Proceedings of the 2008 International Conference on Cognitive Systems
University of Karlsruhe, Karlsruhe, Germany, April 2-4, 2008

Bridging the Gap: Learning Sensorimotor-Linked Population Codes for Planning and Motor Control

Martin V. Butz, Kevin Reif, & Oliver Herbort

Abstract— **Humans and animals are able to flexibly learn internal, cognitive maps of their environments and are able to use these maps to approach goals efficiently, reliably, and flexibly. Recent neuroscientific evidence suggests that such maps are formed in the hippocampus by means of interconnected place, view, and head direction cell encodings. This paper presents a neural learning architecture that develops an interconnected population code of place cells during random exploration. Connections develop dependent on the experienced sensorimotor contingencies. The learned spatial representation enables the agent to flexibly plan shortest paths to any goal location within the explored environment by means of dynamic programming. It approaches activated goal locations by means of closed loop control. While the algorithm currently relies on the Markov property, it is able to connect the network over radical sensory changes as long as they are close in sensorimotor distance. Moreover, the agent is able to flexibly adjust its behavior dependent on current constraints without further learning. This paper introduces the algorithm and evaluates its robustness and consequent behavioral flexibility.**

I. INTRODUCTION

The ideomotor principle of cognitive psychology [1], [2], [3] suggests that we learn spatial representations and inverse control structures solely based on the observation of sensory-motor correlations during exploratory movements. Once suitable representations emerge, goal-oriented behavior becomes increasingly efficient. Given a current goal, those actions are executed that were previously associated with the goal representation. Tolman showed that animals can latently learn sophisticated cognitive maps, which represent the outline of an explored environment, such as a maze [4]. Cognitive maps allow the simulation of movements and thus the offline or online planning of goal-oriented behavior. Dependent on the representation of the cognitive map, its accuracy, and connectivity, it may be more or less suitable to reach goals in the environment.

While various immediate motor control approaches exist that realize the ideomotor principle (cf. [5]), few investigate the efficient learning of a neural cognitive map based on these principles. This paper introduces an algorithm, termed "time-growing neural gas" (TGNG), which is an unsupervised planning and control algorithm. TGNG latently learns a cognitive map and uses this map for the execution of efficient goal-directed behavior.

This work was supported by the EmmyNoether Program of the German Research Foundation, BU 1335/1

Department of Psychology III (Cognitive Psychology), University of Würzburg, Röntgenring 11, 97070 Würzburg, Germany butz@psychologie.uni-wuerzburg.de
kevin.reif@stud-mail.uni-wuerzburg.de
oliver.herbort@psychologie.uni-wuerzburg.de

The hippocampus is well-known to be crucially involved in episodic memory formation and remembering of past episodes, places, people, etc. In animals (mainly rats), it has been shown that the hippocampus represents (among many other things) locations and orientations within the environment by means of place and head-direction cells, respectively [6]. Place cells have *firing fields* and are mainly active when the animal is located (or moves through) the respective fields. Head-direction cells encode the orientation of the animal in the maze environment, effectively providing compass information[1]. Recent evidence suggests that place cells are also crucially involved in goal-directed planning, since they are necessary for the successful completion of spatial navigation tasks [7], have representational properties that indicate anticipatory encodings [8], show goal-dependent sharp-wave replays of behavior [9], and also exhibit goal-related firing properties [10].

The introduced TGNG grows a place cell representation of the explored environment according to the growing neural gas (GNG) algorithm [11], [12] based on successive sensory input. Different from GNG, neurons, which represent place cells, are connected dependent on the sequential firing of neurons. Due to this approach, connections between place cells do not depend on sensory proximity but rather on sensorimotor proximity, that is, the proximity of sensory inputs dependent on motor activity. Thus, TGNG develops a cognitive map that is represented as a graph in which an edge encodes the possibility to move from one node to the connected neighbor. In comparison with properties of the hippocampus, place cells are nodes in the graph and connections associate head-direction cell activity.

We show that the representation is very suitable to efficiently plan and execute goal-directed behavioral sequences, simply by activating goal states and propagating this activity through the graph by means of dynamic programming. We study the effects of parameters and noise on network size and goal-directed behavior. We also show that the representation is mostly independent of the sensory representation of the environment used, as long as the environment stays Markovian. Moreover, we show that the learned representation allows the flexible adjustment of behavior to novel constraints, such as preferred movements in a certain direction.

The article is structured as follows. We first give an overview over the architecture of TGNG. Next, we evaluate the general properties of the algorithm in various maze tasks.

[1]Although, of course, there is no absolute encoding for e.g. North, but the orientation is relative to the landmark distribution in the space.

Furthermore, we show noise-robustness of the algorithm. Finally, we show that TGNG is able to flexibly adjust behavior due to additional behavioral task constraints without further learning. A final discussion concludes the paper.

II. ARCHITECTURE

Based on the Growing Neural Gas (GNG) architecture [12], the presented architecture simulates the learning of a cognitive map in two dimensional environments. To establish an interconnected network (nodes connected by edges) of place cells, which represent the cognitive map, a simulated robot randomly explores the environment for a certain amount of time (measured in steps of a predefined length). Should the robot reach an unmapped area of the environment, new nodes (place cells) are created and connected to the rest of the network . After every step, the robot's movements in x- and y-direction (its motor vector) is stored in those edges best representing its trajectory. In every iteration an edge is updated by increasing its experience and updating the associated motor vector. Eventually, an evenly distributed network of nodes and edges evolves, of which the nodes represent place cells and the edges store the average motor vector applied when traversing the edge. Previous approaches have successfully shown that such a network has strong relations to biological neural networks and can be used to reach goals effectively [13], [14]. Here, we show that edges can be formed over time without the need for perceptual proximity, thus bridging perceptually distant locations while still allowing efficient planning and goal-directed behavior.

A. Growing Neural Gas

As shown by Fritzke [12], GNG can be used to learn important topological relations in a given set of input vectors by means of a simple Hebb-like learning rule. Enhancing and modifying Fritzke's algorithm in several respects, the presented architecture is now able to learn a cognitive map of a complex environment by random exploration. Fritzke's algorithm is shown in Fig. 1.

While GNG was designed for static clustering applications, our intention was to enhance the approach for the generation of cognitive maps that encode distances over time. Thus, connections should be time-dependent instead of sensor-dependent. This goal lead to the following modifications of GNG, creating the time-growing neural gas algorithm TGNG.

B. Time Growing Neural Gas

Several modifications lead to the TGNG algorithm: instead of sampling input signals at random (Step 2), our model uses a robot that explores the environment. The most significant difference can be found in Step 3: instead of using the Euclidean distance between the input signal and the existing nodes to determine two winning nodes, TGNG uses a combination of the Euclidean distance and the temporal relation between two input signals. The first winner (as Fritzke's) is the node nearest to the current input signal. The second winner, however, is the node that used to be the nearest

1) Start with two nodes at random locations.
2) **Receive an input signal.**
3) **Find the nearest and second-nearest node (Winner 1 and 2) to input signal.**
4) Increment the age of all edges emanating from Winner 1.
5) **Add the squared distance between input signal and Winner 1 to a Winner 1's error variable.**
6) Move Winner 1 and its direct neighbors towards the input signal by certain hard coded fractions.
7) **If Winner 1 and 2 are already connected by an edge, set its age to 0, otherwise create an edge.**
8) Remove all edges older than a predefined maximum age. If this results in nodes with no emanating edges, remove them as well.
9) After a certain number of input signals (integer multiple of a hard coded variable) insert a new node as follows:
   a) Determine the node q with the maximum accumulated error.
   b) Insert a new node r halfway between q and its neighbor f with the largest error variable.
   c) Insert edges between r and q and r and f, and remove the edge between q and f.
   d) Decrease the error variables of q and f by a constant fraction.
   e) 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.
11) Repeat steps 2 to 11 until a predefined stopping criterion.

Fig. 1. Fritzke's growing neural gas algorithm [12].

node in the previous iteration. Thus, two different input signals are used to identify the two winners for the next steps in the algorithm. This TGNG learns relations between perceptually distant locations in the maze that, nonetheless, can be traversed with a small number of steps.

Furthermore, to improve the quality of the generated network, we use a global error variable ϵ (Step 5) that is used to determine the location and instant when a new node should be inserted. This closely relates to concepts introduced by Toussaint [14]. The global error variable ϵ is updated every iteration reflecting the low-pass filtered current local error:

$$\epsilon \leftarrow \epsilon + \delta(d - \epsilon), \tag{1}$$

where d is the Euclidean distance to the nearest node in perceptual space and δ is the leak in the low pass filter accounting for a logarithmically weighted recent error distribution. In case this accumulated global error exceeds a hard coded threshold θ_ϵ, a new node gets inserted at the location of the current input signal and connected to the nearest node. In this way, it is possible to map the given maze in a far shorter time while preserving the network's accuracy.

A final modification of Fritzke's algorithm lies in optionally not only updating the edge between the two winner nodes (Step 7). Instead it is also possible to update all the edges that connect all those nodes that were in the vicinity of the robot before with all those that are in its vicinity after the robot moved.

Similar to storing direction estimates [14], our architecture stores motor vectors in the edges of the network that later can be used to plan to any given goal location within the representation. After every step, the robot's current motor vector is stored in the edge between the two winning nodes (over time). Optionally, all edges representing the robot's trajectory are updated. To determine these edges, two lists are created. All nodes in the lists are direct neighbors of the current winner node or the current winner node itself. The

first list contains all those nodes that were nearer to the input signal at time t1 than at time t2. The second one consists of all those nodes that were nearer to the input signal at time t2 than at time t1. Thus, two sets of nodes are created of which all connecting edges between nodes of the two sets (and the possibly newly created edge between winners 1 and 2) are updated in their experiences and motor vectors using the moyenne adaptive modifiée [15] technique:

$$mv_{ij} \leftarrow \begin{cases} \frac{exp_{ij} \cdot mv_{ij} + mv \cdot frac}{exp_{ij} + frac} & \text{if } exp_{ij} < 1/\beta_m \\ \frac{mv_{ij} + mv \cdot \beta_m \cdot frac}{exp_{ij} + \beta_m frac} & \text{otherwise} \end{cases}, \quad (2)$$

where exp_{ij} denotes the number of update fractions the edge connecting node i to node j has been updated, $frac$ denotes the update fraction for this edge, mv is the currently executed motor vector, and β_m is the learning rate for the motor vector update. This method ensures that an average motor vector is generated as fast as possible. Later on, a moving average is preferred to ensure continuous adaptivity. If only the connection between the winning nodes is used, $frac$ equals one. However, if also neighboring node connections are updated, then a full update is distributed over the edges that are updated based on the extend the previous node's activity decreased and the current node's activity increased.

As stated above, the intention to build this sensorimotor-based cognitive map is to use it for efficient, goal-directed behavior. To reach a given goal location, goal-originating activity can be propagated backwards through the network until the activity reaches the node that coincides the the robot's starting location. The activity propagates through edges from nodes j to nodes i whose experience exp_{ij} (the direction pointing towards node j) is greater than 0. The following equation specifies the activity propagation:

$$a_i \leftarrow a_i + \beta_a \ (\max\{a_i(g), \gamma \max_{j \text{ with } exp_{ij}>0} a_j\} - a_i), \quad (3)$$

where a_i denotes the activation of a node i, $a_i(g)$ is its associated goal significance, and β_a denotes the activity-related propagation rate. Essentially, this corresponds to a typical Q-learning and model-based reinforcement learning approach [16].

Once the activation values saturate, propagation is stopped. The resulting node activations represent their relation to the goal node. The higher the activation in a node, the easier it is to reach the goal from this node. Movement penalties can be accounted for within the proposed equation either adjusting the propagation rate γ based on the motor vector associated with edge ij or by adding a fixed "effort" penalty dependent on the executed motor vector.

Given a current activity distribution, the robot can walk to a goal by activating the motor vectors that are associated with edges that point towards higher activity nodes. Two ways are contrasted to determine the appropriate motor vectors: either, only the edge to the highest activity node that emanates from the currently active node (that is, the edge connecting to the node with the highest activity) is selected and its motor vector is used for the robot's next step; or, a weighted average over all motor vectors of those edges connecting to nodes with a higher activation than the currently active one is used. A node stays active as long as it is the nearest node to the robot's current position.

III. EMPIRICAL EVALUATION

To evaluate TGNG, we tested the system on a simple simulated robot platform in various maze environments. Depending on the mode of the experiment, the simulated robot perceives either its absolute location in the maze (x,y real-valued coordinates) or eight distance measures surrounding the robot equally spaced. Based on this sensation, the nodes are created and controlled.

The maze environment consists of a number of square blocks that are either defined passable or impassable for the exploring robot. Exceptions are "teleporter" blocks. When entered, these blocks teleport the robot immediately to the respective counterpart block. Along the lines of Toussaint's work [14], the robot changes its current direction with a probability of $p = .2$ and uses a predefined step length.

For the evaluations, we used three mazes[2] (Fig. 2). All had the same number and size of blocks. The first environment was an empty room with no obstacles. The second one was a complex maze without teleporters. The last maze was identical to the second but with three pairs of teleporters for maximum complexity. Every maze was randomly explored and each experiment was repeated ten times with different random number generator seeds.

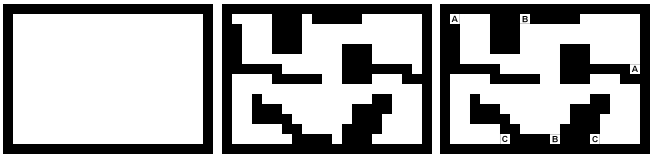

Fig. 2. Performance was evaluated in three mazes: (1) An empty room, (2) a complex maze with various pathways, and (3) the same complex maze with additional teleporter locations, which transferred the robot agent to the open side of the counterpart teleporter.

To test the quality of the developing sensorimotor maps, we interrupted learning at pre-defined intervals and generated a set of nine locations: the most left, center, and most right positions in the top, center, and bottom (unblocked) rows of the maze. The nodes closest to the locations served as start or goal nodes. Each combination of start and goal locations is tested, yielding $9 \times 8 = 72$ start-goal combinations. After propagating goal-originating activations through the network, the robot has to find its way to the goal using the edge-associated motor vectors as described above. The performance analysis below considers the percentage of successful goal reaches, path length of planned and executed paths, as well as the density of the generated networks. Using these measures, we now evaluate parameter dependencies as well as the capability to account for behavioral constraints.

[2]The mazes were chosen arbitrarily. Other maze configurations should generally only alter the outcome if the Markov property is violated.

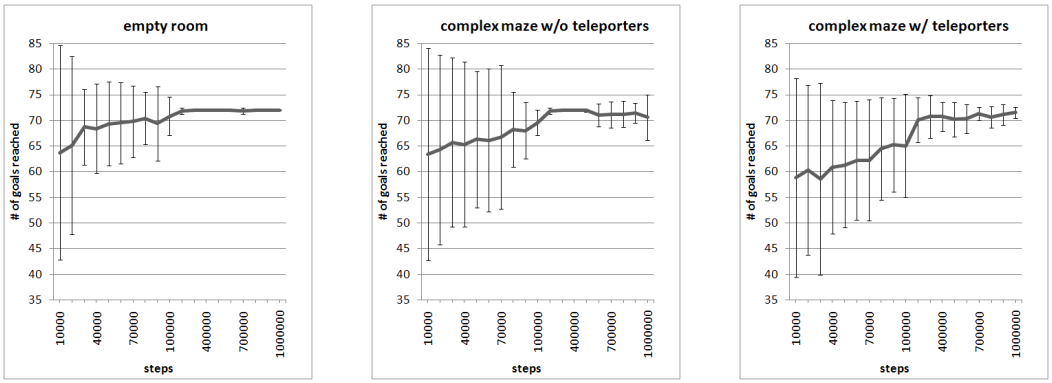

Fig. 3. Learning in the three tested maze environments confirms fast and accurate learning in the empty maze. Also in the more complex mazes near-accurate performance is reached reliably.

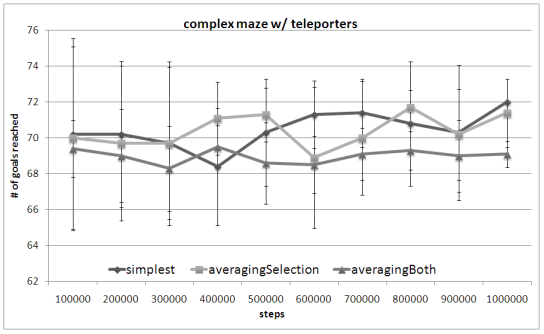

Fig. 4. Successful behavior depends on the method with which motor vectors are associated with edges and how the associated motor vectors are used to control goal-oriented behavior.

A. Learning Curves

Before delving into the details of the algorithm and its constraints, we evaluate the overall learning performance achievable with the general setup. Generally, learning progresses rapidly. Fig. 3 shows that in an empty maze after $10,000$ learning iterations (i.e. steps of the robot) already more than 60 of the 72 goals are reached[3]—albeit with a high standard deviation of $\approx 20\%$. After $100,000$ steps almost all of the 72 goal nodes are reached and the standard deviation drops to $\approx 3\%$. The more complex mazes pose a slightly harder learning challenge. Nonetheless, nearly all goals are reached reliably and also the maze with teleporters can be learned efficiently.

B. Parameter Dependencies

To determine which mode of storing and selecting the motor vectors is best for the robot's performance, we tested several combinations of learning and goal-directed behavioral settings: For the routine that associates motor vectors with edges, we either associated only the winning edge or all trajectory-representing edges, as specified above (2). For selecting the motor vector for goal-directed behavior, we either executed only the motor vector associated with the best edge or generated a weighted average of the motor vectors linking to higher activated nodes).

Best performance was achieved when the activity-weighted average motor vector was used during goal-directed behavior and only the winning edge was considered during training. Fig. 4 compares the three settings: either only the best connection was updated and selected during goal-directed behavior, or the motor vector generation used the averaging, or also learning used the averaging for the motor vector updates. Results are only shown in the maze with teleporters since the differences in the other two mazes were even less pronounced. In the teleporter maze, slightly better performance is achieved when the motor vectors are averaged during goal-directed behavior but not during learning. This indicates that motor vector averaging during learning can yield disruptive performance when attempting to traverse a teleporter.

Network Density: Our architecture is able to adapt the density of the created network by means of a hard-coded threshold. A higher threshold implies a smaller number of nodes, thus leading to a runtime advantage. On the other hand, a low threshold leads to the creation of more nodes. Fig. 5 shows how performance depends on network density. Thresholds larger than $.4$ showed to decrease the success rates. However, even with very small thresholds TGNG was

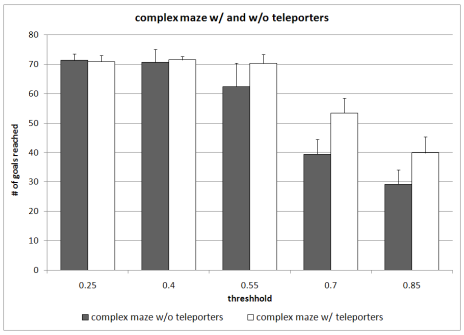

Fig. 5. Dependent on the error threshold θ_ϵ, which specifies the tolerance to sensory change, the resulting network yields different success rates during goal-directed behavior (performance after 1M learning iterations).

[3]Since we use arbitrary movements to explore an environment it takes the robot between $8,000$ and $15,000$ steps to visit every part of the maze at least once.

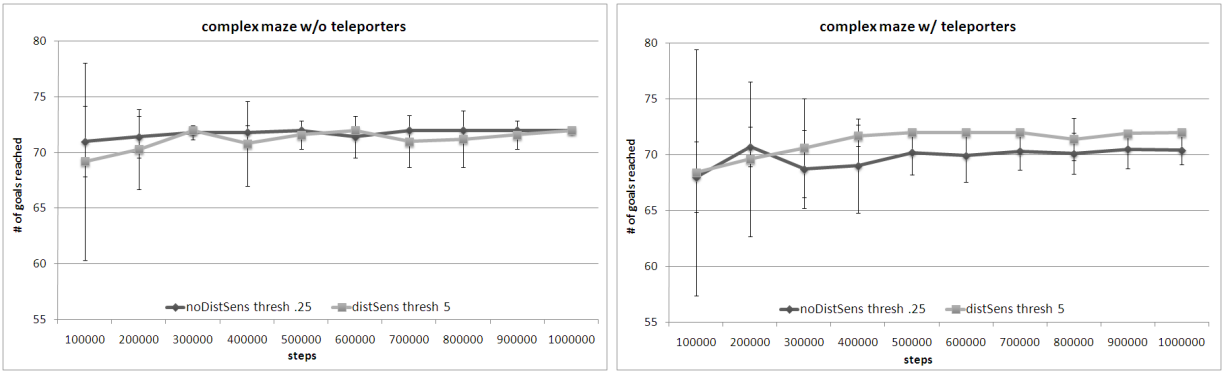

Fig. 6. Comparing performance with x,y sensors and distance sensors shows that the algorithm also works robustly in either case.

TABLE I

NUMBER OF NODES AND EDGES AFTER 1M LEARNING STEPS.

(x,y) sensors	θ_ϵ	.25	.4	.55	.7	.85
	nodes	1675	848	545	409	321
	edges	4719	2257	1397	1024	783
dist. sensors	θ_ϵ	3	5	6	9	
	nodes	5431	2511	1968	1137	
	edges	18293	7844	6037	3407	

able to generate a cognitive map appropriate for goal-directed behavior. This suggests that the grid needs to be fine enough. An overly fine network, however, only affects performance speed but not accuracy. Table I shows the resulting network densities after one million steps of learning. The number of rows and edges in the developing networks are highly dependent on the threshold θ_ϵ. However, the ratio between nodes and edges stays nearly constant at around three.

C. Distance Sensors

So far, all evaluations were based on an absolute location sensation, that is, the sensation of the coordinates of the robot's current location. Clearly, such a sensation information is rather unrealistic for a cognitive system. Thus, we conducted the same evaluation with a sensory space of eight distance sensors, which encode the distance to the nearest wall in eight directions uniformly surrounding the robot. The only adjustment necessary to learn in this environment was to increase the threshold θ_ϵ. If not stated differently, in the runs with distance sensations we set the threshold to 5.

Fig. 6 shows that the performance with distance sensors hardly differs from the one encoding the absolute robot location. In fact, with distance sensors, performance is slightly better than without distance sensors. This may be due to fact that the developing cognitive map in the encoding with distance sensors contains more nodes (1635 vs. 2616 in the maze without teleporters and 1586 vs. 2511 in the maze with teleporters). However, Fig. 7 shows that this performance is robust even with a threshold of 9 in the teleporter maze, in which the number of nodes and edges is smaller than in the case with x, y sensors and a threshold of .25 (cf. Table I. Thus, the better performance appears to be caused by the finer density of the network around corners and teleporters.

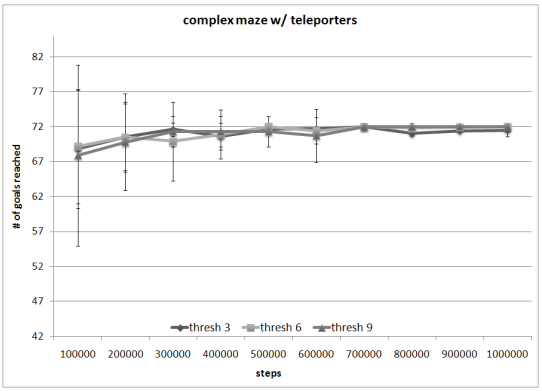

Fig. 7. In the maze with teleporters, a wide variety of error thresholds θ_ϵ yield optimal behavior in the setting with distance sensors.

This second fact is due to the sensory property of distance sensors: at corners that point inside the maze, the distance sensors encounter large changes, consequently causing the generation of more nodes than at other locations in the maze.

D. Paths and Movement Constraints

To illustrate how certain constraints can immediately affect behavior, we introduced a movement constraint. To do so, we distinguished movements in four directions (positive and negative x and y direction) and added a penalty to one of the directions. The penalty was used as a multiplicative decrease in the activity propagation (3) dependent on the percentage of movement in the penalized direction. For example, given that the motor vector associated with an edge encodes that 70% of the movement is taken in positive x direction when traversing this edge and that there is a penalty of $.25$ to move in positive x direction, then activity propagated through that edge is decreased by $\gamma \times (1 - .25) \times .7 + .3 = .825$ instead of γ only. Note that such a constraint can be added without any further learning effort. Due to the model-based behavioral approach, the system can thus adjust immediately to its current movement preferences.

We compared walked paths without and with constraints. In the mazes without teleporters, the constraints did not have any behavioral effect because the mazes do not allow alternative routes. However, in the maze with teleporters,

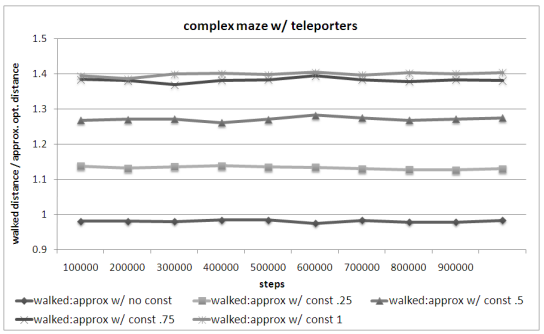

Fig. 8. When increasing the strength of the movement constraint, the robot takes increasingly long detours to minimize movements to the right.

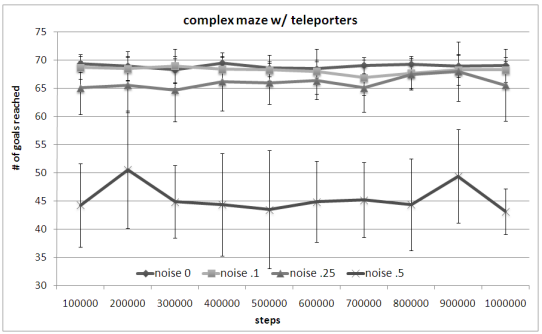

Fig. 9. Moderate noise levels below the node creation threshold $\theta_\epsilon = .4$ hardly affect performance. However, once noise approaches the chosen threshold, performance degrades significantly.

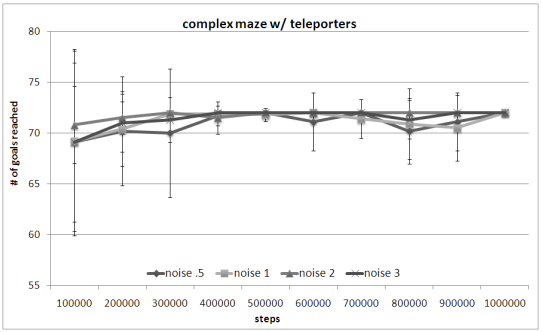

Fig. 10. In the case with distance sensors, noise did not affect performance.

the paths taken became longer since the system prefers a longer path in negative x direction instead of the direct path in positive x direction. While in the unconstrained case approximately 35% of the walked paths was in positive x direction, with a constraint of .5, the movement decreased to 14%.

Fig. 8 compares the relative path lengths taken with different constraint strengths. Each path was compared to an approximate optimal distance deduced for the maze. It can be clearly seen that the approximate path length without constraints is slightly better than the approximate optimal path length[4]. However, when adding constraints, paths become longer because the system prefers more complex routes (detours) through teleporters (or not through teleporters) in order to avoid movements to the right.

E. Noisy Sensations

To further study the robustness of the algorithm, we also added independent Gaussian noise with a certain standard-deviation to each sensory input. Fig. 9 shows that noise deviations below the threshold $\theta_\epsilon = .4$ hardly affect performance. Larger thresholds, however, strongly increase the network size (953 nodes and 3846 edges with Gaussian noise of standard deviation .5) and lead to learning disruptions due to aliased perceptual inputs and unsuitable place cell locations and connections. A similar observation holds also for the case when distance sensory are employed (Fig. 10). Hereby, noise levels of deviation 3 still hardly affected performance. However, the number of nodes in the network increased significantly from 2617 nodes (8256 edges) with a noise of 1 to 3604 (13193) with a noise of 3. For values above 3, the number of nodes and edges increased so much so that a reasonable runtime below one hour per experiment was not achievable any longer.

IV. SUMMARY AND CONCLUSIONS

This paper has shown that a relatively simple algorithm, the time-growing neural gas (TGNG), can effectively and robustly learn cognitive maps of environments. The algorithm proved to be noise-robust and relatively parameter independent, while generating near-optimal paths to goals. Moreover, the algorithm can flexibly account for additional movement constraints, such as a movement direction preferences. Finally, behavioral success does not depend on any prior spatial knowledge of the environment nor any distance or directional knowledge. The only prior information used was a compass and the capability to move relative to the compass orientation. Thus, the information available to TGNG is comparable to the one available to the hippocampus, which encodes place cells and head direction cells. As presumably in the hippocampus, connectivity between place cells and the attachment of place cells to spatial areas happens on the fly during random exploration.

In comparison to Toussaint's algorithm [14], which also learned cognitive maps in a maze environment, we showed that TGNG is able to build cognitive maps over time bridging gaps and strong changes in sensory input. This enables the formation of connections through teleporter gateways, which instantly moved the robot to a different location in the environment. Moreover, it enables the formation of maps with distance sensors, regardless of strong sudden changes of the distance sensor input.

In relation to previous robot platform application, Arleo and Gerstner [13] have build a similar hippocampal place cell model, which processes actual pre-processed visual input in multiple layers for the generation of place fields. However, connections between place cells were not encoded explicitly

[4]The approximate optimal path length was determined by considering movements from the center of one block to the center of any of the eight neighboring blocks. As well as movements through teleporters from neighboring blocks. Thus, this is an approximation that nonetheless serves well for the comparison of relative qualities of generated paths.

but were derived by the provision of path-integration cells, which provided allocentric spatial representations. To activate behavior, the authors learned an explicit state-action-value map, which prevents the fast adaptation to changed goal locations. TGNG does not require any direct allocentric information and state-action mappings are generated online by the employed activation propagation mechanisms, which is possible since a complete cognitive map is learned. Another related model is that of Cuperlier et al. [17]. While this model is able to bridge strong sensory changes by the formation of transition cells without any explicit allocentric information, it remains unclear how robust the employed algorithm works and how robust it is to noise. TGNG shows that the usage of transition cells is not necessary still yielding strong behavioral robustness also in the case of noisy sensory information.

While TGNG seems to be a promising algorithm for the autonomous learning of cognitive maps, several current challenges need to be addressed. First, the algorithm is mainly dependent on one parameter: the threshold θ_ϵ, which determines when a new node should be created. An interesting idea is to make the threshold dependent on the average change in sensory input during random movements. Second, the algorithm currently assumes that all sensory inputs are relevant for the place cell formation. Thus, behavioral relevance measures, grouping measures, and generalization capabilities need to be added to appropriately generalize the sensory input. Clearly, in the brain this occurs already mainly before sensory information reaches the hippocampus. Third, the algorithm relies on the Markov property. That is, sensory input needs to provide enough information to localize the agent in the environment and to be able to predict average action effects deterministically. Since this is not the case in general, internal state representations need to be added to distinguish different subspaces in the environment. Hierarchical reinforcement learning provides several approaches to develop such representations during random exploration [18], [19], [20]. Finally, the hippocampus re-uses place cells and re-connects place cells highly plastic in different environments [7]. Thus, cognitive maps should be flexibly reactivated, dependent on the sub-environment currently in. Neural gating techniques seem to be necessary to be able to re-activate different memory structures context-dependently.

Despite the various challenges ahead, this paper has shown that a simple self-organizing map algorithm can be employed to structure cognitive maps over time and that a simple association mechanism is sufficient to associate appropriate motor vectors. Thus, TGNG builds cognitive maps without any prior distance or other spatial information. Distances are derived by the encountered sensorimotor contiguity alone. Future research will apply TGNG on a real robot platform and tackle the challenges discussed above combining the system with other interactive sensory and motor processing mechanisms.

V. ACKNOWLEDGMENTS

The authors are grateful to Prof. Hoffmann and all the colleagues at the Department of Cognitive Psychology. Moreover, the authors are grateful to Prof. Puppe for his support.

REFERENCES

[1] J. F. Herbart, *Psychologie als Wissenschaft neu gegründet auf Erfahrung, Metaphysik und Mathematik. Zweiter, analytischer Teil*. Königsberg, Germany: August Wilhem Unzer, 1825.

[2] J. Hoffmann, C. Stöcker, and W. Kunde, "Anticipatory control of actions," *International Journal of Sport and Exercise Psychology*, vol. 2, pp. 346–361, 2004.

[3] W. James, *The principles of psychology*. New York: Holt, 1890.

[4] E. C. Tolman, "There is more than one kind of learning," *Psychological Review*, vol. 5b, pp. 144–155, 1949.

[5] G. Pezzulo, G. Baldassarre, M. V. Butz, C. Castelfranchi, and J. Hoffmann, "From actions to goals and vice-versa: Theoretical analysis and models of the ideomotor principle and tote," in *Anticipatory Behavior in Adaptive Learning Systems: From Brains to Individual and Social Behavior*, M. V. Butz, O. Sigaud, G. Pezzulo, and G. Baldassarre, Eds. Springer-Verlag, 2007.

[6] S. I. Wiener, A. Berthoz, and M. B. Zugaro, "Multisensory processing in the elaboration of place and head direction responses by limbic system neurons," *Cognitive Brain Research*, vol. 14, p. 7590, 2002.

[7] B. Poucet, P. P. Lenck-Santini, V. Hok, E. Save, J. P. Banquet, P. Gaussier, and R. U. Muller, "Spatial navigation and hippocampal place cell firing: The problem of goal encoding," *Reviews in Neurosciences*, vol. 15, pp. 89–107, 2004.

[8] J. G. Fleischer, "Neural correlates of anticipation in cerebellum, basal ganglia, and hippocampus," in *Anticipatory Behavior in Adaptive Learning Systems: From Brains to Individual and Social Behavior*, M. V. Butz, O. Sigaud, G. Pezzulo, and G. Baldassarre, Eds. Springer-Verlag, 2007.

[9] D. J. Foster and M. A. Wilson, "Reverse replay of behavioural sequences in hippocampal place cells during the awake state," *Nature*, vol. 440, pp. 680–683, 2006.

[10] V. Hok, P.-P. Lenck-Santini, S. R. andEtienne Save, R. U. Muller, and B. Poucet, "Goal-related activity in hippocampal place cells," *The Journal of Neuroscience*, vol. 27, p. 472 482, 2007.

[11] T. M. Martinetz, S. G. Berkovitsch, and K. J. Schulten, ""Neural-gas" network for vector quantization and its application to time-series prediction," *IEEE Transactions on Neural Networks*, vol. 4, pp. 558–569, 1993.

[12] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. Cambridge MA: MIT Press, 1995, pp. 625–632.

[13] A. Arleo and W. Gerstner, "Spatial cognition and neuro-mimetic navigation: A model of hippocampal place cell activity," *Biological Cybernetics*, vol. 83, pp. 287–299, 2000.

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

[15] G. Venturini, "Adaptation in dynamic environments through a minimal probability of exploration," *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pp. 371–381, 1994.

[16] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press, 1998.

[17] N. Cuperlier, M. Quoy, C. Giovannangeli, P. Gaussier, and P. Laroque, "Transition cells for navigation and planning in an unknown environment," *From Animals to Animats*, vol. 9, pp. 286–297, 2006.

[18] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dynamic Systems*, vol. 13, pp. 341–379, 2003.

[19] M. V. Butz, S. Swarup, and D. E. Goldberg, "Effective online detection of task-independent landmarks," Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, IlliGAL report 2004002, 2004.

[20] Ö. Simsek and A. G. Barto, "Using relative novelty to identify useful temporal abstractions in reinforcement learning," *Proceedings of the Twenty-First International Conference on Machine Learning (ICML-2004)*, pp. 751–758, 2004.