used during evaluation (i.e., Support Vector Machine (SVM) [38,39], Euclidean distance [40], Manhattan distance [41], Correlation distance [42], k-d tree [43] and SciMK [26]).

9.2. Performance of the ERM

In Fig. 4, one can see the percentage of correct detection, false positive and false negative made by the ERM implemented in the MDP-based Task Manager and in the POMDP-based Task Manager. Before being implemented in the POMDP-based Task Manager, the ERM was trained twice with a *SimU* displaying two different behaviors (training γ and ζ). During training γ, the *SimU* could randomly choose to follow a

Algorithm 2 MC policy optimisation algorithm - ERM

Inputs:
E: set of machine's error detection states e
$Q(b, e)$: expected cost for detecting e when in b
$N(b, e)$: number of times e is selected when in b
B_e : initial set of belief states b
ε : initial policy, with $\varepsilon : B_e \rightarrow E$
repeat
  $k \leftarrow 0$
  $b = b_0$ (probability 1 to be in the initial state $s_0 = \emptyset$)
  Generate a tea trial with the Simulated User:
  repeat
    $k \leftarrow k + 1$
    Get Simulated User's ouput $a_{u,k}$ and update b.
    $e_k \leftarrow \varepsilon(b)$ or $\varepsilon(b_n)$
    with $b_n \leftarrow$ nearest neighbor of b in B_e.
    record $\langle\, b_k, e_k, c(b_k, e_k)\,\rangle$, $K \leftarrow k$
  until the Simulated User terminates the trial
  Scan trial, update B and Q
  for $r \in [0, K]$ **do**
    C $\leftarrow$ sum of the costs $c(b_r, e_r)$ from b_r to b_K.
    if $\exists b_r \in B_e$ **and** $\mid b_k - b_r \mid < \epsilon$ **then**
      $Q(b_r, a_r) \leftarrow \frac{Q(b_r,e_r)*N(b_r,e_r)+C}{N(b_r,e_r)+1}$
      $N(b_r, e_r) \leftarrow N(b_r, e_r) + 1$
    else
      create a new point b_w in B_e
      initialise $Q(b_w, e_r) \leftarrow c(b_w, e_k)$
    end if
  end for
  Update the policy
  $\varepsilon(b_r) = \arg\min_e Q(b_r, e), \forall b_r \in B_e, \forall e \in E.$
until converged

Fig. 3 – Algorithm for error recognition under uncertainty.

Table 2 – Virtual ARS confusion matrix. ARS error rate 10%.

	FillK	BoilW	AddTB	PourK	AddSu	AddMi	StirT	RemTB	PourW	ToyBW	NoAct
FillK	**90**	0	0	0	0	0	0	0	0	10	0
BoilW	0	**90**	0	0	0	0	0	0	0	10	0
AddTB	10	0	**90**	0	0	0	0	0	0	0	0
PourK	0	0	0	**90**	0	0	0	0	0	10	0
AddSu	0	0	10	0	**90**	0	0	0	0	0	0
AddMi	10	0	0	0	0	**90**	0	0	0	0	0
StirT	0	0	10	0	0	0	**90**	0	0	0	0
RemTB	10	0	0	0	0	0	0	**90**	0	0	0
PourW	10	0	0	0	0	0	0	0	**80**	10	0
ToyBW	10	0	0	0	0	0	0	0	10	**80**	0
NoAct	0	0	0	0	0	0	0	0	0	0	**100**

predefined sequence of actions, or select actions following its intrinsic configuration and comply to the POMDP-based APM outputs when the latter were passed to it. During training ζ, the *SimU* was configured to be 0% compliant to the system's outputs; so it performs the tasks by itself. During evaluation, the simulated user defined in [22] was configured to be 100% compliant to the APM's outputs when the latters were sent to it. Contrary to the ERM implemented in the POMDP-based Task Manager, the ERM implemented in the MDP-based Task Manager is not trained; it systematically checks if any errors defined in Section 4.2 appears in the simulated user's state.

As one can see in Fig. 4, the performance of the ERM implemented in the POMDP-based Task Manager outperforms the one implemented in the MDP-based Task Manager. Indeed, at 30% ARS error rate during "Black tea", the ERM implemented in the MDP-based system correctly detected

Table 3 – Virtual ARS confusion matrix. ARS error rate 20%.

	FillK	BoilW	AddTB	PourK	AddSu	AddMi	StirT	RemTB	PourW	ToyBW	NoAct
FillK	**80**	0	0	0	0	0	0	0	10	10	0
BoilW	0	**80**	0	0	0	0	0	0	10	10	0
AddTB	20	0	**80**	0	0	0	0	0	0	0	0
PourK	0	0	0	**80**	0	0	0	0	10	10	0
AddSu	0	0	10	0	**80**	0	0	0	0	10	0
AddMi	10	0	0	0	0	**80**	0	0	0	10	0
StirT	0	0	10	0	10	0	**80**	0	0	0	0
RemTB	10	0	0	0	10	0	0	**80**	0	0	0
PourW	10	0	0	0	0	0	0	0	**70**	20	0
ToyBW	10	0	0	0	0	0	0	0	20	**70**	0
NoAct	0	0	0	0	0	0	0	0	0	0	**100**

Table 4 – Virtual ARS confusion matrix. ARS error rate 30%.

	FillK	BoilW	AddTB	PourK	AddSu	AddMi	StirT	RemTB	PourW	ToyBW	NoAct
FillK	**70**	0	0	0	0	0	0	10	10	10	0
BoilW	0	**70**	0	0	0	0	0	10	10	10	0
AddTB	30	0	**70**	0	0	0	0	0	0	0	0
PourK	0	0	0	**70**	0	0	0	0	15	15	0
AddSu	0	0	15	0	**70**	0	0	0	0	15	0
AddMi	10	0	0	0	0	**70**	10	0	0	10	0
StirT	0	0	15	0	15	0	**70**	0	0	0	0
RemTB	10	0	0	0	10	10	0	**70**	0	0	0
PourW	0	0	10	0	0	0	0	0	**60**	30	0
ToyBW	0	0	10	0	0	0	0	0	30	**60**	0
NoAct	0	0	0	0	0	0	0	0	0	0	**100**

Table 5 – SimU success rate at varying ARS error rate and NNS techniques. ARS confusion matrix is observable. Task: "Black tea".

	SimU success rate			
0% compliance	78%	78%	78%	78%
MDP	**100%**	82%	62%	43%
Euclidean	**100%**	**100%**	**100%**	96%
SVM	**100%**	**100%**	**100%**	**100%**
KD-tree	**100%**	**100%**	**100%**	97%
Manhattan	**100%**	**100%**	**100%**	96%
Correlation	**100%**	**100%**	**100%**	**100%**
SciMK	**100%**	**100%**	**100%**	**100%**
ARS error rate	0%	10%	20%	30%

Table 6 – SimU success rate at varying ARS error rate and NNS techniques. ARS confusion matrix is observable. Task: "Black tea with sugar".

	SimU success rate			
0% compliance	21%	21%	21%	21%
MDP	**100%**	71%	50%	26%
Euclidean	**100%**	**100%**	82%	55%
SVM	**100%**	**100%**	**100%**	74%
KD-tree	**100%**	**100%**	74%	51%
Manhattan	**100%**	99%	97%	66%
Correlation	**100%**	**100%**	**100%**	65%
SciMK	**100%**	**100%**	**100%**	**83%**
ARS error rate	0%	10%	20%	30%

when *SimU* made errors or not 59% of the time, while the ERM implemented in the POMDP-based Task Manager was able to correctly detect True positive and True negative around 97%

Table 7 – SimU success rate at varying ARS error rate and NNS techniques. ARS confusion matrix is observable. Task: "White tea with sugar".

	SimU success rate			
0% compliance	10%	10%	10%	10%
MDP	97%	70%	43%	23%
Euclidean	**99%**	77%	63%	40%
SVM	**99%**	64%	43%	20%
KD-tree	**99%**	77%	57%	39%
Manhattan	**99%**	80%	64%	37%
Correlation	**99%**	84%	70%	40%
SciMK	**99%**	**90%**	**72%**	**46%**
ARS error rate	0%	10%	20%	30%

of the time when trained following training γ and 96% when trained following training ζ. We can note that the behavior of the *SimU* implemented to train the ERM (POMDP-based system) did not have a significant impact of its ability to detect True positives and True negatives during evaluation.

10. Discussion

- **APM performance**: The results obtained as far as the APM is concerned showed that at least 3 parameters can influence the ability of the system to provide accurate prompts to users:
  - **ARS error rate**: When the ARS error rate increases, the ARS tends to send wrong observations to the Task Manager (i.e., observations that do not correspond to

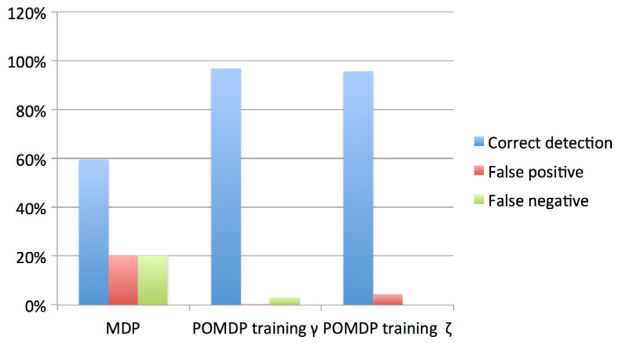

Fig. 4 – ERM evaluation: percentage of correct detection, false positive and false negative. ARS error rate: 30%. Task: "Black tea".

the true user's behavior). Even thought the POMDP-based TM is theoretically modelled to cope with such uncertainties, if the number of errors made by the ARS is too high or the level of training the POMDP-based TM went through was insufficient, the number of erroneous prompts output by the system will tend to increase.

- **Task complexity**: The ability of the POMDP-based TM to output wrong prompts is also linked to the complexity of the task. For example, during a task such as "White tea with sugar", each belief state is a probability distribution over 1539 states (i.e., each time the APM receives an observation, it may believe that the user is potentially in 1539 different states at the same time). By contrast, during "Black tea", each belief state is a probability distribution over 33 states only. It is known that one of the difficulties in solving POMDPs can be attributed to the curse of dimensionality [44,45]. The curse of dimensionality refers to the increase in complexity because of the number of hidden states.
- **NNS technique**: Depending on the NNS technique implemented, the POMDP-based APM will cope differently with the difficulties created by the ARS error rate and task complexity. Results highlighted that in all cases, "SciMK" outperformed other techniques, eventhought the system's performance tend to decrease at increasing ARS error rates and task complexity.

- **ERM performance**: Contrary to the ERM implemented in the MDP-based Task Manager, the ERM implemented in the POMDP-based Task Manager can only detect if an error has potentially be made by users or not; it currently cannot detect which type of error (i.e., see errors types in 4.2) could have been made. Hence, the results presented in Fig. 4 do not relate to error types, but highlight the ability of the ERM in the MDP and POMDP-based systems to correclty detect errors in general. As part of future work the training of the ERM implemented in the POMDP-based TM will have to be improved in order to detect error types under uncertainty. Indeed, this is an important feature to enable as it can allow the system to detect when the task needs to be stopped to prevent safety issues (see "Fatal errors" in 4.2).

11. Conclusion

As expected, results showed that most of the time, the POMDP-based system has a better ability to cope with uncertainty in its inputs than the MDP-based system. Thus, the focus was put on the impact of Nearest Neighbor Search techniques on the POMDP-based system performance. We found that the novel algorithm presented in this paper constantly outperformed the other methods it has been compared with. This novel algorithm referred to as "SciMK" allowed the POMDP-based Task Manager to select appropriate actions more often than other methods, at increasing ARS error rate and tasks complexity. In other words, SciMK helps the user succeed the task more often. A novel algorithm for human's error detection was also proposed and results showed that it could successfully detect when a user made errors or not under uncertainty.

In the future, it would be interesting to improve the training of the ERM so it can detect error types, and compare the performance of the CogWatch Task Manager with other assistive systems' planning system (for example, COACH [16], TEBRA [20]). Currently, this is not practically feasible. Although CogWatch, TEBRA and COACH focus on sequential activities of daily living, all have been implemented to model different tasks; and the performance of these systems during these tasks have been assessed in different ways by their authors. Moreover, the Task Manager performance highly depends on the system's ARS error rate, which is also different in all systems. The development of a common framework will make it possible to have a better understanding of the strengths and limitations of assistive systems for cognition, and help in the improvement of novel techniques that could be beneficial in the field.

REFERENCES

[1] S. Association, The Stroke Association website, 2014. URL http://www.stroke.org.uk/about-stroke.
[2] I.S.W. Party, Royal college of physicians national sentinel stroke clinical audit 2010 round 7 public report for england, wales and northern ireland., Tech. Rep., 2010.
[3] A. Zwinkels, C. Geusgens, P. van de Sande, C.V. Heugten, Assessment of apraxia: inter-rater reliability of a new apraxia test, association between apraxia and other cognitive deficits and prevalence of apraxia in a rehabilitation setting, Clin. Rehabil. (2004) 819–827.
[4] C. Barbieri, E.D. Renzi, The executive and ideational components of apraxia, Cortex (1988) 535–543.
[5] W. Bickerton, M. Riddoch, D. Samson, A. Balani, B. Mistry, G. Humphreys, Systematic assessment of apraxia and functional predictions from the birmingham cognitive screen, J. Neurol. Neurosurg. Psychiatry (2012) 513–521.
[6] B. Petreska, M. Adriani, O. Blanke, A.G. Billard, Apraxia: a review: From action to cognition, Prog. Brain Res. (2007) 61–83.
[7] G. Goldenberg, M. Daumuller, S. Hagmann, Assessment and therapy of complex activities of daily living in apraxia, Neuropsychol. Rehabil. (2001) 147–169.
[8] K. Morady, G. Humphreys, Comparing action disorganization syndrome and dual-task load on normal performance in everyday action tasks, Neurocase (2009) 1–12.
[9] S.U.T. Collaboration, Organised inpatient (stroke unit) care for stroke, Cochrane Database Sys. (2007) 1.

[10] P. Langhorne, L. Legg, Evidence behind stroke rehabilitation, J. Neurol. Neurosurg. Psychiatry (2003) iv18–iv21.
[11] A.D. Carlo, Human and economic burden of stroke, Age Ageing (2009) 4–5.
[12] I. Proot, H. Crebolder, H. Abu-Saad, T. Macor, R.T. Meulen, Facilitating and constraining factors on autonomy: The views of stroke patients on admission into nursing homes, Clin. Nurs. Res. (2000) 460–478.
[13] E.F. LoPresti, A. Mihailidis, N. Kirsch, Assistive technology for cognitive rehabilitation: State of the art, Neuropsychol. Rehabil. (2004) 5–39.
[14] A. Gillespie, C. Best, B. O'Neill, Cognitive function and assistive technology for cognition: A systematic review, Int. Neuropsychol. Soc. (2011) 1–19.
[15] T. Hart, T. O'Neil-Pirozzi, C. Morita, Clinician expectations for portable electronic devices as cognitive-behavioural orthoses in traumatic brain injury rehabilitation, Brain Injury 17 (5) (2003) 401–411.
[16] J. Boger, J. Hoey, P. Poupart, C. Boutilier, G. Fernie, A. Mihailidis, A planning system based on markov decision processes to guide people with dementia through activities of daily living, IEEE Trans. Inf. Technol. Biomed. (2006) 323–333.
[17] A. Mihailidis, B. Carmichael, J. Boger, The use of computer vision in an intelligent environment to support aging-in-place, safety, and independence in the home, Trans. Inf. Tech. Biomed. (2004) 238–247.
[18] J. Hoey, P. Poupart, A. von Bertoldi, T. Craig, C. Boutilier, A. Mihailidis, Automated handwashing assistance for persons with dementia using video and a partially observable markov decision process, Comput. Vis. Image Underst. (2010) 503–519.
[19] J. Mihailidis, J.N. Boger, T. Craig, J. Hoey, The coach prompting system to assist older adults with dementia through handwashing: An efficacy study, BMC Geriatr. (2008) 28.
[20] C. Peters, T. Hermann, S. Wachsmuth, TEBRA - an automatic prompting system for persons with cognitive disabilities in brushing teeth, in: Proceedings of the 6th International Conference on Health Informatics, HealthInf, 2013, pp. 12–23.
[21] C. Peters, T. Hermann, S. Wachsmuth, J. Hoey, Automatic task assistance for people with cognitive disabilities in brushing teeth - a user study with the tebra system, ACM Trans. Accessible Comput. (2014) 10:1–10:34.
[22] E. Jean-Baptiste, R. Nabiei, M. Parekh, E. Fringi, B. Drozdowska, C. Baber, P. Jancovic, P. Rotshein, M. Russell, Intelligent assistive system using real-time action recognition for stroke survivors, in: Proceedings of the IEEE International Conference on Healthcare Informatics, ICHI, 2014.
[23] J. Hermsdörfer, M. Bienkiewicz, M. Cogollor, J. Russell, E. Jean-Baptiste, M. Parekh, A. Wing, M. Ferre, C. Hugues, Cogwatch automated assistance and rehabilitation of stroke-induced action disorders in the home environment, in: Proc. of HCI Aspects of Optimal Healing Environments, 2013.
[24] E. Jean-Baptiste, J. Howe, P. Rotshtein, M. Russell, Cogwatch: Intelligent agent-based system to assist stroke survivors during tea-making, in: Proceedings of the IEEE/SAI Intelligent Systems Conference, 2015.
[25] E. Jean-Baptiste, M. Russell, P. Rothstein, Assistive system for people with apraxia using a Markov decision process, in: MIE, Studies in Health Technology and Informatics, 2014, pp. 687–691.
[26] E. Jean-Baptiste, P. Rotshein, M. Russell, POMDP based action planning and human error detection, in: Proceedings of the 11th International Conference on Artificial Intelligence Applications and Innovations, 2015.
[27] R. Nabiei, M. Parekh, E. Jean-Baptiste, P. Jančovič, M. Russell, Object-centred recognition of human activity for assistance and rehabilitation of stroke patients, in: IEEE Int. Conf. On Healthcare Informatics, 2015.
[28] L. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, in: Readings in Speech Recognition, 1990, pp. 267–296.
[29] M.F. Schwartz, E. Reed, M. Montgomery, C. Palmer, N. Mayer, The quantitative description of action disorganisation brain damage: A case study, Cogn. Neuropsychol. (1991) 381–414.
[30] J. Williams, Applying pomdps to dialog systems in the troubleshooting domain., in: Proc HLT/NAACL Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technology, Rochester, NY, USA, 2007.
[31] B. Thomsonn, S. Young, Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems, Comput. Speech Lang. (2010) 562–588.
[32] S. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, K. Yu, The hidden information state model: a practical framework for pomdp-based spoken dialogue management, Comput. Speech Lang. (2010) 150–174.
[33] W. Turkett, Robust multiagent plan generation and execution with decision theoretic planners (Ph.D. thesis), University of South Carolina, 1998.
[34] R. Brafman, A heuristic variable grid solution method for POMDPs, in: Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence, 1997.
[35] K.Q. Weinberger, L.K.S. Lawrence, Distance metric learning for large margin nearest neighbor classification, J. Mach. Learn. Res. (2009) 207–244.
[36] A. Hinneburg, C.C. Aggarwal, D. Keim, What is the nearest neighbor in high dimensional spaces?, in: Proceedings of the 26th International Conference on Very Large Data Bases, 2000.
[37] C.C. Aggarwal, A. Hinneburg, D.A. Keim, On the surprising behavior of distance metrics in high dimensional space, 2001.
[38] V. Vapnik, Statistical Learning Theory, 1998.
[39] Y. Lee, Y. Lin, G. Wahba, Multicategory support vector machines, J. Amer. Statist. Assoc. (2001) 67–81.
[40] M. Abbasifard, B. Ghahremani, H. Naderi, Article: A survey on nearest neighbor search methods, Int. J. Comput. Appl. (2014) 39–52.
[41] E. Krause, Taxicab Geometry: An Adventure in Non-Euclidean Geometry, Dover Publ., 1987.
[42] K. Pearson, Notes on the history of correlation, Biometrika (1920) 25–45.
[43] J. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (9) (1975) 509–517.
[44] J. Pineau, G. Gordon, T. Sebastian, Point-based value iteration: An anytime algorithm for POMDPs, in: Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI'03, 2003.
[45] L.P. Kaelbling, M.L. Littman, A.R. Cassandra, Planning and acting in partially observable stochastic domains, Artificial Intelligence (1998) 99–134.