A Neural Approach for Robot Navigation based on Cognitive Map Learning

Wenjie Yan
University of Hamburg
Department of Computer Science
Knowledge Technology
Vogt-Kölln-Straße 30
D - 22527 Hamburg, Germany
yan@informatik.uni-hamburg.de

Cornelius Weber
University of Hamburg
Department of Computer Science
Knowledge Technology
Vogt-Kölln-Straße 30
D - 22527 Hamburg, Germany
weber@informatik.uni-hamburg.de

Stefan Wermter
University of Hamburg
Department of Computer Science
Knowledge Technology
Vogt-Kölln-Straße 30
D - 22527 Hamburg, Germany
wermter@informatik.uni-hamburg.de

***Abstract*—This paper presents a neural network architecture for a robot learning new navigation behavior by observing a human's movement in a room. While indoor robot navigation is challenging due to the high complexity of real environments and the possible dynamic changes in a room, a human can explore a room easily without any collisions. We therefore propose a neural network that builds up a memory for spatial representations and path planning using a person's movements as observed from a ceiling-mounted camera. Based on the human's motion, the robot learns a map that is used for path planning and motor-action codings. We evaluate our model with a detailed case study and show that the robot navigates effectively.**

I. Introduction

In recent years, robot navigation based on cognitive mapping has become an important research topic. A cognitive map is a mental model that represents information of an environment with features and relationships similar to humans and animals realizing navigation [1]. Unlike geometry-based models [2], [3], which rely on accurate measurements, a cognitive model works on a higher abstract level and is robust against noise. For example, a person can find the destination by simple instruction like "left to the post office" without precise position information. He can plan his walking path easily according to the cognitive map, and when the environment changes, he can also adjust his path plan quickly by modifying the spatial relationships in the cognitive map. Hence, a cognitive map can allow a robot flexible navigation and adaptation in a real complex environment. This adaptivity is desired to integrate a robotic agent into an ambient assistant living (AAL) setup that accommodates the needs of users in different contexts and environments [4]. A robot and a person can be localized robustly in an AAL lab for instance with a ceiling-mounted camera [5]. Based on the topological structure of the free space in the room, a robot can plan its path and approach a person. Since the location is represented by the states in the cognitive map rather than precise coordinate information, the navigation can work with the raw camera image without calibration.

To build up a cognitive map, a robot usually has to explore the unknown environment actively [2], [3]. While such exploration is relative easy to achieve for a person, a robot requires a longer learning period and may suffer from possible collisions. Considering that an animal may learn new behavior through observing others' actions [6], a robot could accelerate the spatial learning by observing a person's navigation behavior. As a self-protection behavior, a person avoids dangerous areas or collisions unconsciously while moving [7]. For this reason, where a person walks, it should be safe also for the robot to move. However, since a robot and a person may differ from their behavior, for example a person can sit on a sofa but a robot cannot, 1) some reflexive behavior is essential and 2) interactive robotic exploration is needed to protect the robot and to convert the spatial knowledge from the human to the robot itself.

Cognitive maps have been researched for a long time in various areas. Since place cells have been found in the hippocampus of rats [8], where the activities of these neurons are related to the location in an environment, it has been shown that a cognitive map is presented by the place cells in the hippocampus and grid cells in the entorhinal cortex [9]. Based on these discoveries, many neurobiological-inspired models have been proposed for spatial learning and robot navigation. A cognitive map can be represented in different ways, for example as a topological map [10], [11], or by a continuous attractor network [12], [13], [14], etc. Furthermore, by using a self-growing mechanism [15], Toussaint [16] developed a model that can represent a map with a dynamic size, which is flexible for exploring an unknown environment.

To acquire behavior-based robot navigation, Weiller [17] proposed an unsupervised learning method to learn reflexive behavior associated with state transitions. Weber [18] and Witkowski [19] present neural network models that learn associations between adjoining states and the action that links them. A neural fields model has been seen as a simple but effective way to model motion perception [20] and robot behavior [21], [22], [16], [23]. In addition, closed-loop control models for other behaviors, e.g. arm reaching, may apply similar methods of planning [24].

This paper presents a neurocomputational model on the basis of Cuperlier's and Toussaint's frameworks [22], [16] for behavior-based robot navigation in an indoor home environ-

ment. Compared with other similar works, our model 1) learns a cognitive map quickly by observing a person's location distribution (method described in [5]) from a ceiling-mounted camera view and 2) realizes robust robot navigation in a real complex rather than simulated environment. A humanoid robot will control its orientation and walk to a person according to the learned map. During the navigation, an interaction model provides the robot with a reflex behavior and enables it to adjust the cognitive map obtained from observing the person, which provides a safer and faster way for spatial learning than active exploration by the robot alone.

Our model consists of three layers of networks: 1) a growing neural gas map (GNG) [15] for spatial learning and robot path planning, 2) a Sigma-Pi network [25] trained to represent the inverse control model and 3) a dynamic neural fields (DNF) model [21] that controls the navigation behavior. Using a person's movement as input, the GNG will learn a topological map of the free space in a room. The path planning will then be done based on the spatial representation on the GNG and rewards spreading from the person's position to the robot's position. According to the current and the desired next state representations in the GNG, the Sigma-Pi network will produce an action code which is trained during the room mapping phase. Stimulated by an action coding from the Sigma-Pi network, the DNF will update the activities which present the expected pose for the next step. The robot will thereby navigate with human-like behavior, approaching the target by adjusting its orientation continuously instead of changing the direction suddenly. In addition, a robot-environment interaction method is included for adjusting the cognitive map on the basis of a reflex-like behavior for obstacle avoidance. Details about these models will be described in the following sections. Our model is implemented and tested on a humanoid robot in an AAL lab and the results are evaluated.

II. Methods

The overall architecture of our model is illustrated in Figure 1. The target person's and the robot's location will be estimated by two sets of particles from the ceiling-camera view as described in [5], which form the input to our model. Particle filters [26] are an approximation method that represents a probability distribution of an agent's state s_t by a set of particles. Each particle consists of a feature vector ξ, in our case describing position information, and a weight value w. The feature vector of robot's particles contains the x, y coordinate information and the orientation value o, i.e. $\xi^r = \{x, y, o\}$, because we can predict the orientation based on the action the robot will execute, while the feature vector of particles for person localization contains only the x and y ($\xi^p = \{x, y\}$) since the direction of a person's motion is hard to predict.

Our model works in two phases: *exploration* and *navigation*. In the exploration phase, a cognitive map will be learned in the GNG using a person's location represented by the particles. We assign an input activation of the Sigma-Pi network for each

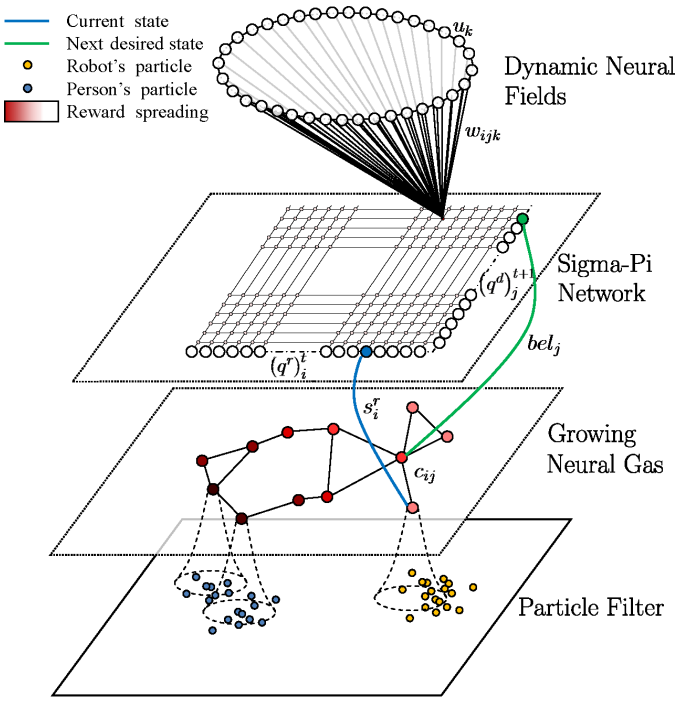

Fig. 1. Architecture

connection in the GNG, which learns the inverse control model of the robot's movement. According to the inverse control model and the robot's orientation estimation obtained from the raw camera image, the robot will control itself by adjusting its walking direction. Since both of them are based on the same pixel coordinate system, it would not matter if the camera was rotated before building up the cognitive map.

In the navigation phase, the robot's and the person's location will first be represented by the neuron activities in the learned GNG. Based on this representation, path planning will be done by spreading a reward signal recursively from the person's position to the robot's position with an exponential decrease. The robot's next state representation will be selected by searching for the neighborhood neuron with the highest reward signal. The Sigma-Pi network will produce appropriate action coding with respect to the robot's current and the next state representation, and the DNF will be stimulated and control the robot's motion. During navigation, robot-environment interaction provides the robot with reflex-like obstacle avoidance ability and adjusts the GNG model by adapting a lateral connection weight c_{ij} in the GNG. We will describe the details of each model in the following sections.

A. Spatial Learning

A cognitive map is learned in a GNG [15] using a person's position, obtained from its particle distribution, as input. A GNG consists of a set A of neurons, each associated with a feature vector $v = \mathbf{R}^n$ (in our case the x, y coordinate information on an image: $v = \{x, y\}$), and a set N of connections to describe the relations between neurons in A. Each neuron is assigned an age factor which can increase incrementally. Using

a competitive Hebbian learning rule, neurons and connections will be allocated or updated dynamically, and will be deleted when the connection age is over a threshold $a_{\max}$ or a neuron is isolated. The learning algorithm is shown in algorithm 1.

Algorithm 1 Growing Neural Gas [15]

1. Start with two neurons a and b with random weights v_a and v_b in $\mathbf{R}^n$.
2. Generate an input signal ξ according to $P(\xi)$.
3. Find the nearest unit i^* and the second-nearest unit i^{**}.
4. Increment the age of all edges emanating from i^*.
5. Add the squared distance between the input signal and the nearest unit in input space to a local counter variable:

$$\Delta\text{error}_{\mathrm{i}^*} = ||v_{\mathrm{i}^*} - \xi||^2$$

6. Move i^* and its direct topological neighbors towards η by fractions ϵ_b and ϵ_n with respect to the total distance:

$$\begin{aligned} \Delta v_{\mathrm{i}^*} &= \epsilon_b(\xi - v_{\mathrm{i}^*}) \\ \Delta v_n &= \epsilon_n(\xi - v_n) \quad \text{for all direct neighbors } n \text{ of } \mathrm{i}^* \end{aligned}$$

7. If i^* and i^{**} are connected by an edge, set the age of this edge to zero. If such an edge does not exit, create it.
8. Remove edges with an age larger than a_{max}. If this results in points having no emanating edges, remove them as well.
9. If the number of input signals generated so far is an integer multiple of a parameter λ, insert a new neuron as follows:
   - Determine the neuron q with the maximum accumulated error.
   - Insert a new neuron r halfway between q and its neighbor f with the largest error variable:

$$w_r = 0.5(w_q + w_f)$$

   - Insert edges connecting the new neuron r with neurons q and f, and remove the original edge between q and f.
   - Decrease the error variables of q and f by multiplying them with a constant α. Initialize the error variable of r with the new value of the error variable of q.
10. Decrease all error variables by multiplying them with a constant d.
11. If a stopping criterion (e.g., net size or some performance measure) is not yet fulfilled go to step 2.

To localize the robot and the person in the GNG and for the further robot navigation, we assign each neuron $\{i\}$ in the map two activities: s_i^r for the robot and s_i^p for the person, which are stimulated by the person's and the robot's particles. During the spatial learning phase, the person's activity s_i^p is also used to adapt the GNG. Each neuron has a Gaussian-form observation area with variance σ_x^2, σ_y^2 in x, y directions (both are set as 20 pixels on the image), and the activity is computed based on the position ξ_j^p and weight w_j^p of the person's particles $\{j\}$:

$$s_i^p = a\sum_j w_j^p e^{-\left(\frac{\left(v_{i,0}-\xi_{j,0}^p\right)^2}{2\sigma_x^2}+\frac{\left(v_{i,1}-\xi_{j,1}^p\right)^2}{2\sigma_y^2}\right)} \tag{1}$$

where a is a scaling factor, $v_{i,0}$ and $v_{i,1}$ are the x and y position of neuron i in the GNG. Similarly, $\xi_{j,0}^p$, $\xi_{j,1}^p$ denote the position of a person's particle j. The winner neuron with the highest activities will be selected as follows:

$$\mathrm{i}^* = \arg\max_i(s_i^p) \tag{2}$$

and the accumulative error of the winner neuron $\{\mathrm{i}^*\}$ will be updated according to the following equation:

$$\Delta\text{error}_{\mathrm{i}^*} = \sum_j w_j^p ||v_{\mathrm{i}^*} - \xi_j^p||^2 \tag{3}$$

These error values are used for the growth of the network (see Algorithm 1). The feature vectors of the winner neuron $\{\mathrm{i}^*\}$ and its neighborhood neurons $\{n\}$, which are connected to the winner neuron in the GNG, will then be updated with:

$$\begin{aligned} \Delta v_{\mathrm{i}^*} &= \epsilon_{\mathrm{w}} \sum_j w_j^p(\xi_j^p - v_{\mathrm{i}^*}) \\ \Delta v_{\mathrm{n}} &= \epsilon_{\mathrm{n}} \sum_j w_j^p(\xi_j^p - v_{\mathrm{n}}) \end{aligned} \tag{4}$$

here ϵ_{w} and ϵ_{n} are two fixed learning rates and j is the index of particle. Since the "original model" takes input stimuli that are independent in time one after the other, i.e. the distribution is stationary, an over-learning problem may occur when the person stays at one position for a long time. We therefore only update the network when the person's movement is detected. As a result of learning with Eq. (4), the GNG units will topologically represent the entire space where the person has moved.

Similar to Eq. (1), the robot's activities s_i^r are computed as:

$$s_i^r = a\sum_j w_j^r e^{-\left(\frac{\left(v_{i,0}-\xi_{j,0}^r\right)^2}{2\sigma_x^2}+\frac{\left(v_{i,1}-\xi_{j,1}^r\right)^2}{2\sigma_y^2}\right)} \tag{5}$$

where w_j^r is the weight of robot's particles $\{j\}$. Regarding that a person can move differently than a robot, some connections learned from a person's movement might not be suitable for a robot. A robot-environment interaction is essential in this case to enable the robot to adapt its navigation strategy. We therefore define a connection weight $c_{ij} \in [0, 1]$ for each connection to indicate whether one link between two neurons is suitable for the robot to move along. The higher c_{ij} is the easier is the connection for the robot to walk through. When an obstacle is detected or the robot has difficulties walking further, c_{ij} will be decreased and may reach zero. When a connection is built, its connection weight c_{ij} will be initialized to 1 and adapted during the robot navigation. Details about this adaptation will be described in the navigation section.

B. Path Planning

Based on the person's location, each neuron $\{j\}$ in the GNG will receive a reward signal r_j spreading from the person's state representations iteratively with an exponential decrease. We give first the winner neuron i^* (see Eq. (2)) an initial reward r, i.e. $r_{\mathrm{i}^*}^p(0) = r$, and the reward signal will spread to the neighbor neurons (listed in n) iteratively with a discount factor λ:

$$r_j^p(t+1) = \lambda r_i^p(t), \quad \text{for } j \in n \text{ and } r_j^p(t) = 0 \tag{6}$$

For each step, the neighborhood list n will be updated for the next iteration as follows:

$$\begin{aligned} n' &\leftarrow i \quad \text{if } i \text{ connects with a neuron in } n \\ &\qquad \text{and } r_i^p(t+1) = 0 \\ n &= n' \end{aligned} \tag{7}$$

After the reward signal has spread over the entire GNG map, the robot plans its action for the next step by calculating the next position it should reach. Assume that the robot's position is represented by a group of neurons α in the GNG. The next possible position should be among the neighborhood neurons that connect with neurons in α directly. We calculate a belief value bel_j of these neighborhood neurons j, which connect with neurons $i \in \alpha$, as follows:

$$\widetilde{bel}_j = \sum_{i \in \alpha} c_{ij} s_i^r r_j \tag{8}$$

and normalize the values with a soft-max activation function:

$$bel_j = \frac{e^{\widetilde{bel}_j}}{\sum_i e^{\widetilde{bel}_i}} \tag{9}$$

where s_i^r is neuron activity of the robot detection and c_{ij} is the connection weight. The higher bel_j is, the more desirable it is for the robot to be at this position in the next step.

C. Action Coding in Sigma-Pi Network

According to the current and the desired next state representations in the GNG, the robot knows where it is and where to go. However, to realize navigation, the robot also needs to know how to reach the next position. A Sigma-Pi network therefore learns an inverse control model. It receives as input from the GNG a pair of neural activation representing the current and the desired next state (hence, the multiplicative input of the Sigma-Pi units), and returns the suggested action codes to the DNF. Since only the layer connection weights connecting to the DNF need to be trained, the action code can be learned online using a simple updating rule.

As shown in Figure 1, the Sigma-Pi network has two dimensions of input neurons. The neurons in one dimension receive the neuron activities of the robot's current state representation s^r and those in the other receive the belief value bel of the robot's next desired state. Both dimensions have the same number of neurons as in the GNG. During the robot navigation, the units in the Sigma-Pi network connecting to the robot's current and the next desired state representation will be stimulated and the inverse control model will produce appropriate action codes that lead the robot to reach the next state. The inputs of the Sigma-Pi network in its two input dimensions are: $(q^r)_i^t = s_i^r$ and $\left(q^d\right)_j^{t+1} = bel_j$.

We index the neurons in the current state layer with i and in the next desired state layer with j, and the output of the Sigma-Pi network with k. The input I_k to the k-th neuron in the DNF can be computed as:

$$I_k = \sum_{i,j} w_{ijk} (q^r)_i^t (q^d)_j^{t+1} \tag{10}$$

where $(q^r)_i^t$, $(q^d)_j^{t+1}$ are the neuron activities in the input layers and w_{ijk} are the Sigma-Pi connection weights, which are initialized with 0. The size of these layers will be adapted when adding or removing neurons in the GNG. For example, when a neuron is inserted in the GNG, two input units (one for each input layer) will be added and be associated with the new neuron in the GNG. Suppose that there are N neurons in the GNG and 36 neurons in the DNF, the total number of Sigma-Pi connections $\{w_{ijk}\}$ is $36N^2$.

As we can see, there are two possible orientations for each link and for each firing combination i, j there are 36 connection weights that can be trained to represent an action responsible for moving the robot to follow the link in the GNG. We therefore use the person's movement to learn the action codes by updating the GNG. Assume that the neurons $\{i\}$, $\{j\}$ of connection c_{ij} adjust their positions in the GNG, or a new neuron $\{j\}$ is insert in the GNG and a new link c_{ij} connecting with the neuron $\{i\}$ and the new neuron $\{j\}$ is allocated. Then the Sigma-Pi weights w_{ijk}, w_{jik} of this connection will be updated as follows:

1) According to the current position (x_i, y_i) of neuron $\{i\}$ and (x_j, y_j) of neuron $\{j\}$ based on the image coordinate, we calculate both possible orientations o_{ij} and o_{ij} of connection c_{ij} using inverse trigonometric functions:

$$\begin{aligned} o_{ij} &= \arctan\left(\frac{y_j - y_i}{x_j - x_i}\right) \\ o_{ij} &= o_{ij} + \pi \qquad \text{if } x_j - x_i < 0 \\ o_{ji} &= o_{ij} + \pi \end{aligned} \tag{11}$$

2) Two bumps of activation with the size of the DNF will be created in the shape of a circular normal distribution around the link orientation:

$$\begin{aligned} p_{ijk} &= \frac{e^{\kappa \cos(\frac{k \cdot 10\pi}{180} - o_{ij})}}{2\pi I_0(\kappa)} \\ p_{jik} &= \frac{e^{\kappa \cos(\frac{k \cdot 10\pi}{180} - o_{ji})}}{2\pi I_0(\kappa)} \end{aligned} \tag{12}$$

where p_{ijk} is the k-th neuron of the action coding for orientation o_{ij}, κ is a constant and $I_0(\kappa)$ is the modified Bessel function of order 0 [27]:

$$I_0(\kappa) = \frac{1}{\pi} \int_0^{\pi} e^{\kappa \cos(\theta)} \mathrm{d}\theta \tag{13}$$

3) Minimize the reconstruction errors $||p_{ijk} - w_{ijk}||^2$ using gradient descent:

$$\begin{aligned} \Delta w_{ijk} &= \eta(p_{ijk} - w_{ijk}) \\ \Delta w_{jik} &= \eta(p_{jik} - w_{jik}) \end{aligned} \tag{14}$$

where η is a fixed learning rate.

This defines the weights w_{ijk} that are used to compute the input I_k to the neural field (see Eq. (10)), which encodes the desired orientation of the robot.

D. Behavior Control in Dynamic Neural Fields

The DNF is a biologically-inspired model of the neural dynamics in cortical tissues [28], which is of interest in the robotic area to generate dynamic behavior [21], [22]. In our work, a one-dimensional ring-form DNF with 36 neurons is applied to control the robot's navigation behavior over time. Based on the action codes from the Sigma-Pi network, the DNF will dynamically integrate the activities, which represent the suggested orientation for the next step. The DNF will adjust the robot's motion control producing a smooth and natural orienting behavior.

In the DNF, each neuron k has a membrane potential u_k and lateral connections n_{kj} with other neighbor neurons j. A bump of neurons' activations is generated and moved based on the stimuli from the Sigma-Pi network. The bump stabilizes itself by associating with the neighborhood neurons, which leads to more robustness in the model. The membrane potential is updated as follows:

$$\tau \Delta u_k = -u_k + \sum_{j=1}^{36} n_{kj} f(u_j) + I_k + h \tag{15}$$

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 n_{kj} to describe the lateral interaction of neurons:

$$n_{kj} = \beta e^{-\frac{(k-j)^2}{2\sigma^2}} - c \tag{16}$$

where β is a scaling factor, σ^2 a variance, k, j the positions of neurons and c is a positive constant. The function $f(u)$ is a sigmoid transfer function of a single neuron:

$$f(u_j) = \frac{1}{1 + e^{-u_j}} \tag{17}$$

The robot's desired orientation O^d is calculated using vector averaging:

$$\hat{v}_k = \begin{pmatrix} \sum v_{kx} \\ \sum v_{ky} \end{pmatrix} = \begin{pmatrix} \sum u_k \sin(\frac{10k}{180}\pi) \\ \sum u_k \cos(\frac{10k}{180}\pi) \end{pmatrix} \tag{18}$$

$$O^d = \begin{cases} \arctan\left(\frac{\hat{v}_{ky}}{\hat{v}_{kx}}\right) & \text{if } \hat{v}_{kx} > 0 \text{ and } \hat{v}_{ky} > 0 \\ \arctan\left(\frac{\hat{v}_{ky}}{\hat{v}_{kx}}\right) + \pi & \text{if } \hat{v}_{kx} < 0 \\ \arctan\left(\frac{\hat{v}_{ky}}{\hat{v}_{kx}}\right) + 2\pi & \text{if } \hat{v}_{kx} > 0 \text{ and } \hat{v}_{ky} < 0 \end{cases} \tag{19}$$

We control the robot's navigation by giving it a differential orientation command:

$$\Delta O = \begin{cases} -c & \text{if } O^d - O^p > d \\ c & \text{if } O^d - O^p < -d \\ 0 & \text{else} \end{cases} \tag{20}$$

where O^p is the robot's estimated orientation from the particle filter, c is a rotation speed parameter and d is a constant threshold.

E. Robot-Environment Interaction

We consider that the GNG map is built using a person's actual movement, but some of the positions (table, sofa, for instance) might not be accessible for the robot. An interaction mechanism is essential in this case to provide the robot with some "reflex" behavior to avoid these positions and to adapt the navigation strategy online. The connection weights c_{ij} in the GNG are used here to indicate how "easy" the robot can follow that link. The adaptation of c_{ij} depends on the interaction with the environment. Feedback is supplied by two sonar sensors mounted on the chest of the robot. Both sensors can detect the distance to obstacles between 30 and 80 cm. The higher the sensor value, the larger distance is the robot to the object. The weights adaptation is processed as follows:

$$\Delta c_{ij} = \tau(G(s_1, s_2) - c_{ij}) \tag{21}$$

where τ is a learning rate and $G(s_1, s_2)$ is a non-linear function using two sonar sensor values s_1, s_2:

$$G(s_1, s_2) = \frac{1}{1 + e^{(s_1 + s_2 - c)}} \tag{22}$$

here c is a constant offset. When the robot approaches an obstacle, the connection weight c_{ij} will be decreased and even converge to zero when the obstacle gets too close.

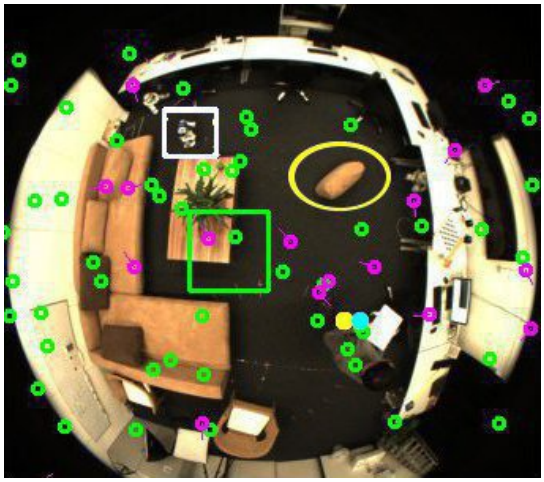

Fig. 2. Experimental setup from the ceiling-mounted camera view. The pink circles with a short bar are the particles for the robot position and orientation and the green particles are the particles for the person. When the person is localized, his location will be estimated by a green bounding box. The white bounding box shows where the robot is and the yellow eclipse shows an obstacle on the ground. The cyan and the yellow filled circles are the first winner and the second winner neuron in the GNG (see text for details).

III. Experimental Results

This section presents a test case of how our model performs in the real world. As shown in Figure 2, a ceiling-mounted camera is used as input for localization and navigation to test our algorithm in a home laboratory (for details about the localization model please see [5]). A fish-eye lens is applied to get a wide field of view of the whole room with a single

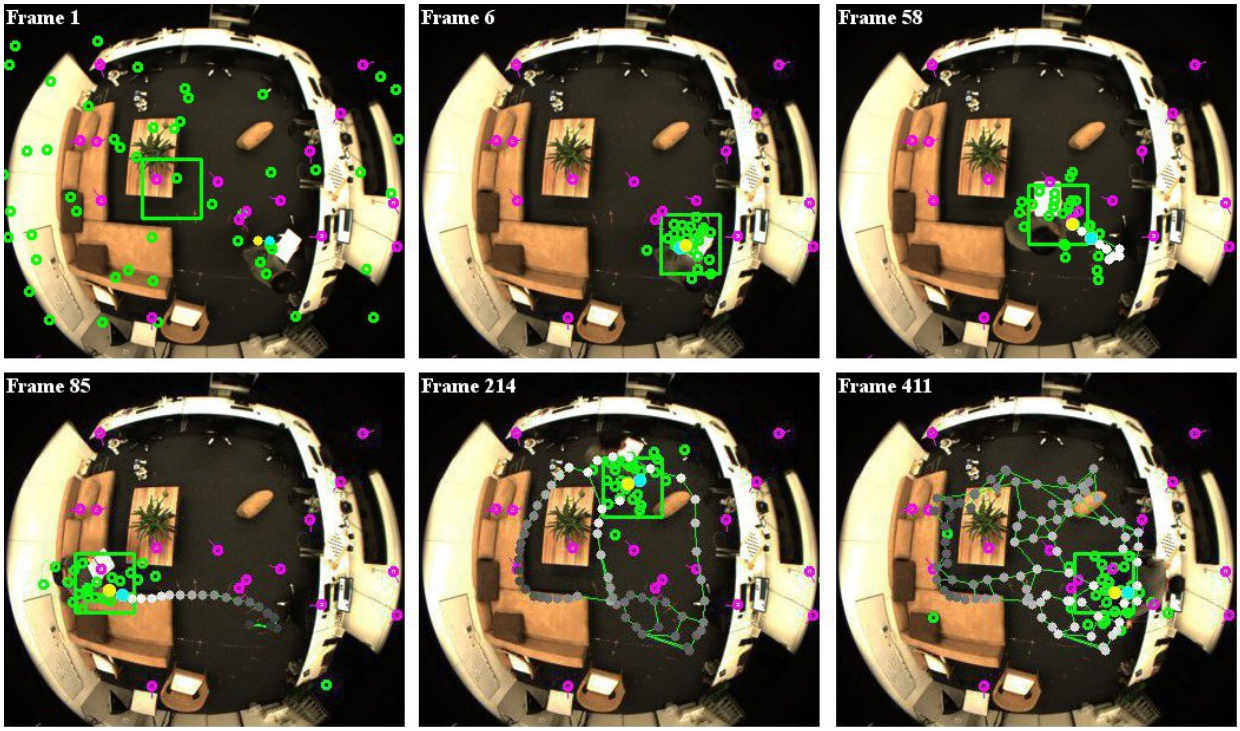

Fig. 3. Learning a cognitive map.
The topological map grows when a person explores the room, and the grey-scaled brightness shows the reward information spreading from the person's location.

camera, however, at the price of strong image distortion. The use of position representation in the cognitive map instead of the coordinate information avoids possible coordinate transformation errors. A humanoid Nao robot is used for navigation (labeled with a white bounding box in Figure 2). An obstacle (labeled with a yellow ellipse) is placed on the floor after the spatial learning phase to test the interaction model using Nao's sonar sensors as input.

Our experiments are split into two parts. We first introduce how we train the cognitive map by observing a person's movement. After that, the robot approaches a person from an arbitrary location in the room based on this learned map[1]. The sonar sensors can detect furniture during the navigation to avoid collision automatically. Details are described in the following sections.

A. Learning a Cognitive Map

The GNG is initialized with two neurons linked with each other with a connection at the beginning of learning (Figure 3 Frame 1). When a person enters the room, she will be detected by the localization model. The winner (yellow) and the second winner (cyan) neurons will be selected, which are the closest to a person's location (Figure 3 Frame 6). The winner and it's neighborhood neurons will be drawn to the person's position and new neurons will be inserted (Frame 58). The GNG will grow automatically when a person explores the room (Frame 85, Frame 214), until all the free space has been visited (Frame 411 where many neurons are placed). The gray-scaled brightness of the neurons indicates the reward spreading from the person's location. The brighter the color is, the higher reward this neuron has. The reward information helps the robot for path planning, which is described in the next section.

[1]For a demonstration please see the video at:
http://www.informatik.uni-hamburg.de/WTM/material/videos/SN1.avi

B. Spatial Navigation

As we can see, the cognitive map contains the position information of most of the free space in the room. When the navigation task starts, the robot will be localized by the particle filter (see Figure 4 at Frame 502, most particles are at the robot's position and the estimated orientation is visualized with a yellow bar) and the position will be represented by the neuron activities. Based on this map, the robot can move to the person from an arbitrary position by finding the next state with the highest reward signal. Based on the current state representation of the robot and the learned map, the robot will find the next state representation by finding the neighborhood neurons with the highest rewards. The Sigma-Pi network will then produce the action codes to the DNF with respect to the current and the next state representations in the GNG. We visualize the neuron activities of the DNF by a red circle surrounding the robot's position with a basic radius of 15 pixels where activation are zero. An activation bump is built up which helps the robot to estimate the desired orientation and control the navigation behavior by controlling the orientation. During the robot's movement, the state representations are

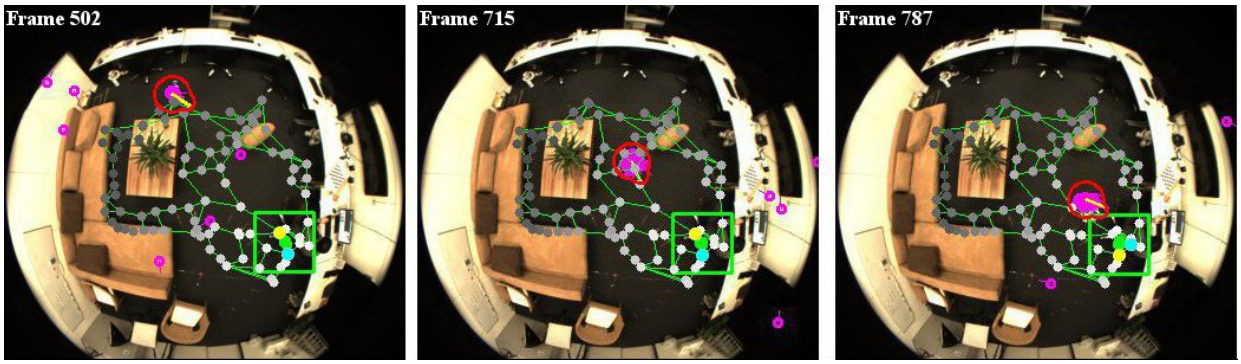

Fig. 4. Sequence of Navigation

Based on the current location of the robot (pink particles) and the person (green particles and the green bounding box), the robot plans its next position by searching for the more activated neighborhood neurons (displayed brighter). The DNF is stimulated and produces the desired robot orientation, which is visualized by a red circle surrounding the robot. The robot controls then its orientation and approaches the person.

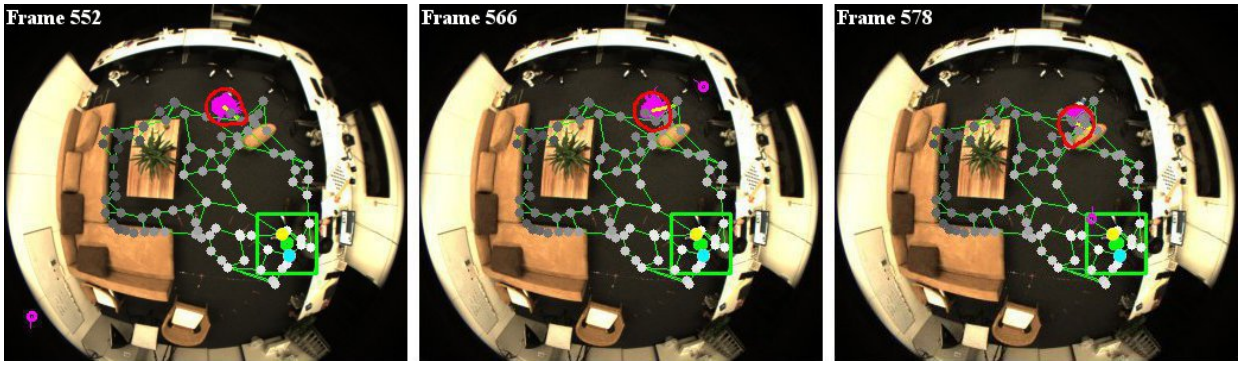

Fig. 5. Robot environment interaction

When the robot sensors detect an obstacle, the reflex behavior is triggered and the robot starts walking slowly and turning around to avoid the obstacle. The current activation bump (see the red circle) weakens (Frame 566) and another bump builds up (Frame 578) to adjust the path planning.

changing and the activation bump will be updated to the new orientation. The robot updates its action (it starts turning left) and will walk towards the person. When the robot gets close to the person, the navigation will be achieved and the robot will stop walking (Frame 787). Because of the dynamic behavior of the DNF, the robot will adjust its orientation slowly and walk in a natural way instead of reacting suddenly to possibly noisy measurements.

C. Interaction

This section presents a test case of the robot environment interaction model by a simple obstacle avoidance. As shown in Figure 5, when an obstacle is detected by the robot's sonar sensors (Frame 552), the robot stops. A reflex control model compares the two sonar sensor values and rotates the robot slowly. Meanwhile, the connection weights c_{ij} of the currently used connection decreases and the activation bump of the DNF (see the red circle at Frame 566) shrinks and the robot replans its navigation policy and changes the desired orientation by building up a new activation bump (Frame 566 and 578). Since the robot is close to the obstacle, the robot will change its orientation slowly, until no obstacle is detected by the sonar sensors anymore. Then the robot starts walking again in the new direction.

IV. Discussion

As we have demonstrated in the experiments, a cognitive map can be generated by observing a person's movement and a robot can plan and navigate successfully in a real environment based on the learned map. We use a GNG to simulate place cells by clustering the location information with neuron activities. Compared with a self-organizing map, a GNG has the advantage of adapting its size as well as its topology efficiently.

The robot has a mixed navigation behavior using pure behavior-based and pure path-based navigation: it does not choose simply the shortest trajectory without considering the difficulty for the robot to walk through. This enables the robot to walk smartly in a complex environment and escape for instance from a U-shape "trap" with a quite natural behavior. Moreover, because the cognitive map is learned by the person's movement and no calibration is needed for the camera, this system is very easy to install and the robot navigation can be done with little exploration by the person.

The quality of the cognitive map will influence the navigation performance significantly. When an area in the map is not explored by the person, the robot will not walk into this area, since it will not know how to approach the person if

it starts navigating into this area. To avoid this problem, we consider developing a self-exploration method that allows the robot to complete its map when no information exists, and to improve the spatial learning by using different methods for map building. In addition, although our model works with the person's and the robot's location obtained from the ceiling-mounted camera, the cognitive map learning could also be achieved using other sensors. It would be helpful to extend our model with different sensors for a wider scope of application. Therefore, we are currently refining the robot's input module to use the robot's internal cameras and the velocity estimation with the internal gyroscope.

V. Conclusion

We have presented a cognitive neural network architecture that controls a robot to follow a person successfully in a realistic home-like environment. The architecture combines localization and mapping with planning and navigation. The desired behavior is perceptually supported by the use of a ceiling camera, which also allows using a person's movements to learn the room topology quickly. Because the position is encoded as an abstract state representation, neither explicit coordinate transformation nor camera calibration is needed. A behavior-based control system increases the robustness and allows the robot to navigate naturally and effectively.

Acknowledgment

The research leading to these results is part of the KSERA project (http://www.ksera-project.eu) funded by the European Commission under the 7th Framework Programme (FP7) for Research and Technological Development under grant agreement n°2010-248085.

References

[1] G. Gron, A. Wunderlich, M. Spitzer, R. Tomczak, and M. Riepe, "Brain activation during human navigation: gender-different neural networks as substrate of performance," *Nature Neuroscience*, vol. 3, pp. 404–408, 2000.

[2] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the 18th National conference on Artificial Intelligencem AAAI'02*, 2002, pp. 593–598.

[3] A. Diosi and L. Kleeman, "Advanced sonar and laser range finder fusion for simultaneous localization and mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2004*, vol. 2, 2004, pp. 1854–1859.

[4] P. L. Emiliani and C. Stephanidis, "Universal access to ambient intelligence environments: Opportunities and challenges for people with disabilities," *IBM Systems Journal*, vol. 44, no. 3, pp. 605–619, 2005.

[5] W. Yan, C. Weber, and S. Wermter, "A hybrid probabilistic neural model for person tracking based on a ceiling-mounted camera," *Journal of Ambient Intelligence and Smart Environments*, vol. 3, pp. 237–252, 2011.

[6] G. Rizzolatti and L. Craighero, "The mirror-neuron system," *Annu. Rev. Neurosci.*, vol. 27, pp. 169–192, 2004.

[7] E. Harmon-Jones, "Neural bases of approach and avoidance," in *Handbook of Self-Enhancement and Self-Protection*, C. S. Mark D. Alicke, Ed. New York, NY, USA: The Guilford Press, 2011, pp. 23–48.

[8] J. O'Keefe and D. H. Conway, "Hippocampal place units in the freely moving rat: Why they fire where they fire," *Experimental Brain Research*, vol. 31, pp. 573–590, 1978.

[9] F. Sargolini, M. Fyhn, T. Hafting, B. L. McNaughton, M. P. Witter, M.-B. Moser, and E. I. Moser, "Conjunctive representation of position, direction, and velocity in entorhinal cortex," *Science*, vol. 312, no. 5774, pp. 758–762, 2006.

[10] L.-E. Martinet, D. Sheynikhovich, K. Benchenane, and A. Arleo, "Spatial learning and action planning in a prefrontal cortical network model," *PLoS Comput Biol*, vol. 7, no. 5, p. e1002045, 2011.

[11] N. Cuperlier, M. Quoy, and P. Gaussier, "Neurobiologically inspired mobile robot navigation and planning," *Frontiers in Neurorobotics*, vol. 1, no. 0, 2007.

[12] A. Samsonovich and B. L. McNaughton, "Path integration and cognitive mapping in a continuous attractor neural network model," *The Journal of Neuroscience*, vol. 17, no. 15, pp. 5900–5920, 1997.

[13] A. V. Samsonovich and G. A. Ascoli, "A simple neural network model of the hippocampus suggesting its pathfinding role in episodic memory retrieval," *Learning & Memory*, vol. 12, no. 2, pp. 193–208, 2005.

[14] M. Milford, G. Wyeth, and D. Prasser, "RatSLAM: a hippocampal model for simultaneous localization and mapping," in *2004 IEEE International Conference on Robotics and Automation, ICRA '04.*, vol. 1, 2004, pp. 403–408.

[15] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems 7*. MIT Press, 1995, pp. 625–632.

[16] M. Toussaint, "A sensorimotor map: Modulating lateral interactions for anticipation and planning," *Neural Computation*, vol. 18, pp. 1132–1155, 2006.

[17] D. Weiller, L. Läer, A. Engel, and P. König, "Unsupervised learning of reflexive and action-based affordances to model adaptive navigational behavior," *Frontiers in Neurorobotics*, vol. 4, no. 2, 2010.

[18] C. Weber and J. Triesch, "From exploration to planning," in *Artificial Neural Networks - ICANN 2008*, ser. Lecture Notes in Computer Science, V. Kurkov, R. Neruda, and J. Koutnk, Eds. Springer Berlin / Heidelberg, 2008, vol. 5163, pp. 740–749.

[19] M. Witkowski, "An action-selection calculus," *Adaptive Behavior*, vol. 15, no. 1, pp. 73–97, 2007.

[20] J. S. Johnson, J. P. Spencer, S. J. Luck, and G. Schner, "A dynamic neural field model of visual working memory and change detection," *Psychological Science*, vol. 20, no. 5, pp. 568–577, 2009.

[21] W. Erlhagen and E. Bicho, "The dynamic neural field approach to cognitive robotics," *Journal of Neural Engineering*, vol. 3, no. 3, p. R36, 2006.

[22] N. Cuperlier, M. Quoy, P. Laroque, and P. Gaussier, "Transition cells and neural fields for navigation and planning," in *Mechanisms, Symbols, and Models Underlying Cognition*, ser. Lecture Notes in Computer Science, J. Mira and J. Álvarez, Eds. Springer Berlin / Heidelberg, 2005, vol. 3561, pp. 147–152.

[23] E. Torta, R. H. Cuijpers, and J. F. Juola, "A model of the user's proximity for Bayesian inference," in *Proceedings of the 6th international conference on Human-robot interaction*, ser. HRI '11. New York, NY, USA: ACM, 2011, pp. 273–274.

[24] O. Herbort, M. Butz, and G. Pedersen, "The SURE_REACH model for motor learning and control of a redundant arm: From modeling human behavior to applications in robotics," in *From Motor Learning to Interaction Learning in Robots*, ser. Studies in Computational Intelligence, O. Sigaud and J. Peters, Eds. Springer Berlin / Heidelberg, 2010, vol. 264, pp. 85–106.

[25] C. Weber and S. Wermter, "A self-organizing map of sigma-pi units," *Neurocomputing*, vol. 70, no. 13-15, pp. 2552–2560, 2007.

[26] D. Fox, S. Thrun, W. Burgard, and F. Dellaert, "Particle filters for mobile robot localization," in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, Eds. Springer-Verlag, 2001, pp. 499–516.

[27] M. Abramowitz and I. A. Stegun, Eds., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, 1965, ch. 9.6 Modified Bessel Functions $\boldsymbol{I}$ and $\boldsymbol{K}$, pp. 374–377.

[28] S. Amari, "Dynamics of pattern formation in lateral-inhibition type neural fields," *Biological Cybernetics*, vol. 27, pp. 77–87, 1977.