ICT-2009.7.1
KSERA Project
2010-248085

Deliverable D2.5

Navigation based on context awareness and intention reading

30th September 2012
Public Document

The KSERA project (http://www.ksera-project.eu) has received funding from the European Commission under the 7th Framework Programme (FP7) for Research and Technological Development under grant agreement n°2010-248085.

Project acronym:	KSERA
Project full title:	Knowledgeable SErvice Robots for Aging
Work package:	2
Document number:	D2.5
Document title:	Navigation based on context awareness and intention reading
Version:	0
Delivery date:	30 September 2012 (month 32)
Actual publication date:	8 October 2012
Dissemination level:	Public
Nature:	Report
Editor(s) / lead beneficiary:	Stefan Wermter (UH)
Authors(s):	Cornelius Weber (UH), Wenjie Yan (UH), Elena Torta (TU/e)
Reviewer(s):	Nils Meins

Contents

Executive summary 5
Purpose of this deliverable 6
Suggested readers 6
Relationship to other documents 6
1 Introduction 7
2 Context awareness 8
2.1 Person/robot localization using ceiling-mounted camera 8
2.2 Robot navigation based on map learning 9
2.2.1 Spatial memory layer 10
2.2.2 Action memory model 11
2.2.3 Action layer 11
2.2.4 Navigation planning 12
2.2.5 Experiments 14
2.3 Robot navigation based on behaviour control 15
2.4 Robot mobile behaviour based on ubiquitous monitoring and user context 18
3 Intention reading: predicting user movement goals 19
3.1 Hierarchical Hidden Markov Model 21
3.2 Experiments 22
3.2.1 Test Method 22
3.2.2 Results 23
3.2.3 Discussion and Conclusions 27
4 Intention reading: user experience 28
4.1 Method 28
4.1.1 Experimental Setup 28
4.1.2 Participants and task 29
4.1.3 Design 29
4.1.4 Procedure 29
4.1.5 Results 30
4.1.6 Conclusion 32
5 Summary and outlook 33
Appendix 36
A. Hidden Markov Model 36
Definitions 36
Training 36
Prediction 36
Parameter updating 37

B. Hierarchical Hidden Markov Model....38
Definitions....38
Training....38
Prediction....38

Glossary of terms used

Acronym	Definition
AAL	Ambient Assisted Living
HRI	Human Robot Interaction
ROS	Robot Operating System, a middleware
SMACH	an extended State MACHine
PT1	Prototype 1
PT2	Prototype 2

Executive summary

This report describes the current research progress of robot localization and navigation based on context awareness and user intention reading. The KSERA project develops new technologies for a small assistive robot that serves as a unique interface between the ambient assisted living (AAL) system and the person and provides contact to remote care givers. The system's awareness of the environment as well as the locations of the robot and the person is crucial to realize flexible robot navigation ability and to achieve essential care-giving functionalities.

We present our improved person- and robot localization model using vision inputs from a ceiling-mounted camera. An additional vision cue using the TLD algorithm is added to the person localization model to obtain a more robust tracking behaviour, particularly while the user is sitting. Based on this location information, a robot can execute navigation tasks such as approaching the person, walking back to the home/projector position, etc. In order to accomplish different navigation phases, two navigation methods are developed: a *map-based approach* that learns the traversable space as well as the appearance of the environment, and a *behaviour-based approach* through which the robot respects the presence of the user. We demonstrated and tested our methods and results show that robust navigation behaviour can be achieved.

Purpose of this deliverable

This deliverable describes the research progress of robot navigation behaviour based on context awareness and intention reading. It describes experimental tests that have been conducted for the developed methods and parts of them that have been integrated in the KSERA system for the field trials.

Suggested readers

This document is recommended to all KSERA partners. Furthermore, it may be helpful to external researchers involved in developing mobile care- and service robots in ambient assisted living scenarios.

Relationship to other documents

The robot mobile behaviour of the final prototype is a further development of the robot mobile behaviour of the first prototype, described in D2.3. Both build on the concept of context awareness for navigation, described in D2.2. It takes into consideration the formative evaluation described in D5.3 and the scenarios added to D1.1. The robot mobile behaviour of the final prototype is a significant component of the whole KSERA system that is coordinated by the rule engine software, SMACH. Hence, the description of this prototype makes reference to the rule engine, which is described in detail in the overall KSERA prototype in D5.4.

1 Introduction

This deliverable reports the research advances of the robot navigation behaviour based on context awareness and user intention reading. As one of the key components of the KSERA system, a service assistive robot is designed as a unique interface between the automated assistive-living (AAL) system and the person, therefore the robot mobile behaviour is crucial to realize the functionalities of the system such as demonstrating exercises, providing contact to remote care givers, etc. We focus on investigating the role of context awareness within the robot navigation tasks and the influence of the intention reading on the user acceptance. Towards this end, we developed corresponding navigation methods based on environment learning and interaction with the user and the AAL system.

A number of projects and research have addressed issues related to context awareness and intention reading for robotic navigation in intelligent homes [1], [2]. Context awareness is a broad concept especially when referred to conditions in which a robot shares an intelligent environment with a human. In the KSERA project we refer to it as the ability of the intelligent robot/system to perceive the environment, infer the context from environmental data (i.e. the state of the user or the disposition of objects in the environment) and compute appropriate navigation actions based on the detected context.

Such a context-aware navigation system requires the development of algorithms for environmental sensing and environmental interaction. Therefore, based on the feedback of the first field trials, we improved our person tracking model of the PT1 by adding an extra vision cue using the TLD algorithm [3] so that it could track a person more robustly while the person is sitting, and further developed a navigation model based on cognitive map learning and a dynamic neural field model. The neural field model provides a sub-symbolic form of object representation and a layer for behaviour coordination

In addition, the interpretation of data gathered from the user or from the outside world is essential to trigger the appropriate navigation behaviour. The appropriate navigation behaviour should be executed based on the awareness of the state of the user (i.e. the user utters a message or performs a measurement), or the state of the outside world (i.e. incoming video connection). An intelligent server is designed in the KSERA system to coordinate the robot mobile behaviour with the user context and the ubiquitous monitoring.

With intention reading we refer to the task of recognizing the intentions of a human in the intelligent home by analyzing his actions and/or the effects of his actions in the environment [4]. This is related to the robot mobile behaviour because if a robot knows a person's intentions it can anticipate the person's destination and move to an appropriate location in the appropriate time window [5]. We investigated how to determine user’s intention based on data that covers larger spaces, considering occupancy of different rooms in a house from real COPD patients, and found encouraging results. In the KSERA project we also worked on investigating how people perceive different robot’s anticipatory behaviours based on intention reading. The results of our preliminary user study suggest that, if the scenario is limited to a single room, users tend not to distinguish between anticipatory navigation behaviour of the Nao robot and non-anticipatory behaviour, but their perception of the robot is rather influenced by the current interaction type and use case scenario.

This deliverable is organized as follows: The section “Context Awareness” describes the concept of the robot localization and navigation modules including the improved person/robot localization model using a ceiling-mounted camera, and the robot navigation methods including the behaviour-based navigation method and the map-based navigation method. In section “Intention Reading” we investigate the relationship between the anticipatory behaviour of the robot and the perception of

the person, and we investigate the general possibility to anticipate a person's behaviour. Section "Summary and outlook" concludes this report and highlights the future view of our research.

2 Context awareness

This section provides an overview of the technologies we developed for person/robot localization, robot navigation based on map learning and robot navigation based on behaviour control. We introduce also briefly the decision making of the rule engine based on the inputs from the ubiquitous monitoring.

2.1 Person/robot localization using ceiling-mounted camera

The localization module in PT2 is responsible for localizing a person and a robot with the input from the ceiling-mounted camera while the person is alone in the room. This module consists of two parts: person localization with a hybrid probabilistic model and robot localization based on particle filter prediction. As **Figure 1** shows, the person localization is designed for tracking a single person since it uses only a single particle filter. This restriction is justified by the logic that the KSERA system is not required when a visitor or care giver is visiting the person. In the current stage the localization module can distinguish one tracked person and the Nao robot. In the future it would be desirable to extend it to distinguish different persons using computer vision technology. This procedure is considered as the input to the navigation module. The scientific methods and the technical details are described in the publications [6], [7], [8].

Considering the feedback from the evaluation of the PT1, the person localization model is improved by adding a channel that implements the tracking-learning-detection (TLD) algorithm [3]. TLD is a tracking algorithm that learns the shape of the object during tracking. It provides a robust tracking performance when the user sits or moves slowly in the room. As shown in **Figure 2**, the left window shows the detection of the TLD algorithm (labelled in a bounding box), and the window named "Localization" shows the final localization position. During the tracking, the features of the target person are learned autonomously by the TLD that are listed on the thin column of small images shown on the right side. Although a distracting person is moving in the room, the algorithm can still track the target person robustly when he is sitting. Despite this performance in the laboratory setting, in the field trials the localisation is tested only with a single person in the room.

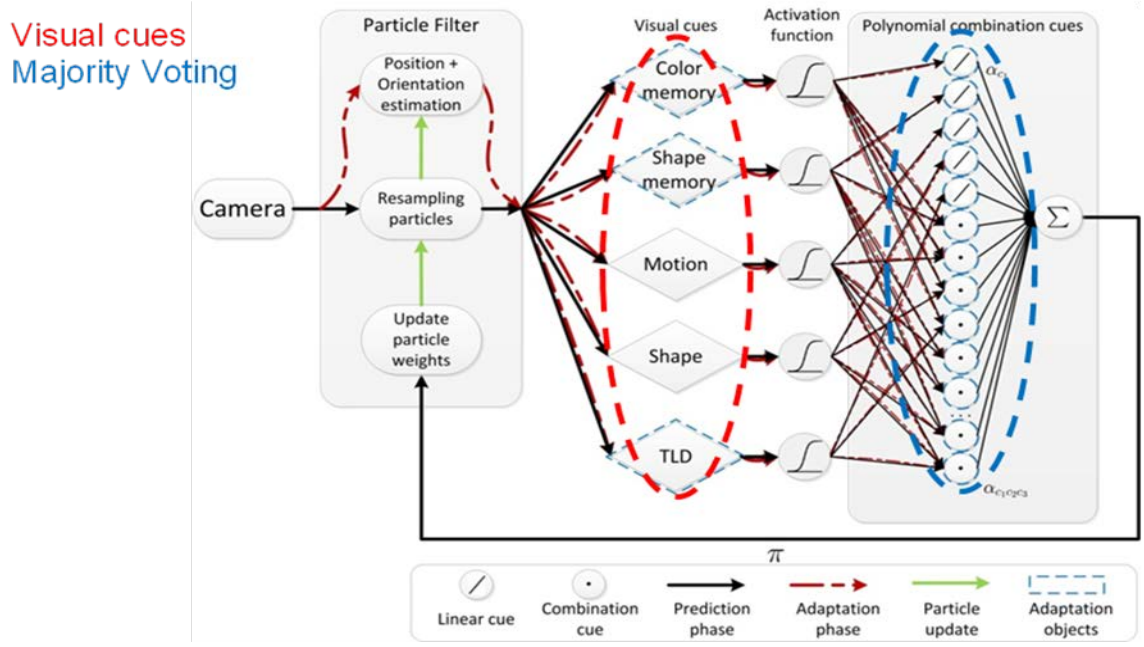

Figure 1. Schema of the person localization model.

The robot localization module takes advantage of the predictive capabilities of a particle filter. Since we know precisely which actions the robot executes, we build up a feed-forward model that simulates the robot's physical motion behaviour. The localization program can not only detect the position of the robot, but also estimate the walking orientation. An example of the robot localization is shown in **Figure 3** right, where the robot's position is localized by the blue particles and the orientation is estimated by a yellow short line.

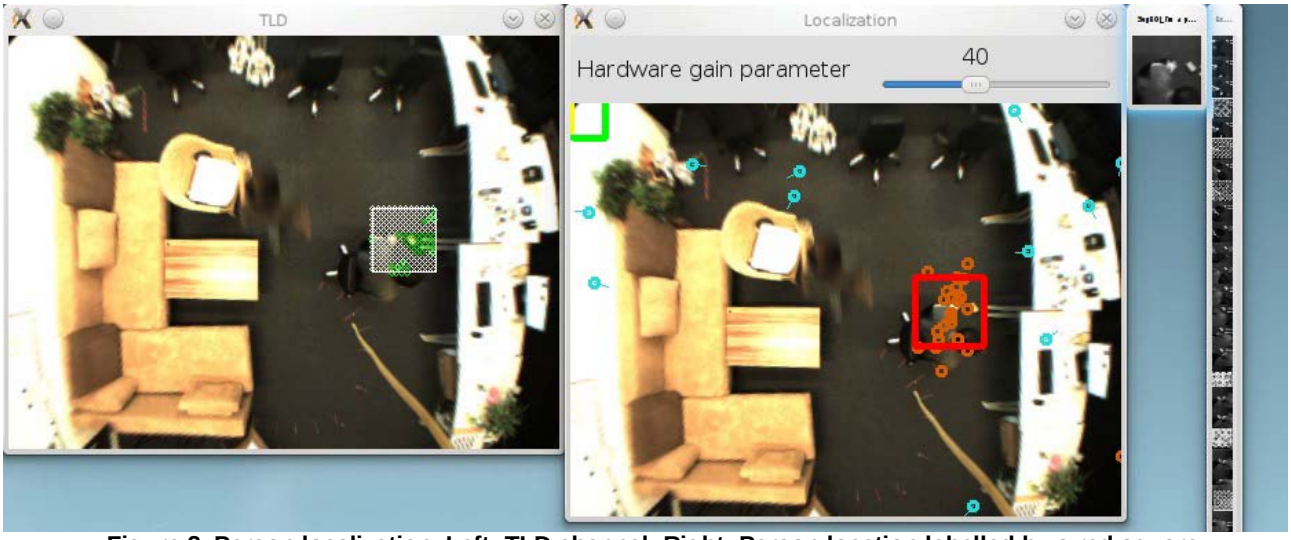

Figure 2. Person localization. Left: TLD channel; Right: Person location labelled by a red square

Figure 3. Robot localization. Left: Using marker in PT1; Right: Using particle filter without marker in PT2

2.2 Robot navigation based on map learning

Based on the output of the localisation module, a cognitive map [7] of the environment is built for robot navigation. A cognitive map is a topological representation of traversable areas, which can be built through the exploratory behaviour of the robot and complemented based on the observed navigation behaviour of a person. Considering that a cognitive map can represent the topological information [9] and the appearance of the environment, we extend this model by combining the information from the ceiling-mounted camera with the visual input of the robot's camera. Salient visual features detected by the robot's camera, while performing exploratory navigation, are linked to the current robot's location and stored in the cognitive map. Thus, a sensorimotor representation of the environment is built up that encodes not only the topological structure of traversable areas

(where), but also the appearance of a location perceived by the robot (what) which helps to increase the robustness of the estimation of the robot orientation

The navigation system consists of three components: *1)* a spatial memory layer that learns the structure and the appearance of the environment, *2)* an action memory model that learns an inverse control model and *3)* an action layer for robot control. These are shown in **Figure 4** and described in detail in the following.

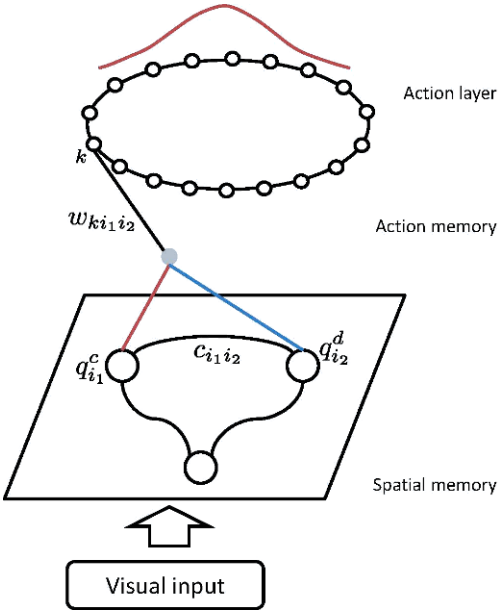

Figure 4. Schema of the map-based navigation model. The red line indicates a connection from the current robot position to an action memory unit. The blue line indicates a connection from the robot's goal to an action memory unit.

2.2.1 Spatial memory layer

The ***spatial memory layer*** presents the spatial knowledge and the spatial relations in a cognitive map. When a person or a robot visits a novel location, features of this location as well as the relation with other places will be stored. Our research here is based on Marsland's Growing When Required (GWR) network [10] to present the spatial information. Compared with the Growing Neural Gas model [11] that we used in [9], GWR does not grow over time but only when novelty is detected, which provides better convergence properties.

The spatial memory layer consists of a set A of neurons, each associated with feature vectors v, and a set N of connections to describe the relations (i_1, i_2) between neurons $i_1,\ i_2 \in A$. Different features can be presented in the neurons. In our case, each neuron contains the x, y coordinate information ("where") on an image from the ceiling-mounted camera $v^c = \{x, y\}$ and the visual keypoints ("what") extracted from the robot's camera $v^r = \{k_1, k_2, \dots\}$. We use SURF features [12] to present the information of keypoints (see **Figure 6**). Each keypoint contains the x, y position of the feature point and a 32-dimensional vector that represents the image gradients.

The closer a neuron is to the target person for navigation, the larger activity this neuron will have. We computed the proximity of the current visual keypoints extracted from the robot camera to the stored visual keypoints, towards using them as a further cue for the visual tracking system, which will increase the robustness of the robot localization, particularly its orientation. Each neuron is assigned a firing counter h to control the adaptation efficiency. Using a competitive Hebbian learning rule, neurons and connections $c_{i_1 i_2}$ will be allocated or updated dynamically, and will be

deleted if the connection age is over a threshold $a_{\max}$ or if a neuron is isolated. The robot's camera captures vision information during the exploratory navigation. The neuron i^* closest to the robot's location will be active in this case and the visual features extracted from the robot's camera will be assigned to $v^r_{i^*}$:

$$\begin{aligned} k_{\text{new}}\{\cdot\} &= \text{Extract_Features}() \\ v^r_{i^*} &\leftarrow k_{\text{new}} \end{aligned} \quad (1)$$

The map can be built just by observing a person's movement from the ceiling-mounted camera. However, it might not be completely accurate for a robot due to the different walking behaviour of a person and a robot, and also the objects in the environment may change their positions after map building. Therefore, a robot-environment interaction is essential for the robot to adapt online its navigation strategy. We define a connection weight $c_{i_1 i_2} \in [0, 1]$ for each connection (i_1, i_2) to indicate how "easy" a robot can move along this connection. The higher $c_{i_1 i_2}$, the easier this connection is for the robot to walk along. When an obstacle is detected or the robot has difficulties walking further, $c_{i_1 i_2}$ will be decreased and may reach zero. When a connection is built, its connection weight $c_{i_1 i_2}$ will be initialized to 1 and adapted during the robot navigation. For details about map building we refer to [9] and [10].

2.2.2 Action memory model

The ***action memory model*** learns the action information coupled with the state transmission in the spatial memory layer. We use Sigma-Pi weights [13] to store the action information associated with the state transfer in the spatial memory. When the robot moves from i_1 to i_2 in the GWR, represented by saying $i_1 \rightarrow i_2$, the corresponding Sigma-Pi weights will be activated with the input signal $q^c_{i_1}$ of the current state i_1 and $q^d_{i_2}$ of the desired next state i_2 (described in [9]). The action information is learned in the weights $w_{ki_1i_2}$ (see **Figure 4**), which connect fully with the neurons k in the action layer. If the neuron i_1 of the current state and the neuron i_2 of the desired state are active, the output towards unit k in the action layer will be computed as follows:

$$I_k = \sum_{i_1, i_2} w_{ki_1i_2} q^c_{i_1} q^d_{i_2} \quad (2)$$

Because multiple action patterns may be triggered at the same time, Eq.(2) merges the different signals and generates a combined action signal to the action layer.

Depending on the way the action layer controls the robot, the action information can be presented in different forms, for example force, velocity, angle value, etc. In our case, the robot has a fixed walking speed and only its orientation is adjusted using a ring-form dynamic neural field with 36 nodes, which will be described in the next paragraph. The action pattern is therefore encoded in $w_{ki_1i_2}$ where $k \in \{1, 2, \ldots, 36\}$. Although the total number of weights $w_{ki_1i_2}$ is $36n^2$, which grows quadratically according to the number n of neurons in the spatial memory layer, most of the weights are zero since the neurons in the GWR are sparsely connected with each other. Because only the connection weights of the output layer that connect to the dynamic neural field need to be trained (i.e. the $w_{ki_1i_2}$), the action code can be learned online based on the sensory input. In our study we train the weights by calculating the desired direction of the robot based on its coordinates in the image of the ceiling-mounted camera. Further details of training the weights $w_{ki_1i_2}$ are described in [9].

2.2.3 Action layer

The ***action layer*** controls the mobile behaviour of the robot based on the active action memory during navigation. A dynamic neural field (DNF) model is used in our algorithm to simulate the

heading direction cells for adjusting the robot's walking orientation in the action layer. The DNF is a biologically-inspired model of the neural dynamics in cortical tissues [14], which is of interest in the robotic area to generate dynamic behaviour [15], [16]. A one-dimensional ring-form DNF with 36 neurons is implemented in our work to control the robot's navigation behaviour over time. The DNF is capable of integrating the multiple action memory signals and adjusts the robot's motion with a smooth and natural orienting behaviour. Each neuron k of the DNF has a membrane potential u_k that represents the activity, and lateral connections n_{kj} with other neighbour neurons j. Through the following updating rule, the DNF dynamically generates an activation bump (indicated in **Figure 4** by a red curve) that represents the suggested orientation for the next step:

$$\tau \Delta u_k = -u_k + \sum_{j=1}^{36} n_{kj} f(u_j) + I_k + h \tag{3}$$

where h is a rest potential, τ is a temporal decay rate of the membrane potential, and I_k is the input stimulus of the k-th neuron received from the Sigma-Pi network that encodes the desired robot orientation.

We use here a Gaussian function with negative offset as the function to describe the lateral connection strengthsn_{kj}:

$$n_{kj} = \beta e^{-\frac{(k-j)^2}{2\sigma^2}} - c \tag{4}$$

where β is a scaling factor, σ^2 a variance, k, j the positions of neurons and c is a positive constant. Because of the annular topology of the 36 DNF neurons, $|k-j|$ can never become larger than18. The function $f(u)$ is a sigmoid transfer function of a single neuron:

$$f(u_j) = \frac{1}{1+e^{-u_j}}, \tag{5}$$

The robot's desired orientation is determined by the peak of the emerging activation bump of u_k (for details see [9]).

2.2.4 Navigation planning

The ***navigation planning*** can be processed as follows based on the learned cognitive map. As described in Eq.(2), the control signal of the DNF network is computed based on the connection weights w, the activity of the current state q^c and the activity of the next desired state q^d. Because the connection weights w are trained during map building and q^c can be computed with respect to the robot's position, the next desired state q^d is computed according to positions of the robot and the target person/location. We assign for each neuron of the GWR a reward signal spreading from the target states representations iteratively with an exponential decrease. Depending on the navigation target, an initial reward signal $r_i(0)$ is determined using the following steps:

1. Calculate the input signals m_i of the neuron i in the GWR. For approaching the target or location (e.g. charging station, beaming position) we calculate the signals based on the distance between the target's position and the neuron's coordinate information v^c:

$$m_i = e^{-\frac{||v_i^c - \xi||^2}{2\sigma^2}}, \tag{6}$$

   where ξ is the position computed from the localization model [7].

2. Normalize the input signals with a softmax function:

$$m_i = \frac{e^{m_i}}{\sum_i e^{m_i}}, \tag{7}$$

3. Assign m_i to initial reward signals with a threshold filter:

$$r_i^p(0) = \begin{cases} r_0 m_i, & \text{if } m_i > \text{threshold}_m, \\ 0, & \text{else} \end{cases} \tag{8}$$

where p indices the neuron of the initial reward signal. Multiple units may contribute to localize the target object and the person since the person's location is presented with a probabilistic distribution and the object may be observed from different positions.

For each $r_i^p(0) > 0$, the reward signals will spread separately to the neighbour neurons (listed in n) iteratively with a discount factor λ:

$$r_j^p(t+1) = \lambda r_i^p(t), \quad \text{for } j \in n \text{ and } r_j^p(t) = 0 \tag{9}$$

where the neighbourhood list n will be updated for each iteration as follows:

$$\begin{aligned} n' &\leftarrow i \quad \text{if } i \text{ connects with a neuron in } n \\ &\qquad\quad \text{and } r_i^p(t+1) = 0 \\ n &= n' \end{aligned} \tag{10}$$

After the spreading phase, the final signal r_j of each neuron $\{j\}$ will be calculated by summing up all the reward signals from the target position:

$$r_j = \sum_p r_j^p \tag{11}$$

As **Figure 5** shows, the reward signals of the GWR are visualized with the brightness of neurons. The closer a neuron is to the target position, the brighter this neuron is displayed.
After the reward signal has spread over the entire GWR network, the robot plans its action for the next step by determining the next position it should reach. We assume that the robot's position is represented by a group of neurons α in the GWR, the next possible position should be among the neighbourhood neurons that connect to neurons in α directly. The activity $q_{i_2}^d$ of these neighbourhood neurons i_2, which connect with neurons $i_1 \in \alpha$, is computed as follows:

$$\widetilde{q_{i_2}^d} = \sum_{i_1 \in \alpha} c_{i_1 i_2} s_{i_1}^r r_{i_2} \tag{12}$$

and normalized with a soft-max activation function:

$$q_{i_2}^d = \frac{e^{\widetilde{q_{i_2}^d}}}{\sum_k e^{\widetilde{q_k^d}}} \tag{13}$$

where $s_{i_1}^r$ is neuron activity of the robot detection and $c_{i_1 i_2}$ is the connection weight. The higher $q_{i_2}^d$ is, the more desirable it is for the robot to be at this position in the next step.

2.2.5 Experiments

We conducted experiments in a home-like laboratory to evaluate the performance of our model. In addition to the ceiling camera, we use one camera in the robot's head to observe the environment and the two pairs of sonar sensors for detecting and avoiding obstacles while walking.

2.2.5.1 Learning a cognitive map

At the beginning of the map learning, the GWR is initialized with two neurons linked with each other with a connection. Then, based on the person's position estimated by the localization model, the closest neuron to the person (see **Figure 5**, the winner neuron is in yellow) and the second closest neuron (second winner neuron in blue) will be computed (**Figure 5** left). The winner and its neighbouring neurons will be drawn to the person's position and new neurons will be inserted. The GWR will grow automatically when a person moves to a new place in the room until most of the free space has been visited (**Figure 5** right).

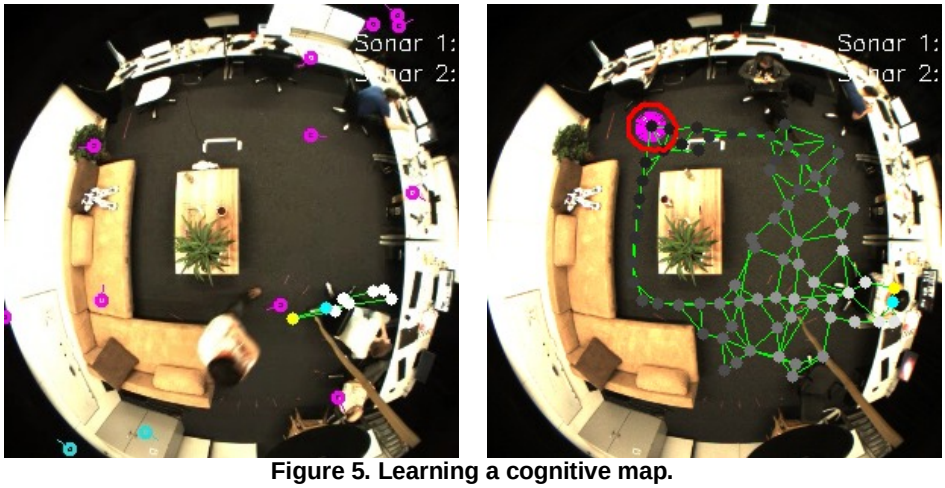

Figure 5. Learning a cognitive map.
Left, a cognitive map is being built based on a person's motion. Right, robot traversing using cognitive map information.

The robot then explores the room itself and memorizes the appearance of the environment. As shown in **Figure 6**, the visual features extracted from the robot's camera view will be registered to the corresponding neuron where the robot is located.

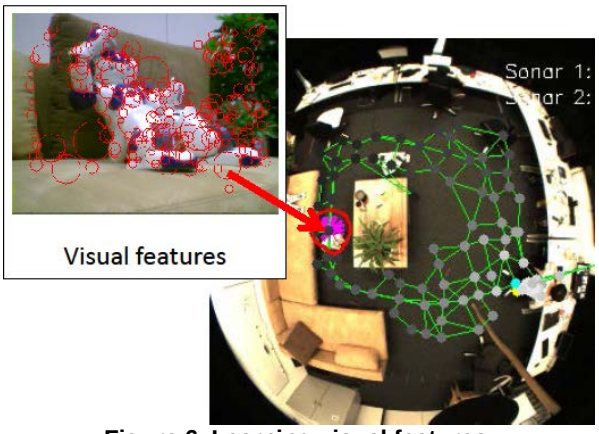

Figure 6. Learning visual features.
Red circles show SURF features detected by the robot's camera. Note that the features describe salient features of the room (sofa, plant), but also transient features of removeable objects (a Nao robot) that present noise to the localisation algorithm.

2.2.5.2 Navigation towards a person

After the map building phase, the robot can approach the user based on the cognitive map. A reward signal spreads first from the person's position through the whole network. As **Figure 6** shows,

the gray-scaled brightness of the neurons indicates the reward signal spreading from the target location (the person near the right). The brighter the colour is, the higher the reward that this neuron represents. Then, based on the current robot position and the reward signals of the cognitive map, the robot determines the next desired state and generates the motion based on the signals in the Sigma-Pi network. The motion signals are integrated activities of the DNF, indicated in **Figure 6** by a red circle surrounding the robot's position with a basic radius of 15 pixels where activation is zero. An activation bump is built up which helps the robot to estimate the desired orientation and navigate through controlling the orientation. During the robot's movement, the state representations are changing and the activation bump will be updated to the new orientation. When the robot gets close to the person, the navigation will be achieved and the robot will stop walking.

2.3 Robot navigation based on behaviour control

Complex navigation behaviours can be generated by the coordination of simpler ones. In the case of the mobile robot's navigation, each simple behaviour solves a navigational subtask without the need of high level world representation [17, 18]. Behaviour based robotics provides real-time adaptation to dynamically changing environments and can be adopted by robots with very limited sensory capabilities, such as the humanoid robot Nao used in the KSERA project. A description of the behaviour-based navigation algorithm adopted by the KSERA project can be found in D2.2 Section 4.1. Further details on the behaviour-based navigation algorithm can be found in [19, 20].

Here we extend that framework exploring the adoption of a mechanism for behaviour coordination inspired by processing principles in biological nervous systems: the Dynamic Neural Field (DNF). The DNF is a biologically-inspired recurrent neural network which represents, in a mathematical form, the dynamics of pattern formation for lateral-inhibition type homogeneous neural fields with general connections [14, 15]. In particular, it provides a form of sub-symbolic short term memory that retains the excitation pattern after the input stimulus has vanished and lateral inhibition mechanism that selects between alternative patterns of activation. Short term memory is in our navigation model influenced by the detected objects in the environment; therefore the introduction of the DNF represents a sub-symbolic form of context awareness. We realized the stage of behaviour coordination by means of the DNF, which can express constructive or selective coordination of behaviours according to the patterns of the input stimuli and the current field state.

The expression of a one-dimensional dynamic neural field, as proposed by Amari [14] and described in Erlhagen and Bicho [15], is given by:

$$\begin{aligned} \tau \dot{u}(\phi,t) &= -u(\phi,t) + S(\phi,t) \\ &\int w(\phi-\phi')g(u(\phi',t))d\phi' + h, \end{aligned} \tag{14}$$

The term $\phi(t)$ represents the domain of the neural field and in most applications is chosen to be circular. Thus if the domain is represented by $\phi \in [\phi_{min}, \phi_{max}]$ we have $\phi_{min} = \phi_{max}$. The term τ represents the field's time constant and its value determines the relaxation time of the field's dynamic. The term $S(\phi,t)$ represents the external input to the field. The term $w(\phi-\phi')$ represents a centre-surround excitation-inhibition function and is commonly modelled by a lowered Gaussian:

$$w(\phi-\phi') = A\exp\left(-\frac{(\phi-\phi')^2}{2\sigma^2}\right) - w_{inhibit} \tag{15}$$

The parameters A and σ denote the amplitude and the width of the lowered Gaussian, respectively. The term $w_{inhibit}$ determines the amount by which the Gaussian is lowered. The term $g()$ is the sigmoid function and it determines the values for which the field is considered to be activated. The term h represents the field threshold and, in order to obtain asymptotically stable single peak solutions, its value is chosen to be $-W_{max} < h < 0$ where $W_{max} = \max\left(W(\phi) = \int_0^{\phi} w(\varepsilon)d\varepsilon\right)$ represents the maximum with respect to ϕ of the integral function of the interaction kernel [14, 15].

The DNF presents two interesting properties [15]: (a) localized excitation patterns can be retained in the field even after the stimulus has vanished and (b) different stimuli can excite or inhibit each other according to the field structure. We regard the first property (a) as a form of memory of the control stimuli, while we regard the inhibition/excitation properties of the DNF (b) as a mechanism to realize the coordination of the control inputs. We introduced a stage of low-level representation of control stimuli that derive from the robot's perception of objects in the environment by means of the DNF. The control layer that comprises the DNF is located between the phase of sensory data acquisition and the phase of motion control. The control scheme is visible in Figure 7. **Objects' representation and behaviour coordination for reactive control of the Nao robot**

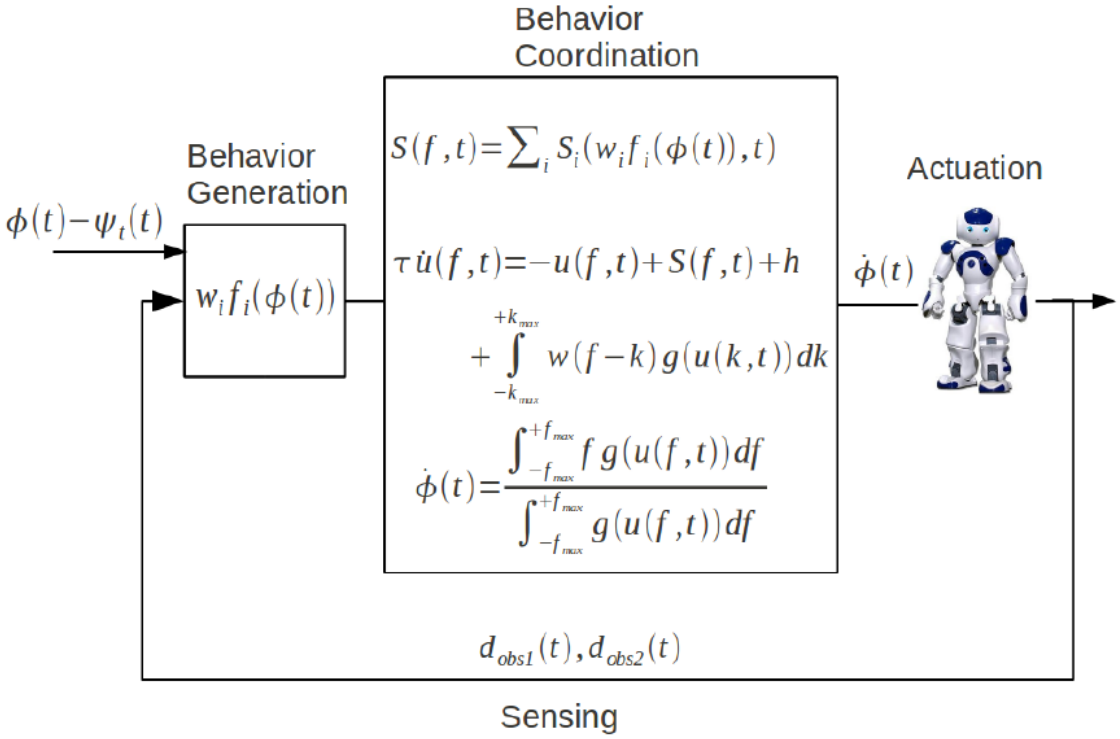

Figure 7. Objects' representation and behaviour coordination for reactive control of the Nao robot

The input $S(f,t)$ represents the contribution of each detected obstacle with its force representation (see D2.2 Section 4) or of the target dynamic to the computation of the control commands. Following a common approach (see [15]), we represent each force as a Gaussian "bump" to the field centred at the force's value. When the input represents a repulsive force, the amplitude of the Gaussian bump is inversely proportional to the distance from the obstacles. In this way, when a force represents a close obstacle, the centre of the bump is more distant from the centre of the field and the amplitude of the bump is larger with respect to the amplitude of the bump associated to the attractor dynamic. Therefore, if forces that represent contrasting behaviours are present in the field at the same time, the field will operate a selection among them. The general expression of the field's input in the case of a repulsive contribution can be seen in Eq.(16).

$$S(f_i,t) = \alpha_i \frac{1}{d_{obs}} \exp\left(-\frac{(\dot{\phi}(t) - f_i(t))^2}{\sigma_i^2}\right), \tag{16}$$

When the input represents an attractive force, the amplitude of the bump is chosen to be constant; therefore its expression is given by:

$$S(f_t, t) =, \alpha_t \exp\left(\frac{(\dot{\phi}(t) - f_t(t))^2}{\sigma_i^2}\right) \quad \texttt{if} \;\; d_{obs} < d_{lim} 0 \;\; \texttt{if} \;\; d_{obs} > d_{lim} \tag{17}$$

We applied the DNF architecture for behaviour coordination to the navigation control algorithm of the Nao. The angular resolution provided by the two sonar sensors of the robot is such that in the majority of conditions both sonar sensors perceive the same obstacle at approximately the same distance, generating control commands with similar module but opposite in sign. When the robot is facing a U-shaped obstacle, it is likely that it would not be able to overcome it, because there is no force that prevails in a steady way over the entire turning movement.

Therefore the experimental set up is designed to show how the DNF can coordinate behaviours also in those situations in which the simple coordination mechanism, expressed by the summation of the forces which is described in D2.2 Section 4, would fail. Furthermore, we show how the use of the DNF as behaviour coordination layer provides an implicit way to represent the location of obstacles with respect to the robot because of its recurrent property defined by the convolution integral of Eq.(15). Indeed, the DNF does not generate the control input directly but it is a form of short term memory for representing the obstacles in the environment and coordinating the control input, which is generated by the already existing navigation algorithm.

Before the start of the KSERA field trials we tested the reactive control scheme shown in Figure 7 in three different scenarios: (a) U-shaped obstacle, see Figure 8; (b) corridor, see Figure 9; (c) complex environment, see Figure 10 (U-shaped obstacle followed by a corridor).

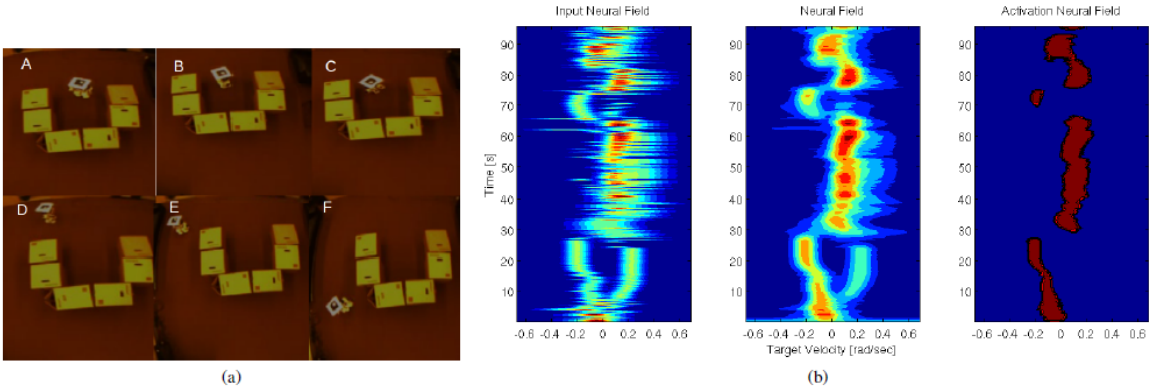

Figure 8. The DNF selects between two competing forces and maintains steady its decision. As a result the robot overcomes a U-shaped obstacle. The trajectory of the robot is qualitatively described in (a), while the evolution of the neural field is displayed in (b). DNF activity is shown with the neurons along the x-axis and time progressing along the y-axis from bottom to top. After an initial forces' integration (0s-3s), repulsive forces become large and the DNF selects between them (4s-26s). The DNF maintains steady its decision even when the selected force becomes weaker (14s-18s). After the robot exits the obstacle, the field mainly integrates the contributions (30s-90s).

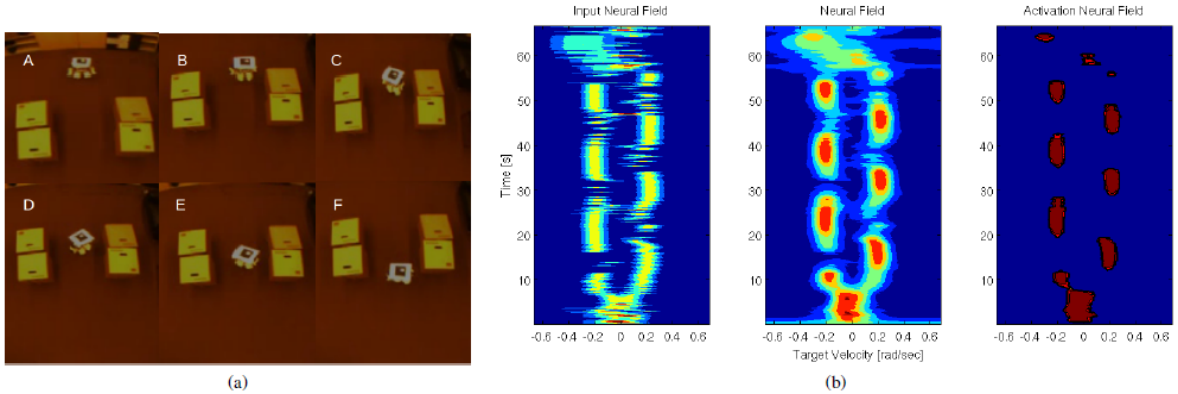

Figure 9. The DNF selects alternatively between two competing forces resulting in the robot being able to pass the corridor. The trajectory of the robot is qualitatively described in (a), while the evolution of the neural field is reported in (b). After an initial forces' integration (0s-6s), repulsive forces become larger and the DNF selects between them (6s-12s). The DNF selects alternating the two competing forces until the robot has passed the corridor (12s - 54s).

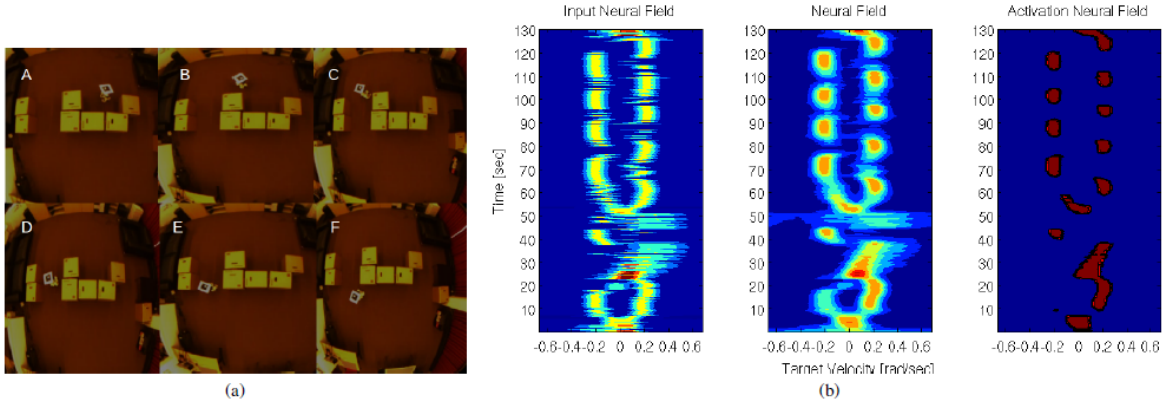

Figure 10. The robot shows two different behaviours when it first encounters the U-shape obstacle and then the corridor due to the selection mechanism of the DNF. The trajectory of the robot is qualitatively described in (a), the evolution of the neural field in (b). After an initial forces' integration (0s-5s), the field selects one of the competing forces until the robot overcomes the U-shape obstacle (5s-21s). When the robot enters the corridor, the DNF operates an alternate selection of the input stimuli until the robot passes the corridor (58s-120s).

Experimental results show that the use of the DNF as behaviour coordination layer allows the robot to complete its task effectively and that the selection mechanism changes according to the context in which the robot operates. Therefore the behaviour shown by the robot, which is determined by the DNF, changes as context changes.

2.4 Robot mobile behaviour based on ubiquitous monitoring and user context

Within the KSERA system the robot might have different navigation goals depending on the detected user and/or environmental context. A detailed description of how navigation behaviours are triggered by the KSERA system depending on the detected user commands, personal data or external events such as an incoming video connection is given in D5.4. The mobile behaviours reported in this deliverable contribute to individual states, which are described in the overall context in the state machine diagrams of D5.4.

3 Intention reading: predicting user movement goals

One of the most important tasks of the socially assistive robot in the KSERA project is navigation towards a person. However, approaching a person introduces new problems: people are not stationary, a domestic environment is dynamically changing and cluttered, etc. Clearly, if a robot knew a person's intentions it could anticipate the person's destination and move to an appropriate location in the appropriate time window [5]. Therefore, we investigated how it is possible to use high-level information regarding the presence or absence of a person in a room to predict the user's future locations in a house with multiple rooms. It should be noted here that KSERA uses single room scenarios. The results are still highly relevant, however. First of all the granularity of the model is easily adjusted to locations within a single room. Second, the scenarios of KSERA are easily scaled up to multiple room scenarios (although there are technical difficulties to realise this).

Several methods for movement prediction of humans have been developed [23, 24, 25], most of which use clustering of movement patterns to predict movement goals. For example, Bennewitz et al. [23] present an implementation of clustering using the expectation-maximization algorithm. An important property of clustering-based approaches is that they use only low-level information (e.g. trajectory data) to predict the goal of the movement, which makes them sensitive to dynamically changing environments. Therefore these methods do not take advantage of the hierarchical nature of many human behaviours in domestic environments. A behaviour, for example getting up in the morning, can be comprised of several sub-behaviours such as making breakfast and taking a shower. These sub-behaviours can also consist of several sub-behaviours such as walking from one room to the other. This subdivision continues until one arrives at the level of action primitives, which could be considered to be the building blocks of more complex behaviours [5]. Thus, low level behaviours can be a part of many higher level-behaviours. This helps us to represent an infinite set of behaviours in a convenient and economical way. It is likely that, if we know what high-level behaviours a person is involved in, we can recognize which lower-level behaviours the person is executing. Indeed, Cuijpers, van Schie, Koppen, Erlhagen & Bekkering [5] show that information about higher-level behaviours can improve recognition of lower-level behaviours. Other research by Kanda, Glas, Shiomi & Hagita [26] use motion clustering to determine human walking behaviour in a shopping mall. They used this information to decide which people their robot should approach.

Several methods for recognizing high-level behaviour use implementation and extensions of Hidden Markov Models (HMM). HMMs model the world using discrete states and changes in the world using transitions between these states. This makes them suitable for describing temporal patterns, for example in the field of speech recognition [27]. Due to their simple structure, efficient inference is possible. Nguyen, Bui, Venkatesh and West [28] use an Abstract Hidden Markov Memory Model to recognize high level behaviours from video data; Duong, Bui., Phung and Venkatesh [29] use a Switching Hidden Semi-Markov Model to recognize activities, and Nguyen, Phung, Venkatesh & Bui [28] use a Hierarchical Hidden Markov Model (HHMM) to infer high-level activities from movement trajectories. Van Kasteren, Noulas, Englebienne & Kröse [30] use both an HMM and conditional random fields for activity detection from sensor readings. HMM's and their adaptations can not only be used to infer activities from observations, it is also possible to use them to do prediction. Gani, Sarwar and Rahman [31] use an HMM to predict wireless network traffic.

In order to make predictions about the future location of a person, we need to model their behaviour. By using a HMM, the world is represented by a finite number of states, and between these states, transitions are possible. The chance of going from one state to another is the transition probability. In an HMM the next state is assumed to be only dependent on the current state, this is the Markov assumption. This assumption allows us to construct a square transition matrix A. The element a_{ij} represents the chance of going to state j given that the current state is i. A special transition is the self-transition where the system stays in the same state or, more formally, $j = i$. **Figure 11** shows a simple example with two states (indicated by the blue circles), including all transitions (indicated by the arrows) and transition probabilities a_{ij}.

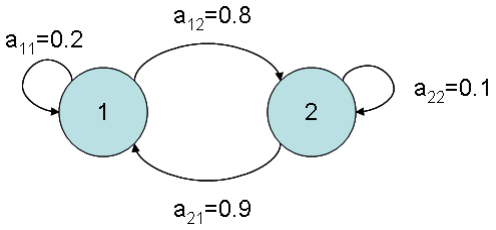

Figure 11. Example of a two-state Markov Model with given transition probabilities.

The transition probabilities can be represented as a transition matrix. The transition matrix for **Figure 11** would be:

$$A = \begin{bmatrix} 0.2 & 0.8 \\ 0.9 & 0.1 \end{bmatrix}$$

If we apply this to our research topic we can use states to represent the location of a person. For example one state could represent that the person is in the bathroom. A state transition would then mean going from one room to another and the transition probability is the probability of going to a certain room given the current location. Usually states are not directly observable and have to be inferred from (noisy) observations. The true state can therefore be thought of as hidden. If we assume that there is a finite number of different observations we can define the observation matrix B, where each element b_{ik} is the chance that k is observed when the system is in state i. Formally, k represents an element from an abstract alphabet. Basically, this can be anything, but in our case it represents observing a person to be present in room k.

In order to test our model we used real data about the location of COPD patients in the home collected during the project: SenterNovem Point One project 'IPTV Portal to Wellness'. These data were obtained by recording the room a person is in using motion sensors. The recordings of the motion sensors however can be faulty; they can signal the presence of a person in a room while the person is not present or, they can fail to identify the presence of a person in a certain room. Therefore our HMM takes into account the probability to observe a certain reading assuming a person is in a certain room. **Figure 12** shows a simplified example of the observation model. Two states model the presence of a person in room 1 or 2 (blue circles) as described in **Figure 11**, but now an observation is made using the sensors (yellow circles). In this example the most likely situation is that the person is observed to be in the room he is actually in (vertical solid arrows). However, there is also a small chance that the sensors erroneously detect the person to be in the other room (diagonal solid arrows).

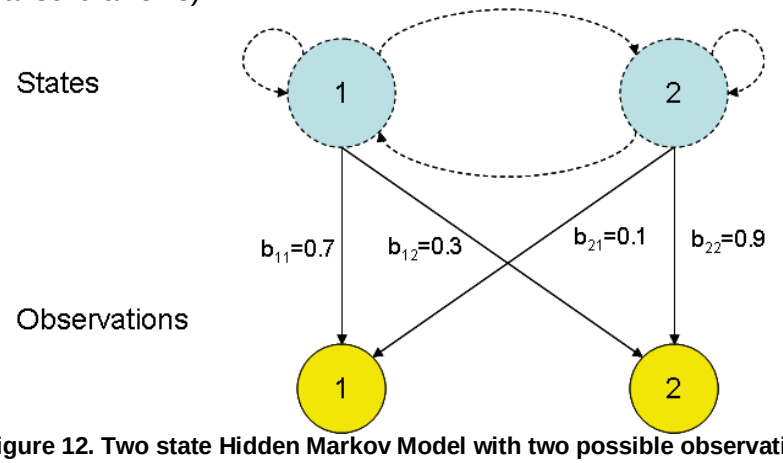

Figure 12. Two state Hidden Markov Model with two possible observations

As before, the observation probabilities can be described by a matrix. For the system in **Figure 12** we obtain:

$$B = \begin{bmatrix} 0.7 & 0.3 \\ 0.1 & 0.9 \end{bmatrix}$$

Usually we only have a sequence of observations but we want to know the most likely state. To determine the likelihood of a state given a sequence we can use a modified version of the forward-backward algorithm described by [27]. When we know this likelihood we can use the transition probabilities to determine the most probable next state.

3.1 Hierarchical Hidden Markov Model

One disadvantage of modelling behaviour using a HMM is that this model ignores the hierarchical nature of behaviours. A behaviour such as “making breakfast” might involve several sub-behaviours, such as going to the kitchen to get food and going to the living room to eat it. On the other hand some sub-behaviours such as going from the kitchen to the living room might be part of more than one higher-level behaviour. If we can infer the behaviour people are involved in, we might be able to make more accurate predictions about the next sub-behaviour. To capture this hierarchical structure we can use the Hierarchical Hidden Markov Model described by Fine, Singer and Tishby [32].This model consists of a layered structure of Markov Models (MM). On the top levels (the parent level) each state activates another MM at the child level. This recursive structure ends at the lowest level where the states produce observations, as in the normal HMM. Each MM behaves as a regular MM until it reaches an end state. From the end state, control of the systems is returned to the layer above. The end state does not produce observations but acts like a flag, signalling the sequence at the current level has ended.

In **Figure 13** our example of the HMM is extended to a two-level HHMM. States in the parent layer (*internal states)* activate a child HMM in the lower level. The states in this child HMM are *production states*, because they do produce observations as in a regular HMM. Which sub-state is activated first is modelled using the initial sub-state probability π. There are separate transition matrices for each child MM. In order to move back from the child states to the parent state the probability of a transition to the end state is added to the transition matrix. As the end state is reached at the child level the model proceeds at the parent level where a transition between the internal parent states may occur.

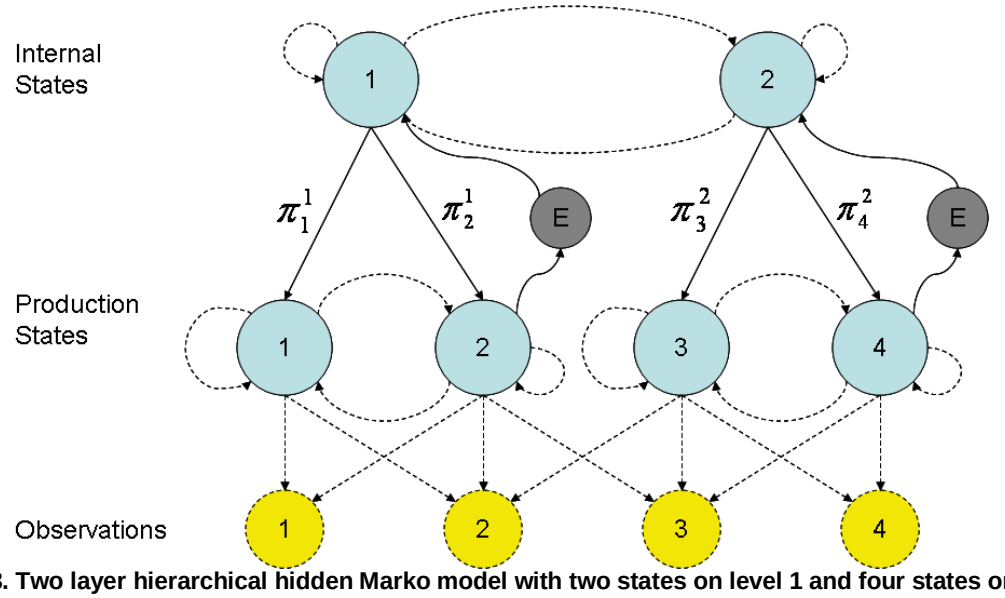

Figure 13. Two layer hierarchical hidden Marko model with two states on level 1 and four states on level 2

So far this model is able to capture the hierarchical nature of behaviour, but not yet its shared sub-structures. To do this we can use *parameter tying*. If we want to model that states 1 and 3 represent the same location but for different activities we can make the observation probabilities for state 1

and 3 equal. With this model we can use the generalized Viterbi algorithm described by Fine et al. [32] to predict the most likely next state given an observation sequence.

3.2 Experiments

To test both models we used data obtained from 2 COPD patients from the IPTV project. This data consists of sensor readings indicating in which room a person was detected. Both participants lived in a similar apartment consisting of:

- Bedroom
- Bathroom
- Kitchen
- Living room with a chair

Each room had a motion sensor and the chair in the living room had a pressure sensor. In **Figure 14** a lay-out for one of the apartments is shown.

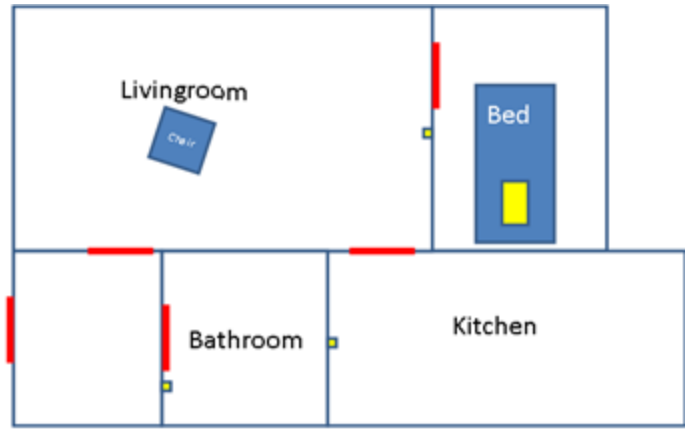

Figure 14. Apartment lay-out, doors are marked red and sensors are marked yellow

For the model, the number of states is 6 (5 locations and 'away') and the number of observations is 7 (5 locations, 'unknown' and 'multiple').

3.2.1 Test Method

To test the performance of the different models the following aspects were tested:

- Prediction accuracy
- Learning over time
- Adapting to different persons.

Prediction accuracy

For a given sequence the prediction accuracy is the ratio of the number of correctly predicted states to the total number of states in the sequence. To test the prediction accuracy the following method is used:

- The model is trained on 8 days of data
- The trained model is used to predict the remaining days
- This is repeated for both data sets.

For comparison prediction was also done using the following method:

- The observation matrix was obtained using 8 days of data
- All transition probabilities were set to $\frac{1}{N-1}$ making every transition equally likely
- Prediction was done again for both datasets.

Learning over time

To test the effect of parameter updating on prediction accuracy the following the following method is used:

- The observation matrix is obtained from training data
- All transitions probabilities in the transition matrix are equal
- An iterative sequence is started where
    - The system is used to test on one day of data
    - The system's parameters are updated using the used day
    - This procedure is repeated for each day in the dataset.

Adapting to a person

To test how quickly and how well a system can adapt to a different user the method for learning over time is used with one addition. After all the data from the first set have been used for prediction the iterative procedure described in the previous paragraph is continued with the other data set.

3.2.2 Results

3.2.2.1 Hidden Markov Model

Accuracy

For each dataset the model was trained using the first 8 days of data and then tested on the remaining days. In **Figure 15** the results for both datasets can be found. The results are summarized in **Table 1**. For both datasets the trained model outperforms the control model with a factor 2.

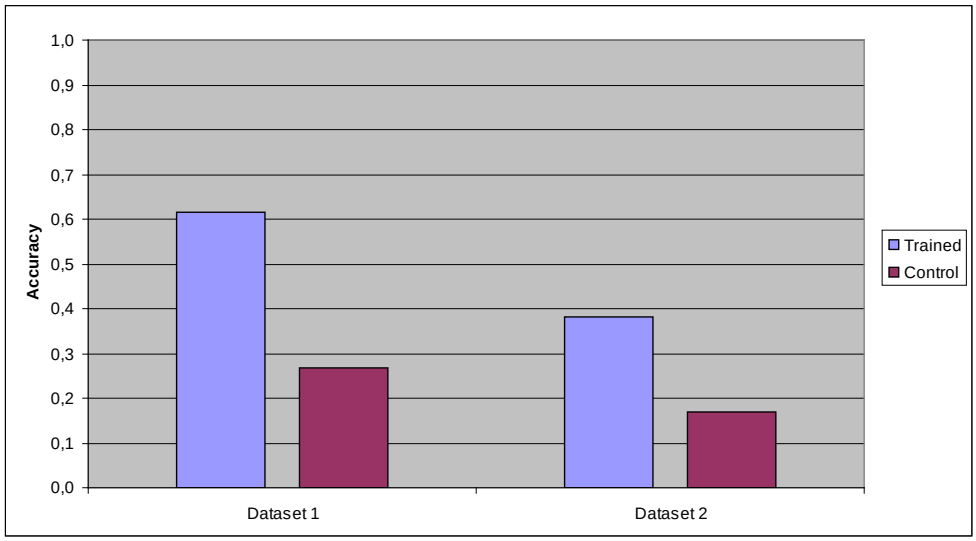

Figure 15. HMM Prediction results for dataset 1 and dataset 2

Table 1. Results of HMM

	Trained	Control
Dataset 1	0.62	0.27
Dataset 2	0.38	0.17

Learning over time

For both systems a learning rate of 0.05 was used. In **Figure 16** the prediction results for dataset 1 for the HMM with parameter updating are shown. We can see that already after one day the prediction accuracy has increased to the level of the trained HMM. The results for dataset 2 show the same pattern, as can be seen in **Figure 17**.

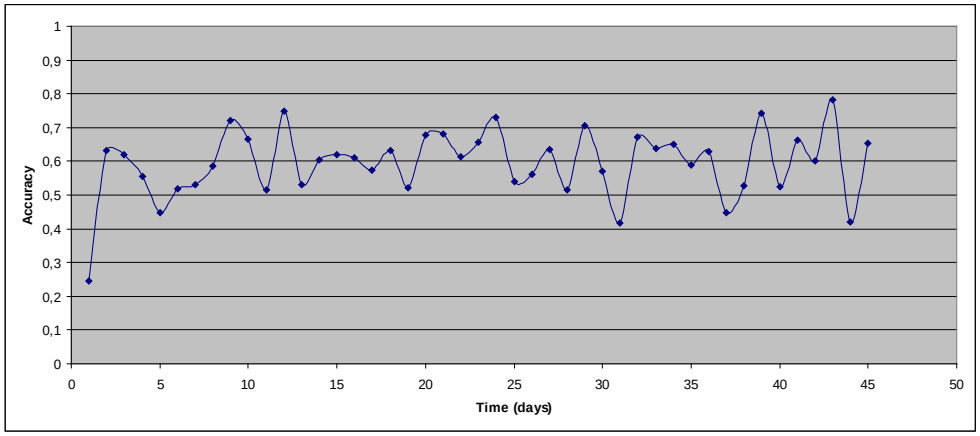

Figure 16. HMM Prediction results for dataset 1 using parameter updating (l=0.05)

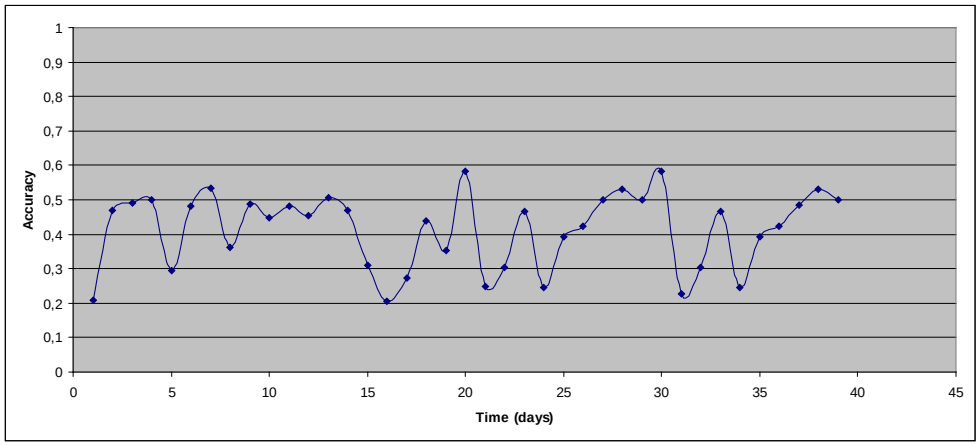

Figure 17. HMM Prediction results for dataset 1 using parameter updating (l=0.05)

Adapting to another person

In **Figure 18** the prediction results are shown for the HMM that is updated first using dataset 1 and then using dataset 2. We can clearly see a drop in accuracy after switching to the other dataset. The model that is first updated using dataset 2 and then using dataset 1 shows the same behaviour, as can be seen in **Figure 19**.

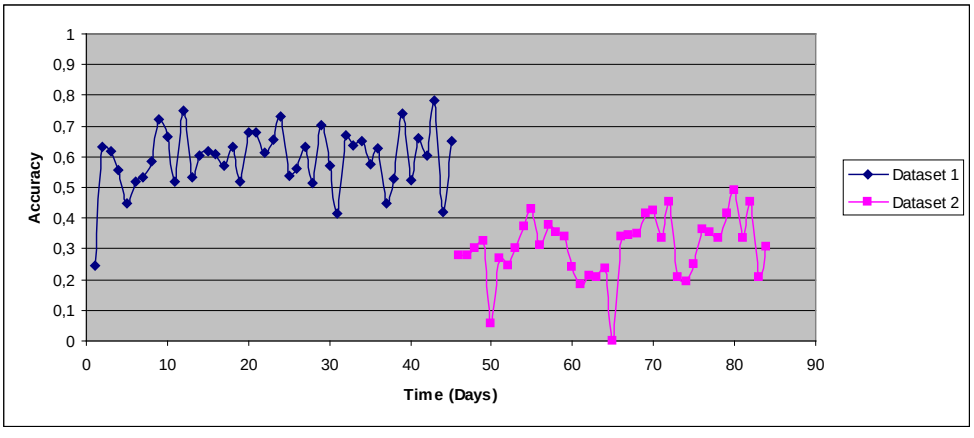

Figure 18. HMM Prediction results when trained on P1 (l=0.05)

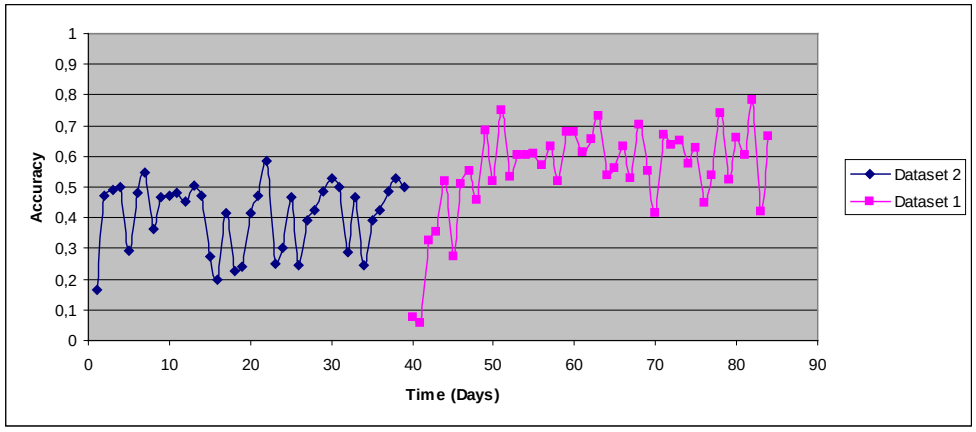

Figure 19. HMM Prediction results when trained on P2 (l=0.05)

3.2.2.2 Hierarchical Hidden Markov Model

Accuracy

For the HHMM we obtain a prediction accuracy for both the internal and the production states. In **Figure 20** the results for both dataset can be seen. In both datasets the prediction accuracy for internal and production states are quite similar.

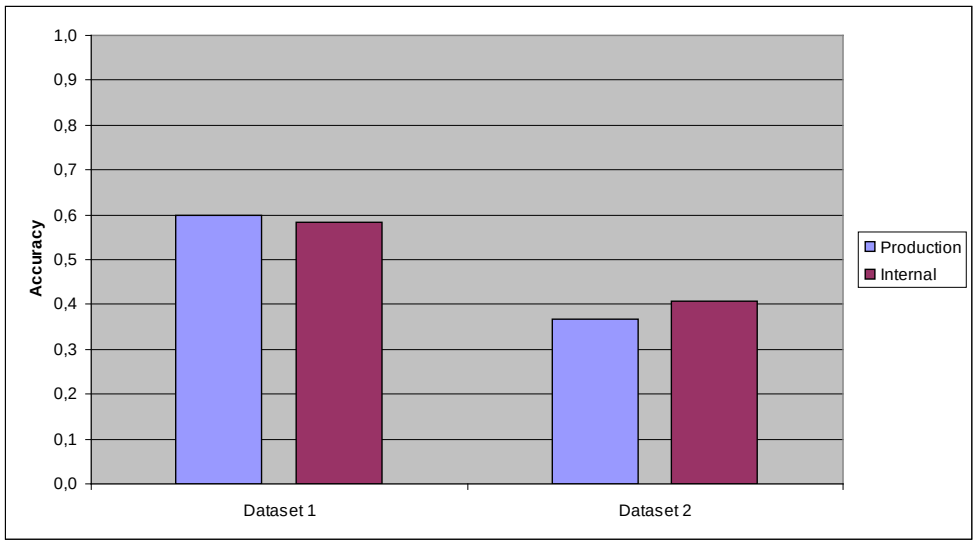

Figure 20. HHMM Prediction results for dataset 1

Learning over time

For both systems a learning parameter of 0.05 was used. The results for dataset 1 and 2 can be found in **Figure 21** and **Figure 22**. For both datasets we can see that just as with the HMM the system quickly reaches the values from the trained systems. For dataset 1 it seems that there is a larger fluctuation in the prediction accuracy for internal states compared to the production states. This is not the case for dataset 2.

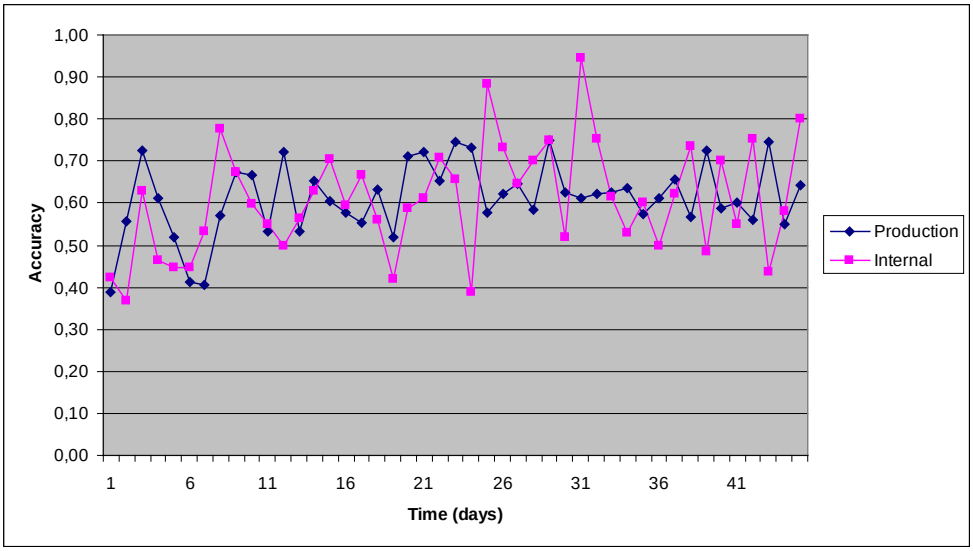

Figure 21. HHMM Prediction results for dataset 1 using parameter updating (l=0.05)

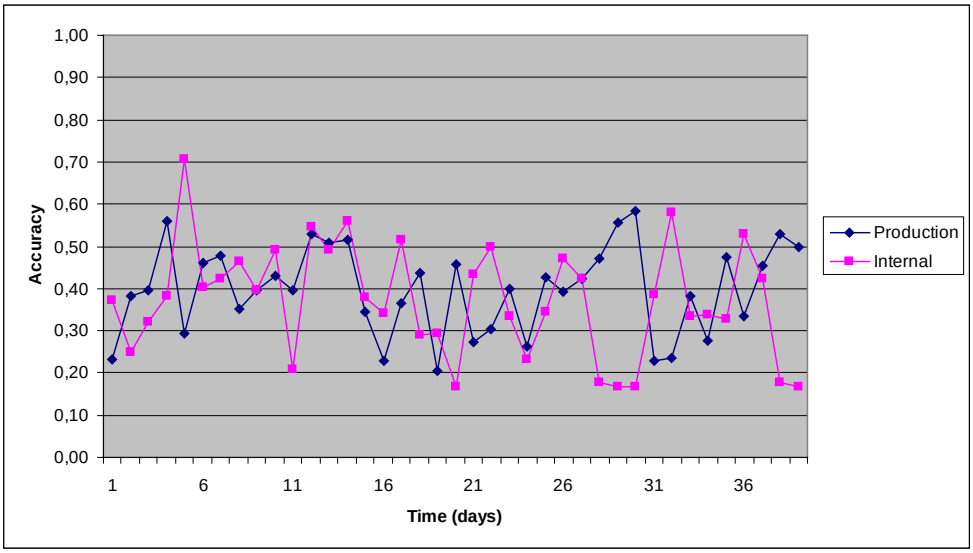

Figure 22. HHMM Prediction results for dataset 2 using parameter updating (l=0.05)

Adapting to another person

In **Figure 23** the prediction accuracy for the HHMM that is first updated using dataset 1 and then using dataset 2 is shown. The prediction accuracy for the model that is first update using dataset 2 and then using dataset 1 can be seen **Figure 24**. For both cases the prediction accuracy drops for both internal and production states after the transition between the datasets.

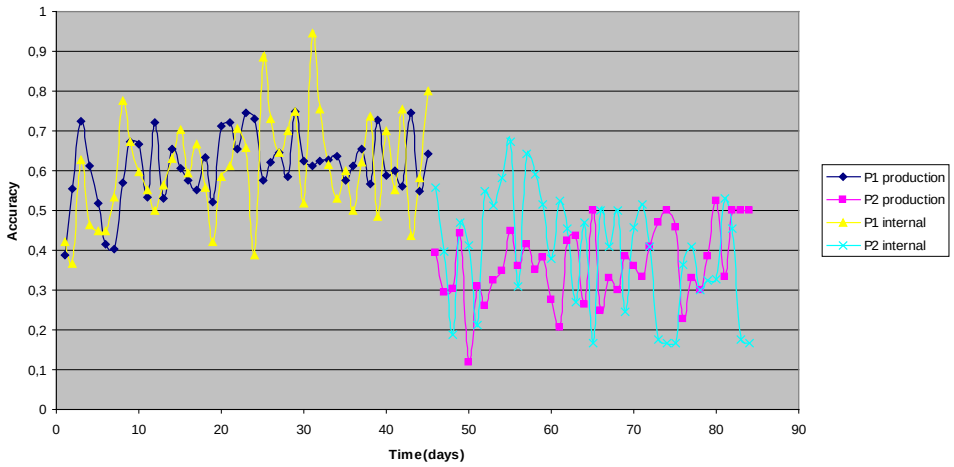

Figure 23. HHMM Prediction results when trained on P1 (l=0.05)

As a conclusion, prediction learnt for one dataset can be applied to another dataset, however, only with a major loss of accuracy. The HHMM leads to better accuracy than the HMM when predicting from dataset 2 to dataset 1, but in the other direction the HHMM did not lead to an improvement. Hence, a predictor for a person's movement within a home needs to be trained to the habits of an individual person and needs to consider the specific situations/context for which it is trained.

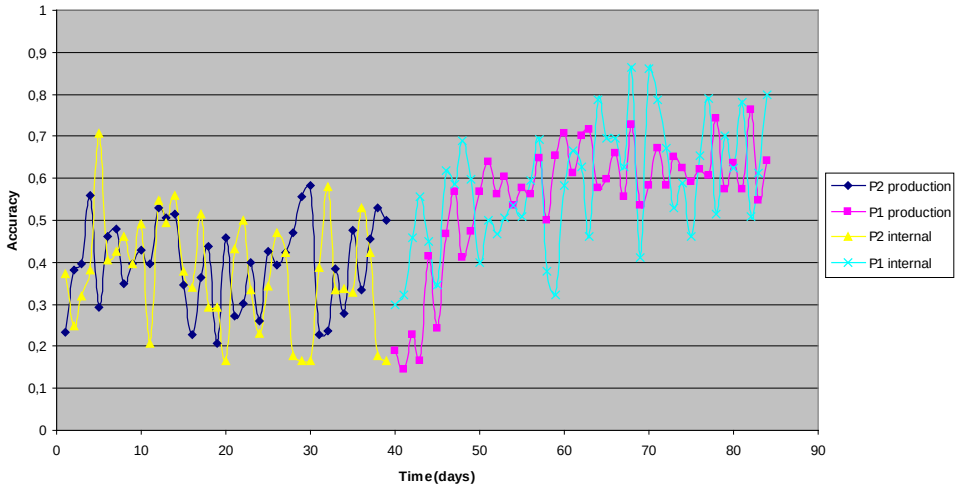

Figure 24. HHMM Prediction results when trained on P2 (l=0.05)

3.2.3 Discussion and Conclusions

The results show that a simple HMM can already predict the next location of a person with a reasonable accuracy. We expected that, in comparison to the simple HMM, an HHMM would increase prediction accuracy. However the results show than performance is roughly equal. An important bottleneck could be the definition of the higher states. Instead of manually defining the model structure a self-organizing algorithm can be used. Another way to improve the usefulness of the data is to record information about high level behaviour (Kasteren et al. 2008). The fact that even with these arbitrarily defined higher states a prediction accuracy of 40-60% was achieved is promising.

We also expected that the HHMM can flexibly adjust to changing circumstances including different people. The results for parameter updating show that the HHMM quickly adapts to another user (or behaviour), converging to the accuracy of the system trained for the new user.

In the data used the position of the person was only known on a room level. When different locations in one room can be distinguished, it could be easier to recognize different behaviours. So these results apply to KSERA, which has single room scenarios, in two ways: firstly, the model can be applied directly to KSERA scenarios that are scaled up to multiple rooms, and, secondly, the model can be used to predict user movements towards points of interest within a room (such as the projection site).

4 Intention reading: user experience

Anticipation is of practical use especially for a slow robot to arrive at a location to interact with the user in due time. We have shown that the Hierarchical Hidden Markov Models are able to learn and predict different human behaviours, and thus, they capture different daily routines and differences between people. The technological benefit is clear: the anticipating robot is faster and, because it is adaptive, personal. Another way to look at this is to ask what the benefit would be from a user's point of view.

Does learning of high-level information about a person's goals improve the robustness and flexibility of a robot's navigational skills in such a way that the robot's behaviour appears more natural, trustworthy and socially intelligent to users? In order to answer this question we looked at whether anticipatory walking behaviour of a robot is perceived as more intelligent, and as a consequence, it appears more trustworthy and likeable. For that purpose, we compared person-following behaviour, which is purely reactive without anticipation, interception behaviour, which extrapolates the person's walking trajectory in time, and walking to the anticipated destination, which requires inferring the goal of a person. It is conceivable that the degree with which these different behaviours are beneficial depends on the particular context. To check this we used three different scenarios which are part of KSERA and are described in D1.1, Section 3.1: phone call, medical emergency, and health exercise. These scenarios have different degrees of urgency. We expected that a health exercise, which is not urgent, is most compatible with waiting at the anticipated location, whereas a medical emergency, which is very urgent, is most compatible with following behaviour, as the robot is always nearby. Thus, we expect scenarios and behaviours to interact. Further details can be found in Cuijpers et al. [21].

4.1 Method

4.1.1 Experimental Setup

The robot was manually controlled by the experimenter from another room through its inbuilt WiFi connection and inbuilt walking behaviours. The experiment took place in a living room setting; the layout of the room is shown in Figure 25 (left). The set-up consisted of a dinner table (labelled by A), a coffee table (labelled by B) and a comfortable chair (labelled by C).

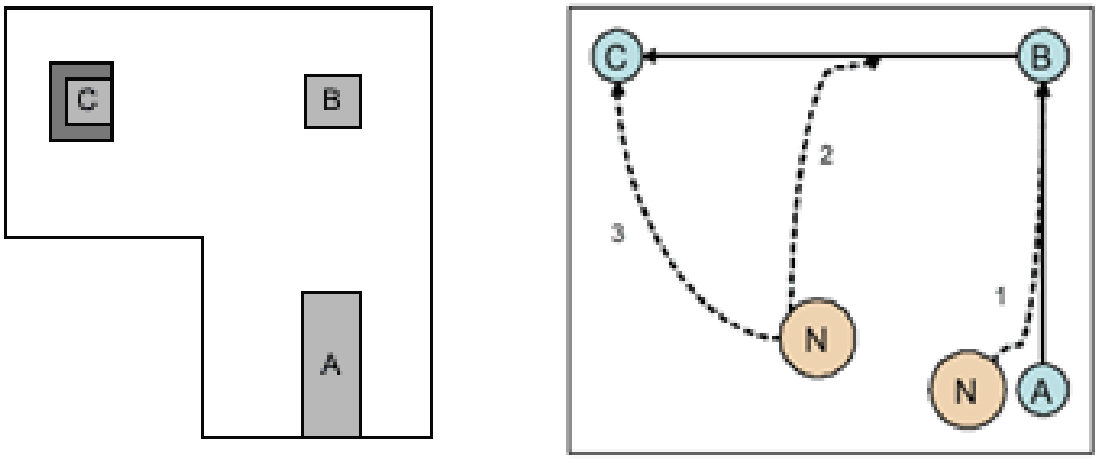

Figure 25. Left: layout of the experimental set up. Right: schematic of robot's trajectories.

4.1.2 Participants and task

Fourteen participants (11 male, 3 female) whose age varied between 19 and 32 (mean age 24.5, SD = 4.3) participated in the experiment. Participants were unfamiliar with the research topic. They were all students or employees of the Eindhoven University of Technology. During the experiment, participants had to perform a task which required their interaction with a robot, after which they had to fill out a questionnaire.

4.1.3 Design

The experiment followed a 3x3 design, with 3 ways of approaching by the robot and 3 different scenarios. The order of presentation was randomized and counter-balanced across participants. The three approaching behaviours were as follows: 1) Following - Nao follows the person. It announces its behaviour by telling the participant “I will follow you”. 2) Intercepting - Nao intercepts the person halfway between the coffee table and the chair. 3) Anticipation - Nao goes to the anticipated destination (the chair) and waits for the participant to arrive there. It announces its behaviour by telling the participant: “I think you will go to your chair”. The three ways of approaching result in three different robot’s trajectories, which are shown in Figure 25 (right).

To see whether the user evaluation of approaching behaviour depends on context, 3 scenarios were devised based on use cases of KSERA reported in D1.1, Section 3.1. The scenarios tested during the experiment were: 1) Phone Call - the robot approaches the participant, and says “There is a phone call for you” and starts making a ringing sound. The participant is instructed to reject the phone call by touching the robot's head. 2) Medical Emergency - the participant is told to imagine that he or she feels ill and has to wait at the coffee table for the robot to arrive there. After arriving close by and in front of the participant the robot says “Do not worry I will call the doctor”. 3) Health exercise - Nao approaches the participant and tells him/her “this might be a good time to exercise”.

4.1.4 Procedure

After arriving in the lab, each participant was asked to sit down at the coffee table and read and sign an informed consent form. When they had read it and had no further questions they could start the trial while the experimenter controlled the robot from another room and could not be seen by the participant. At the beginning of each trial, participants received instructions on paper at the dinner table (A in Figure 25 (left)). They were told that they were to imagine that they had just finished a meal at the dinner table and wanted to read the paper in their favourite chair. To do this they first had to pick up the paper at the coffee table (B in Figure 25 (left)).

On the coffee table there was a laptop, which showed new instructions and participants listened to a 30-second audio clip from the BBC Radio news. This was necessary because Nao walks very slowly, so that participants had to be slowed down to enable Nao to keep up with them. Afterwards participants received a question about the radio clip in order to make sure participants paid attention and to give the impression that it was part of the task. The answers themselves were not used for analysis. After the audio clip had finished they were instructed either to pick up the paper and to walk to the chair (C in Figure 25 (left)) or stay at the coffee table because they supposedly felt ill and had to wait for the Nao robot to come to their aid. When the participants had reached the chair or Nao had reached the participant, the trial was over and participants had to rate the robot's behaviour during that interaction. After the trial was finished the participants had to fill out the Godspeed questionnaire [22] in which they had to rate the robot's behaviour. It also contained the question about the news clip. The participants filled out the questionnaire at the coffee table. They were told that after filling it out, they could immediately start the next trial. Each participant performed 9 trials in total. After the end of the last trial the participant was asked for comments or remarks. The research topic was explained to them; they received their financial reward and were thanked for their participation.

4.1.5 Results

To test the effects of our manipulations on the different dimensions of the questionnaire we used a repeated measures analysis of variance (ANOVA) with the Likert scale scores as dependent variable and both manipulations (approaching behaviour and scenario) as independent variables. Before using the data to compare the effects of the experimental manipulations, the consistency and reliability of the items of each scale were tested. Items that had a strong detrimental effect on consistency or reliability were removed. To test this we computed the Cronbach's alpha statistic. Items are considered to have good consistency if Cronbach's alpha exceeds 0.7. We only found one moderate reliability which occurred for the perceived safety dimension ($\alpha = 0.647$). This could be improved by removing the quiescent - surprised item. In **Table 2**, the resulting number of items and the computed Cronbach's α are given for each dimension.

Table 2. Internal consistency reliability scores

	Items	Cronbach's α
Anthropomorphism	5	0,817
Animacy	6	0,806
Likeability	5	0,883
Perceived Intelligence	5	0,870
Perceived Safety	3	0,831

In **Table 3**, the mean Likert rating is shown averaged across all behaviours, scenarios and items for each dimension. The range and standard deviation for the different dimensions are also shown. In most cases the mean scores are quite high. In fact, the average across all behaviours and scenarios is ≥ 3.50 for all dimensions. This shows that participants in general had a positive attitude toward the Nao robot and the behaviours and scenarios that were used.

Table 3. Likert scale scores averaged across scenarios and robot behaviours for each dimension.

Dimension	Min.	Max.	Mean	SD
Anthropomorphism	2.00	4.80	3.50	0.66
Animacy	2.17	5.00	3.69	0.61
Likeability	2.20	5.00	4.07	0.72
Perceived Intelligence	2.20	5.00	3.89	0.69
Perceived Safety	2.25	5.00	3.87	0.63

Appropriateness	1.00	5.00	3.85	1.05

In Figure 26 the mean scores of the Likert scales are shown for each dimension. Each panel shows the mean scores for a single dimension and for each scenario (x-labels) and robot behaviour (symbols). When comparing the different walking behaviours within a given scenario and dimension, it is clear that in most cases there seems to be no effect of walking behaviour. Indeed, a repeated measures ANOVA reveals that there is no significant main effect of walking behaviour for each of the dimensions (Animacy: $F(2,24) = 2.15, p = 0.14$; Anthropomorphism: $F(2,24) =< 1, p = 0.45$; Likeability: $F(2,24) = 1.54, p = 0.24$; Perceived Intelligence: $F(2,24) = 1.81, p = 0.18$, Perceived Safety: $F(2,24) = 1.22, p = 0.31$, Appropriateness: $F(2,24) = 1.22, p = 0.76$). On the other hand, when comparing between scenarios the 'Exercise' scenario tends to score lower than 'Medical emergency' for the dimensions (Perceived Intelligence, Perceived Safety, and Appropriateness). The repeated measures ANOVA confirms that there is a main effect of scenario for Likeability $(F(2,24) = 3.43, p < 0.05)$, Perceived Intelligence $(F(2,24) = 7.01, p < 0.05)$, Perceived Safety $(F(2,24) = 4.34, p < 0.05)$ and Appropriateness $(F(2,24) = 6.76, p < 0.05)$, but not for Animacy $(F(2,24) = 2.298, p = 0.12)$ and Anthropomorphism $(F(2,24) = 1.29, p = 0.30)$. We also checked whether there were interaction effects between scenarios and robot behaviours, but none were significant.

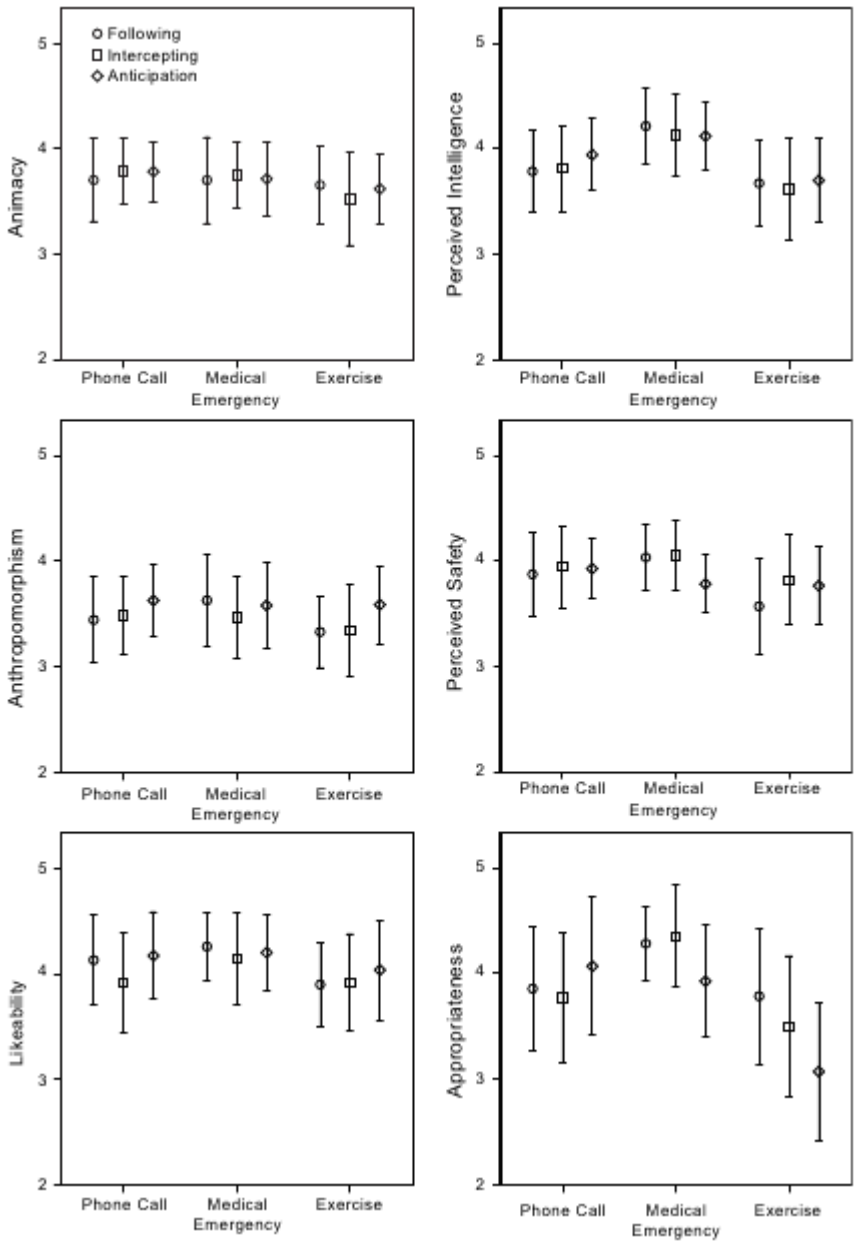

Figure 26. Each panel shows the average Likert scale score for each dimension. The labels on the x-axis indicate the scenarios and the symbols indicate the robot's walking behaviours. The error bars indicate 95% confidence intervals.

4.1.6 Conclusion

To investigate how a person's attitude changes towards a robot depending on the anticipatory character of its walking behaviour and the urgency of particular context, we performed a study where we had subjects rate the robot's behaviour based on the Godspeed questionnaire developed by Bartneck et al. [22]. We expected that people would have a more positive attitude towards robots that show an anticipatory behaviour, but that this effect would be moderated or even reversed depending on the urgency with which a robot should act. Results indicated that the walking behaviour did not affect our participants' ratings. We only found a main effect of the urgency of the scenarios. Medical emergency was rated as more positive than health exercise. We found no interaction effects.

The current study had a number of limitations partly related to the typical KSERA set-up, which could explain in hindsight why there was no effect of walking behaviour. The Nao cannot walk fast

enough to intercept anyone even when he or she has a slow pace. Therefore we introduced a secondary task to delay the participant. This worked well as the robot arrived at the right location at the right time, but it may also have distracted the participants too much. Another explanation is that there was only one possible destination for the participants. Therefore it is possible that the participant did perceive the anticipatory behaviour as anticipatory, even though the robot verbally expressed its behaviour.

The intercepting behaviour gets the lowest ratings for every aspect except for perceived safety. This could be due to several reasons. Unlike in the other two behaviours, the robot did not verbally emphasize its behaviour. Combined with the small absolute path differences with both other behaviours, the intercepting behaviour could be too ambiguous.

The different scenarios however, yield significant differences for animacy, likeability, perceived intelligence, perceived safety and appropriateness ratings. This could be due to differences in interaction styles between the scenarios. In the exercise condition the robot only talks to the person, in the phone call condition there is some feedback required from the user and in the medical emergency condition the robot seems to respond to the user's needs. In other words, it is not the urgency but the degree of interaction that seems to play an important role. We conclude that the attitude towards robots does not depend on its anticipatory behaviour, but on the type of interaction and the context within which this takes place.

5 Summary and outlook

In this deliverable we present the current research progresses in the field of robot navigation based on context awareness and intention reading. We improve our person localization model by adding a state of art tracking method so that the system tracks a sitting person more robustly. In order to achieve different navigation tasks, we developed a *map-based* and a *behaviour-based* navigation model.

The map-based navigation model builds up a cognitive map for an indoor environment by observing a person's movement using the ceiling-mounted camera, and learns the physical context within this room by integrating the vision information obtained from the robot's camera in the cognitive map. As a result Nao can navigate to a person based on the input of the person localization model. Robot localisation and estimation of its orientation can become more reliable by comparing the features extracted from its image with the features stored for each position in the cognitive map.

The behaviour-based navigation model addresses the issue of dynamically changing environments and provides a navigation model with real-time adaptation capacity, which can be adopted by robots with very limited sensory capabilities, for example the Nao.

We performed preliminary work to assess whether Hidden Markov Models and Hierarchical HMMs provide enough accuracy for predicting human movement goals in a complex home setting. Results confirmed that the HHMMs quickly switch between learned movement patterns and that the accuracy of prediction is as good as conventional HMMs. This model enables learning different behaviours and different personalities, and is applicable to both between-room and within-room locations. We also performed a user study to evaluate how people perceive different navigation behaviours of the robot Nao in a single room setting. Results suggest that the inference of human movement goals for improving robotic anticipatory behaviour might not be necessary for a single room scenario as in KSERA.

References

[1] T. Yamazaki, “Ubiquitous home: real-life testbed for home context-aware service,” in *First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. Tridentcom 2005.*, feb. 2005, pp. 54 – 59.
[2] T. Khan, S. Gao, and W. S. Lee, “Context-aware robot navigation based on sensor association rules,” in *Proceedings of the 2009 International Conference on Hybrid Information Technology*, ser. ICHIT ’09. New York, NY, USA: ACM, 2009, pp. 217–220.
[3] Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking-learning-detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409 –1422, 2012.
[4] M. Sindlar and J.-J. Meyer, “Affordance-based intention recognition in virtual spatial environments,” in *Principles and Practice of Multi-Agent Systems*, ser. Lecture Notes in Computer Science, N. Desai, A. Liu, and M. Winikoff, Eds. Springer Berlin / Heidelberg, 2012, vol. 7057, pp. 304–319, 10.1007/978-3-642-25920-3_21. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-25920-3_21
[5] R. H. Cuijpers, H. T. van Schie, M. Koppen, W. Erlhagen, and H. Bekkering, “Goals and means in action observation: A computational approach,” *Neural Networks*, vol. 19, no. 3, pp. 311–322, 2006.
[6] W. Yan, C. Weber, and S. Wermter, “Person tracking based on a hybrid neural probabilistic model,” in *Artificial Neural Networks and Machine Learning – ICANN 2011*, ser. Lecture Notes in Computer Science, T. Honkela, W. Duch, M. Girolami, and S. Kaski, Eds. Springer Berlin / Heidelberg, 2011, vol. 6792, pp. 365–372, 10.1007/978-3-642-21738-8_47.
[7] ——, “A hybrid probabilistic neural model for person tracking based on a ceiling-mounted camera,” *Journal of Ambient Intelligence and Smart Environments*, vol. 3, no. 3, pp. 237–252, 2011.
[8] W. Yan, E. Torta, D. van der Pol, N. Meins, C. Weber, R. H. Cuijpers, and S. Wermter, *Robotic Vision: Technologies for Machine Learning and Vision Applications*. IGI Global, 2012, ch. Learning Robot Vision for Assisted Living (in press).
[9] W. Yan, C. Weber, and S. Wermter, “A neural approach for robot navigation based on cognitive map learning,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2012, pp. 1146–1153.
[10] S. Marsland, J. Shapiro, and U. Nehmzow, “A self-organising network that grows when required,” *Neural Networks*, vol. 15, no. 8-9, pp. 1041–1058, 2002.
[11] B. Fritzke, “Growing grid - a self-organizing network with constant neighborhood range and adaptation strength,” *Neural Processing Letters*, vol. 2, no. 5, pp. 9–13, 1995.
[12] H. Bay, T. Tuytelaars, and L. Gool, “SURF: Speeded up robust features,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds., vol. 3951. Springer Berlin / Heidelberg, 2006, pp. 404–417.
[13] C. Weber and S. Wermter, “A self-organizing map of sigma-pi units,” *Neurocomputing*, vol. 70, no. 13-15, pp. 2552–2560, 2007.
[14] S.-i. Amari, “Dynamics of pattern formation in lateral-inhibition type neural fields,” *Biological Cybernetics*, vol. 27, no. 2, pp. 77–87, 1977.
[15] W. Erlhagen and E. Bicho, “The dynamic neural field approach to cognitive robotics,” *Journal of Neural Engineering*, vol. 3, no. 3, pp. R36–R54, 2006.
[16] N. Cuperlier, M. Quoy, P. Laroque, and P. Gaussier, “Transition cells and neural fields for navigation and planning,” *Mechanisms, Symbols, and Models Underlying Cognition*, vol. 3561, pp. 147–152, 2005.
[17] R. C. Arkin, *Behavior-Based Robotics*. MIT press, 1998.
[18] P. Althaus and H. I. Christensen, “Behavior coordination in structured environments,” *Advanced Robotics*, vol. 17, no. 7, pp. 657–674, Nov. 2003. [Online]. Available: http://-www.ingentaselect.com/rpsv/cgi-bin/cgi?ini=xref&body=linker&reqdoi=10.1163/-156855303769157009
[19] E. Torta, R. Cuijpers, J. Juola, and v. d. P. David, “Modelling and testing proxemic behaviour for humanoid robots (in press),” *International Journal of Humanoid Robotics*, vol. 9, 2012.

[20] E. Torta, R. Cuijpers, and J. F. Juola, “Dynamic neural field as framework for behaviour coordination in mobile robots,” in *Proceedings of the 9th International Symposium on Soft Computing for Industry (ISSCI)*. Puerto Vallarta, Mexico: World Automation Congress, 2012.
[21] R. Cuijpers, M. Bruna, J. Ham, and E. Torta, “Attitude towards robots depends on interaction but not on anticipatory behaviour,” in *Social Robotics*, ser. Lecture Notes in Computer Science, B. Mutlu, C. Bartneck, J. Ham, V. Evers, and T. Kanda, Eds. Springer Berlin / Heidelberg, 2011, vol. 7072, pp. 163–172, 10.1007/978-3-642-25504-5_17. [Online]. Available: http://dx.doi.org/10.1007/-978-3-642-25504-5_17
[22] C. Bartneck, D. Kulic, E. Croft, and S. Zoghbi, “Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots,” *Ínternational Journal of Social Robotics*, vol. 1, no. 1, pp. 71–81, 2009.
[23] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, “Learning motion patterns of people for compliant robot motion,” *Internationl Journal of Robotics Research*, vol. 24, pp. 31–48, 2005.
[24] N. Nguyen, D. Phung, S. Venkatesh, and H. Bui, “Learning and detecting activities from movement trajectories using the hierarchical hidden markov model,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 955–960.
[25] G. Schöne, M. Dose, and C. Engels, “Dynamics of behavior: Theory and applications for autonomous robot architectures,” *Robotics and Autonomous Systems*, vol. 16, no. 2, pp. 213–245, 1995.
[26] T. Kanda, D. Glas, M. Shiomi, and N. Hagita, “Abstracting people’s trajectories for social robots to proactively approach customers,” *Robotics, IEEE Transactions on*, vol. 25, no. 6, pp. 1382–1396, 2009.
[27] L. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
[28] N. Nguyen, D. Phung, S. Venkatesh, and H. Bui, “Learning and detecting activities from movement trajectories using the hierarchical hidden markov model,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 955–960.
[29] T. Duong, H. Bui, D. Phung, and S. Venkatesh, “Activity recognition and abnormality detection with the switching hidden semi-markov model,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 838–845.
[30] T. Van Kasteren, A. Noulas, G. Englebienne, and B. Kröse, “Accurate activity recognition in a home setting,” in *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 2008, pp. 1–9.
[31] M. Gani, H. Sarwar, and C. Rahman, “Prediction of state of wireless network using markov and hidden markov model,” *Journal of Networks*, vol. 4, no. 10, pp. 976–984, 2009.
[32] S. Fine, Y. Singer, and N. Tishby, “The hierarchical hidden markov model: Analysis and applications,” *Machine learning*, vol. 32, no. 1, pp. 41–62, 1998.
[33] M. Bruna, “The benefits of using high-level goal information for robot navigation.”

Appendix

A. Hidden Markov Model

Definitions

A Hidden Markov Model can be described by the following parameters

S_t	State at time t
O_t	Observation at time t
q_i	State i
T	Length of sequence
N	Number of states
K	Number of observations
A	$N \times N$ transition matrix where $a_{ij} = P[S_{t+1} = q_j \mid S_t = q_i]$ and $\sum_{j=1}^{N} a_{ij} = 1$
B	$N \times K$ observation matrix where $b_{ik} = P[O_t = k \mid S_t = q_i]$ and $\sum_{k=1}^{K} b_{ik} = 1$
π	The initial state probability where $\pi_i = P[S = i]$
λ	The model parameters (A, B, π)

Training

To obtain the model parameters we use a procedure called batch learning. A large data set is divided in two sets. One part, the training set is used to determine the model parameters. Then the model is tested using the other set, the test set, to determine the model's performance.

To estimate the transitions probability a_{ij} we count the number of transitions from state i to j in the training set.

$$a_{ij} = \frac{n_{ij}}{\sum_j n_{ij}}$$

Where n_{ij} is the number of transitions from state i to state j in the training set.

In a similar way we can estimate the observation probabilities by counting the number of times k is observed while in state i.

$$b_{ik} = \frac{m_{ik}}{\sum_k m_{ik}}$$

Where m_{ik} is the number of times k was observed in the training set while in state i.

Prediction

To predict the most likely next state we first have to determine the most likely current state. To do this we can use the forward variable α from the forward-backward algorithm described in Rabiner [27]. The forward variable represents the probability of the observation sequence O that ends in state i at given the model parameters λ. $\alpha_i(t) = P[S_t = q_i, O \mid \lambda]$

To estimate the probability of the current state for a given observation sequence and the model parameters we can use a normalized α

$$P\left[S_t = q_i | O, \lambda\right] = \frac{P\left[S_t = q_i, O|\lambda\right]}{P\left[O|\lambda\right]} = \frac{\alpha_t(i)}{\sum_{j=1}^{N} \alpha_t(j)}$$

The forward variable can be computed iteratively using the following formulas

$$\begin{aligned} \alpha_1(i) &= \pi_i \cdot b_i(O_1) \\ \alpha_{t+1}(j) &= \left[\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij}\right] \cdot b_j(O_t) \end{aligned}$$

If we have observations until time T we can calculate $\alpha_i(T)$; to determine the most probable state at time $T+1$ we need $\alpha_i(T+1)$.

Because we don't have an observation at time $T+1$, we assume that all observations are equally likely, so we set $b_j(O_{T+1})$ to $\frac{1}{K}$ for all j, where K is the number of observations. Then the most likely next state j:

$$\begin{aligned} j_{\text{mostprobable}} &= \arg\max_j \left\{ \frac{\alpha_{T+1}(j)}{\sum_{j=1}^{N} \alpha_{T+1}(j)} \right\} \\ &= \arg\max_j \left\{ \frac{\left[\sum_{i=1}^{N} \alpha_T(i) \cdot a_{ij}\right] \cdot \frac{1}{K}}{\sum_{j=1}^{N} \left[\sum_{i=1}^{N} \alpha_T(i) \cdot a_{ij}\right] \cdot \frac{1}{K}} \right\} \\ &= \left\{ \frac{\left[\sum_{i=1}^{N} \alpha_T(i) \cdot a_{ij}\right]}{\sum_{j=1}^{N} \left[\sum_{i=1}^{N} \alpha_T(i) \cdot a_{ij}\right]} \right\} \end{aligned}$$

Because we are interested in the relative probabilities, not in the absolute ones we can simplify the previous expression by removing the normalization step (the denominator) to obtain:

$$j_{\text{mostprobable}} = \arg\max_j \left\{ \left[\sum_{i=1}^{N} \alpha_T(i) \cdot a_{ij}\right] \right\}$$

Parameter updating

There are several methods to update model parameters. One simple way to do this is using an exponentially moving average.

The test set is updated in batches. After a batch of data has been used for prediction that batch is used to construct observation and transition matrices A_{update} and B_{update}. Then the transition matrix is updated using:

$$A_{\text{new}} = A_{\text{old}} \cdot (1 - l) + A_{\text{update}} \cdot l$$

where $l \in [0, 1]$ is the learning rate. The same is done for the observation matrix.

$$B_{\text{new}} = B_{\text{old}} \cdot (1 - l) + B_{\text{update}} \cdot l$$

B. Hierarchical Hidden Markov Model

Definitions

The HHMM can be described by the following parameters

S_t^d State at time t at level d

q_i^d State i at level d

O_t Observation at time t

D Number of layers

N_m Number of states at layer m

K Number of observations

A^m $N_m \times N_m \times N_{m+1}$ transition matrix where

$$a_{ij}^{q_n^d} = P\left[S_{t+1}^{d+1} = q_j^{d+1} | S_t^{d+1} = q_i^{d+1}, S_t^d = q_n^d\right]$$

and

$$\sum_{j=1}^{N} a_{ij}^{a_n^d} + a_{i\ \text{end}}^{a_n^d} = 1$$

B $N \times K$ observation matrix where

$$b_{ik} = P\left[O_t = k | S_t^D = q_i^D\right]$$

and

$$\sum_{k=1}^{K} b_{ik} = 1$$

π The initial state probability where $\pi_i^{q_n^{d-1}} = P\left[S_t^d = q_i^d | S_t^{d-1} = q_n^{d-1}\right]$

Training

Just as for the HMM the data is divided in a training and a test set. The training set is used to estimate model parameters. To estimate the transition matrix we use the same method as for the HMM. We count the number of transitions from i to j given the higher state n. Then for each element in the transitions matrix we set:

$$a_{ij}^{q_n^d} = \frac{c_{ij}^{d_n}}{\sum_j c_{ij}^{d_n}}$$

Where c_{ij}^d is the number of transitions from state i to state j on level $d+1$ while level d is in state n. To construct the observation matrix we use the same method as for the HMM.

Prediction

Similar to future state prediction for the HMM we first need to determine the most probable current state. Because of the hierarchical structure we need a number of steps to do this. Figure 27 shows the most probable state inference procedure schematically.

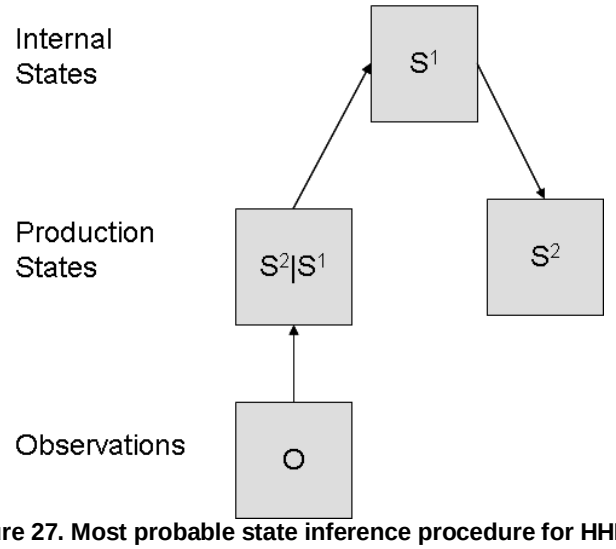

Figure 27. Most probable state inference procedure for HHMM

Again, just as for the HMM, we start with observations. From these we determine the most probable states at the lower level (level 2) for each possible state at the parent level (level 1). We use this to determine the most likely state at level 1 for each time step. Finally we can determine the most likely state at level 2.

This procedure is described by Fine et al. [32] as the generalized Viterbi algorithm. For this algorithm they define three variables for each level. For the variables for the states on level 1 we can omit dependencies on the layer above it (level 0) because there is none.

- $\delta\left(t, t+k, q_i^d, q_h^{d-1}\right)$ is the likelihood of the most probable state sequence generating $o_t \ldots o_{t+k}$ assuming it was solely generated by a recursive activation that started at time t from state q_h^{d-1} and ended at q_i^d which returned to q^{d-1} at time $t+k$.
- $\psi\left(t, t+k, q_i^d, q_h^{d-1}\right)$ is the most probable state to be activated by q_h^{d-1} before q_i^d. If such a state does not exist ($o_t \ldots o_{t+k}$ was solely generated by q_i^d) we set $\psi\left(t, t+k, q_i^d, q_h^{d-1}\right) \stackrel{\text{def}}{=} 0$.
- $\tau\left(t, t+k, q_i^d, q_h^{d-1}\right)$ is the time step at which q_i^d was most probable to be called by q_h^{d-1}. If q_i^d generated the entire subsequence we set $\tau\left(t, t+k, q_i^d, q_h^{d-1}\right) = t$.

All details about updating these variables can be found in Bruna [33] or in Fine et al. [32]. Because this algorithm uses the probability of sequences of states, not individual states, determining the current state and predicting the next state can be done at the same time.

Just as for the HMM we have an observation sequence of length T and we want to predict both the production and the internal state at time $T+1$. Because we don't know the observation at time $T+1$, we assume again that all observations are equally likely: $b_j\left(O_{T+1}\right) = \frac{1}{K}$ where K is the number of observations.

Prediction of internal states

To find the most probable state sequence at level 1 we first determine the most probable final state:
$S_{\text{end}}^1 = \arg\max_{q_i^1}\left\{\delta\left(1, T+1, q_i^1\right)\right\}$
Then we can find the most probable transition time using:
$t_{\text{switch}} = \tau\left(1, T_1, S_{\text{end}}^1\right)$.
The most probable internal state before the switching time is

$q^1 = \psi\left(1, t_{\text{switch}}, S^1_{\text{end}}\right)$.

We repeat this procedure until we have obtained the complete state sequence at level 1.

Prediction of production states

Now that we have the most probable internal state sequence we can find the most probable production state sequence. Just as for the internal state we first find the most probable production state at time $T+1$.

$S^2_{T+1} = \arg\max_{q^1_i} \left\{\delta\left(1, T+1, q^2_i, S^1_{T+1}\right)\right\}$

Then we can determine the most probable previous production state using:

$S^2_{t-1} = \psi\left(t_{\text{switch}}, t, S^2_t, S^1_t\right)$

We repeat this procedure until we have obtained the complete state sequence at level 2.