A biologically inspired meta-control navigation system for the Psikharpax rat robot

K Caluwaerts[1,2], M Staffa[1,3], S N'Guyen[1], C Grand[1], L Dollé[1] A Favre-Felix[1], B Girard[1] and M Khamassi[1]

[1] Institut des Systèmes Intelligents et de Robotique (ISIR) UMR7222, Université Pierre et Marie Curie, CNRS, 4 place Jussieu, 75005 Paris, France

[2] Reservoir Lab, Electronics and Information Systems (ELIS) department, Ghent University, Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium

[3] Dipartimento di Informatica e Sistemistica, Università degli Studi di Napoli Federico II, Via Claudio 21, 80125, Naples, Italy

E-mail: `mehdi.khamassi@isir.upmc.fr`

Abstract. 200 words
A biologically-inspired navigation system for the mobile rat-like robot named Psikharpax is presented, allowing for self-localization and autonomous navigation in an initially unknown environment. The ability of parts of the model (e.g. the strategy selection mechanism) to reproduce rat behavioral data in various maze tasks has been validated before in simulation. But the capacity of the model to work on a real robot platform had not been tested.

This article presents our work on the implementation on the Psikharpax robot of two independent navigation strategies (a place-based *planning* strategy and a cue-guided *taxon* strategy) and a strategy selection meta-controller. We show how our robot can memorize which was the optimal strategy in each situation, by means of a reinforcement learning algorithm. Moreover, a context detector enables the controller to quickly adapt to changes in the environment - recognized as new contexts -, and to restore previously acquired strategy preferences when a previously experienced context is recognized. This produces adaptivity closer to rat behavioral performances and constitutes a computational proposition of the role of the rat prefrontal cortex in strategy shifting. Moreover, such brain-inspired meta-controller may provide an advancement for learning architectures in robotics.

1. Introduction

1.1. The Psikharpax robot

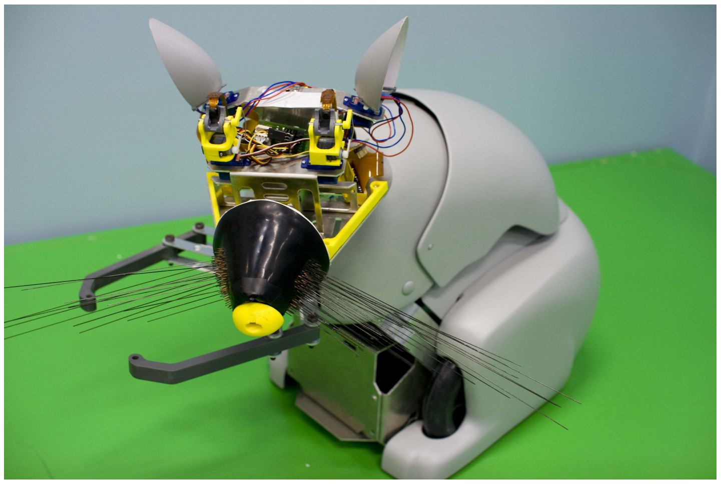


Figure 1. The v2 Psikharpax robot.

The Psikharpax robot [1] is designed as an *artificial rat*, a robotic platform built to integrate computational models of the rat's decision, learning, motivational and navigation circuits. It is used for two purposes: as a tool to contribute to neuroscience by studying how an embodied agent can adapt in the real world with noisy perceptions and continuous time and state spaces, and by testing current neuroscience theories in such context; as a means to test the potential application to robotics by assessing the transferability of neurocomputational models of learning and decision-making to robots operating in dynamic, unknown environments.

This article is the first to report on spatial navigation with the new version of Psikharpax (v2; Fig. 1). The robot has been equipped with a rich sensory set of devices for multimodal perception (binaural auditory equipment, artificial whiskers, binocular vision) and sensory integration. This previously allowed to perform tactile texture discrimination and obstacle avoidance with the whiskers [2], hearing and noise localization [3], vision and adaptive saccadic eye movements [4, 5]. Here we present the upgrade of the robot's cognitive architecture to enable the robot to coordinate and learn multiple strategies for spatial navigation, and perform fast adaptation to environmental changes.

1.2. Multiple navigation strategies in rodents

Mammals are able to use multiple strategies when faced with a navigation problem [6, 7, 8, 9, for reviews], like reaching a hidden platform in a pool, the so-called Morris water maze [10]. Among the numerous possible strategies, experimental neuroscience studies of strategy interactions favored two main families:

- Response strategies, resulting from the learning of direct sensori-motor associations (like swimming towards a cue indicating the platform location, which is called a *taxon strategy*).
- Place strategies, where the animal builds an internal representation (or cognitive map) of the various locations of the environment, using the configuration of multiple allocentric cues. It then uses this information to choose the direction of the next movement by either learning place-action associations (*place recognition triggered response strategy* or *PRTR*), or, more adaptively, by planning a path in a graph connecting the places with the actions allowing the transitions from one place to another (*topological planning strategy*).

It has been shown that the multiple navigation strategies of rodents are operated by parallel independent memory systems [11, 12], which can result in *cooperative* or *competitive* behaviors, depending on the experimental protocol. The basal ganglia (BG) and the hippocampal formation (Hpc) appear to have a central role in this circuitry. The BG can be subdivided in parallel sub-circuits [13], usually identified by the part of the striatum – the main BG input nucleus – they incorporate. The BG operate action selection [14] and use reinforcement learning signals mediated by dopamine [15] to adapt these selections to environmental conditions.

Response strategies are considered to rely on the projections from the sensory and motor cortices to the basal ganglia circuits issued from the dorso-lateral striatum (DLS) to select directions of movement, using reinforcement learning capabilities of the BG to learn which cue is to be followed at a given time [16, 17]. Consistently, lesions of the striatum – or more specifically of the DLS – impair or reduce the expression of response strategies while promoting place strategies [18, 19]. In contrast, lesions of the hippocampal system impair place strategies while sparing response strategies [20, 11, 18]. This suggests that response strategies are independant from the Hpc. On the other hand, place strategies would rely on the Hpc, with its ability to encode places in the so-called *place cells* [21], to provide inputs to work with. The neural circuits exploiting them to either learn place-action associations or to plan trajectories, would be located in the prefrontal cortex, in the ventral striatum (VS) and in the dorso-medial BG circuit (DMS). Indeed, lesions of the DMS reduce the expression of place strategies while promoting response strategies [18, 22]. Lesions of the VS impair animals ability to associate different places with different amounts of reward [23].

1.3. State of the art of neuro-inspired robotic navigation

Several previous projects have tested biomimetic models of rodent navigation on robots, based on these experimental data. Such projects participate to the global approach consisting in transferring neurocomputational models to robotics with a two-fold objective: On the one hand, taking inspiration from the computational principles underlying mammals' behavioural flexibility to contribute to the improvement of current robots' autonomy and adaptivity. On the other hand, using the robot as a platform to test the robustness of current biological hypotheses about spatial cognition, beyond perfectly controlled simulations, and try to learn more about the computational mechanisms at stake by analyzing which solutions enabled the model to work on a physical robot [24, 25, 26].

Arleo and Gerstner developed a computational model of place cells - neurons located in the hippocampus whose activity encode an estimation of the animal's current position - and head-direction cells - neurons selective for the estimated orientation of the animal's head [27]. With this model, they enabled a Khepera robot to navigate in a small arena, using a navigation strategy where a reinforcement learning algorithm learns associations between places and directions of movement (a PRTR strategy). Fleischer, Krichmar and colleagues showed how prospective and retrospective coding at the level of place cells' activity can enable a robot to efficiently solve a spatial memory task [28, 29], here also navigation was performed by a PRTR strategy. Barrera and Weitzenfeld proposed a hybrid PRTR strategy using a graph, where the choice of the next action took into account the next three actions in a prospective manner [30]. Their robot could solve discretized implementations of various rodent laboratory mazes (T and radial mazes). Giovanangeli and Gaussier developped a model of another navigation strategy consisting in planning routes towards the goal in a topological graph ('cognitive map') of the environment. Their model produced efficient navigation in both indoor and outdoor environments [31]. More recently, the RatSLAM algorithm has been implemented as a neural network inspired by the rat's hippocampus in order to perform efficient, continuous and long duration Simultaneous Localization and Mapping (SLAM) on a robotic platform put in a large non-stationary environment [32]. Planning is also used here to perform navigation.

Our contribution relies in transferring to robotics another aspect of rodent navigation abilities: the combination of various navigation strategies, in order to benefit from their respective strengths (accuracy, learning rate, adaptation to changes, etc.), coordinated by a meta-controller for strategy-shifting which has been previously shown to better reproduce rodents' behavioral performance than single navigation strategies [33]. Thus we extracted the principles of each previously studied components of rats' currently known cognitive architecture for navigation: place cells, path integration component, path planner, reinforcement learner. And we focused on the integration of these components in a brain-inspired system used for adaptive strategy shifting.

1.4. The computational model previously used in simulation

In this paper, we first apply to the Psikharpax robotic platform the multiple strategy switching model (see Fig. 2) proposed by [33], which was tested in simulation to replicate rat behavioral experimental results. We then propose an extension of this model allowing a more flexible adaptation when switching from an experimental context to another (i.e. change in goal location).

The [33] model provides a simple mechanism able to replicate experimental results

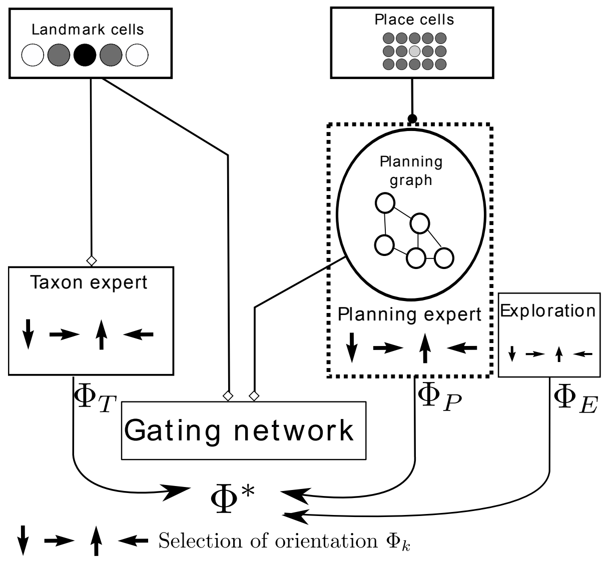


Figure 2. Overview of the (Dollé et al., 2010) model. Different strategies (taxon/topological-map/exploration) are connected to the gating network. Each strategy has a dedicated expert which proposes actions (Φ_T for the taxon, Φ_P for the planing, ...). The gating network decides which of the experts is the winner in the current situation and then the action Φ^* from this strategy is performed.

obtained by [34] and [18] in variations of the Morris water maze protocol: it proposes that a gating network is dedicated to the selection of the strategy to be used, and that it uses reinforcement learning to learn which strategy is the most efficient in each situation, based on all the inputs used by the strategies to take their own decisions (i.e. sensory and place cells activity). All strategies learn simultaneously: those which did not have the control over the last decision use the reward/punishment signals modulated by the angular difference between their movement suggestion and the actual one: the smaller the difference, the more the suggested movement of a non-selected strategy will be rewarded. This is a key element of the model to explain the cooperative effects observed in animals, where the learning process of a slow learning strategy can thus be guided by the selections made by a fast learning one.

In the experiment from [34], external visual cues, allowing the generation of an internal map, are provided, and the hidden platform is indicated by a visual cue standing 20cm away from the platform in a fixed direction (making taxon strategies more difficult to learn and less efficient than when it's directly above the platform). The platform is moved after every session of four trials, so that rats using a map-based strategy perform poorly at the beginning of a new session, while those favoring a taxon strategy are not much affected. Finally, the experiment is carried out with a group of control rats and another one with rats with hippocampus lesions. The model reproduces the differences in performance and in learning dynamics of both groups: lesioned rats learn accross sessions, while control ones also learn within sessions, being less efficient than the lesioned at the beginning and better at the end. The model shows this can be explained by competition between strategies in the beginning of a session –each strategy leads to a different place, as the map-based one leads to the previous location– and cooperation in the end –once the new location is known, the taxon provides the global direction to reach the platform in the beginning of the trajectory, but the map-based strategy is more efficient than the taxon to precisely lead to the platform location, rather than to the cue, at the end of the trajectory.

In [18], nine sessions are carried-out with four groups of rats (control, fornix-, DLS- and DMS-lesioned rats), external cues are provided and the platform is either hidden (sessions 3,6,9) or visible. The tenth session is a test where the platform is visible but has been moved. The DMS model was not simulated as the precise modification to be applied to the model was unclear (should the map-based strategy, or the gating network be affected? And how?). The fornix- and DLS-lesioned groups were simulated by respectively removing the map-based and the taxon strategy. The main characteristics of the groups' behavior is well captured: when the goal is visible, any strategy can lead to the platform; but when it is hidden, the fornix-lesioned group performs poorly; finally, during the test, the DLS-lesioned group performs poorly as it goes to the previous location, while the control group isn't as good as the DLS-lesioned one, as competition occurs between the two strategies. We refer to the original paper for more details on these two simulations [33].

These results were however obtained in simulations which, although in continuous state-space, were perfectly controlled and thus permitted a set of crucial simplifications:

- The model had perfect access to its position and orientation;
- Visual perception was also perfect, permitting the robot to distinguish without errors different landmark cues, and thus making it possible for the model to have taxon submodules which learned to select a movement direction in association to each specific landmark;
- The agent was a virtual point without a body surface, allowing holonomic motion.

Thus it is not clear whether the model can be applied to robotics in the real world, and whether it can still reproduce rodent behavioral performance and adaptivity in such circumstances.

Here we present the integration of this neurocomputational model in the Psikharpax robot, and the solutions adopted to cope with noisy perception and odometry. While in simulation, each strategy could individually solve the rat goal-seeking task but a combination of strategies was required to produce the same behavioral performance as the rat [33], here we find that each strategy can only partially but complementarily solve the task, and the combination of strategies permits to achieve the problem. In addition, we show that the previous strategy-shifting mechanism can adapt to environmental changes, but with slower performances than real rats. We finally add a meta-controller to the model which detects context-switching, permits faster adaptation to environmental changes, and allows to quickly restore previously learned behavior when a known context is presented again to the robot. Such meta-controller may constitute a better model of rat prefrontal cortical functions known to be required for adaptive strategy shifting [35, 36, 37]. It may also provide a more robust solution for strategy shifting in autonomous robots.

The first part of this paper gives a technical overview of the platform. The theoretical foundation of our work was verified by Dollé et al. [33] in simulation, based on almost perfect sensory input and simulated grid cells [38]. Therefore, the second and third parts of this paper present the equivalent navigation strategies and strategy selection mechanism for the real robot. The last part presents the results obtained in a series of robotic tests of the model.

2. Material and Methods

2.1. Architecture overview

Our software architecture was built on the ROS‡ - Robot Operating System - middleware. The robot runs the ROS core and an external quad-core machine is used for the visual system and the navigation strategies.

An overview of the software architecture of our model is given in Fig. 3. The system consists of six distributed subsystems, each consisting of one or more ROS nodes. As can be seen from Fig. 3, the central node of the system is the action selection node. This node interacts with the gating network (see Section 4) to decide upon the next action the robot will take.

Two additional mechanisms - guiding and obstacle avoidance - are not shown in the figure. The obstacle avoidance strategy is implemented as a reflex strategy to prevent the robot from leaving the environment. The guiding procedure is used to lead the robot towards and from the goal at the end of failed and successful trials (see Section 5.4).

2.2. Visual processing and localization

More details on the visual system are given in the Appendix of this article. Here we summarize how visual information concerning landmark cues in the environment is extracted to build a map of place cells for localization. An overview of the model is given in Fig. 4.

The robot is equipped with two small front-facing cameras with a total field of view of about 60 degrees. While real rats have side-facing eyes with a large field of view, their stereoscopic vision is limited to a region of about 76 degrees [39]. The choice for a small but stereoscopic view originally stems from experiments with saccadic eye movements on the Psikharpax platform [5]. We chose to keep this setup as it allows the robot to the distance of objects and it allows to extend the model to include attention. To overcome the limited field of view, the robot is programmed to turn its head around at regular intervals.

‡ ROS is open-source and can be downloaded from http://ros.org

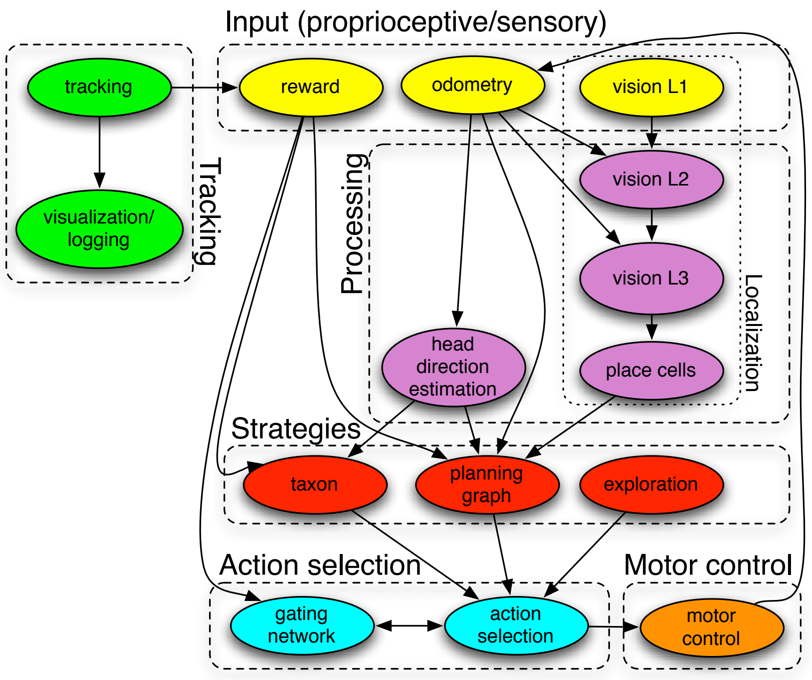


Figure 3. Simplified overview of the software architecture (only the most important nodes and connections are shown)

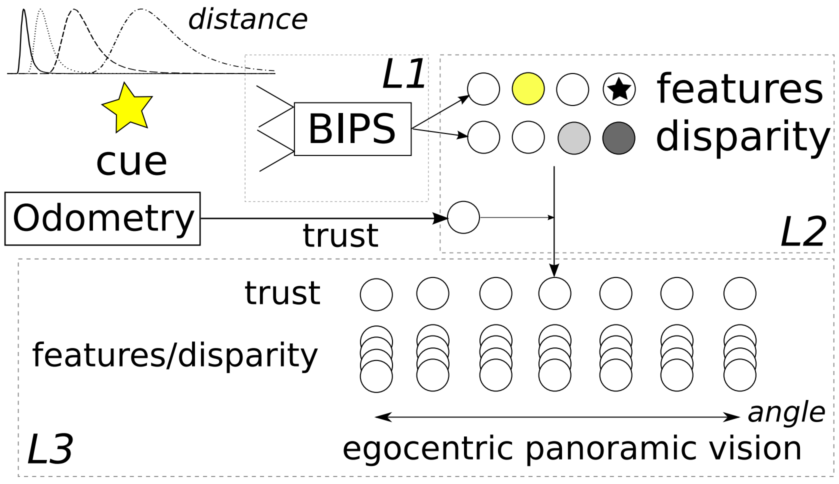


Figure 4. Concise overview of the visual system. In this example there is a brightly-colored star-shaped object at a distance of approx. 3.5 m from the robot's head. The robot sees this object through its two cameras directly connected to the BIPS hardware (L1). The BIPS hardware extracts feature information from the visual object and this information is coded on a set of feature neurons in the second layer of the visual system (L2). Based on the angle at which both cameras see the cue, the disparity neurons are activated to code the distance information (see also Fig. 6). The trust neurons are activated based on odometric information: if the robot's head is moving fast, the trust drops. There are disparity, feature and trust neurons for each direction within the field of view (not shown in the figure). Information in L2 layer is sent to layer L3 and integrated over different orientations to produce a 360 degrees view.

At the lowest layer of the visual system, the cameras are directly connected to an onboard electronic device, called the Bio-Inspired Perception System (BIPS, developed by Brain Vision Systems, BVS) [4]. This layer implements the retina and the first layers of the visual cortex by using a neural network with inhibitory connections to detect and track stable and saturated objects (see Fig. 5). This layer is shown as L1 in Fig. 4. Note that after this layer the raw image is discarded and only the detected objects are used.

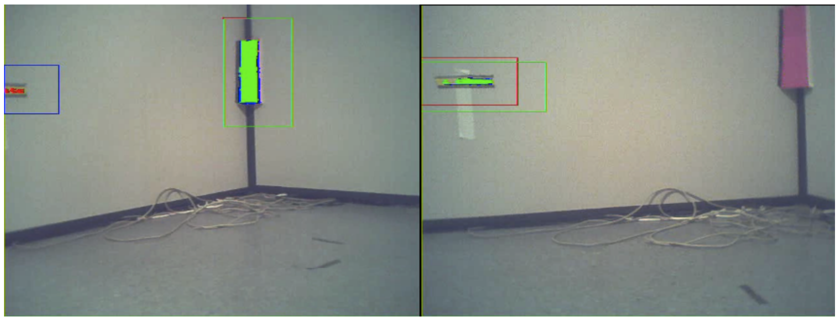


Figure 5. The same part of the environment seen from two different angles can result in an object not being detected. The system should cope with this limitation through the higher layers of the visual system.

The second layer of the visual system codes the visual information onto a set of feature neurons. For each object, the following features are extracted: size, vertical position, orientation, color and disparity. For each orientation within the field of view, such a set of feature neurons exists and a detected object activates the neurons in the direction in which it is seen. We use leaky-integrator neurons to low-pass filter the input. The disparity codes the distance of an object with respect to the robot. Four neurons are used to code disparity information. These neurons have a Gaussian activation function, centered around different disparities. This results in an activation function that has a large tail as function of the distance (Fig. 6). Hence the robot has more precise distance information on nearby objects.

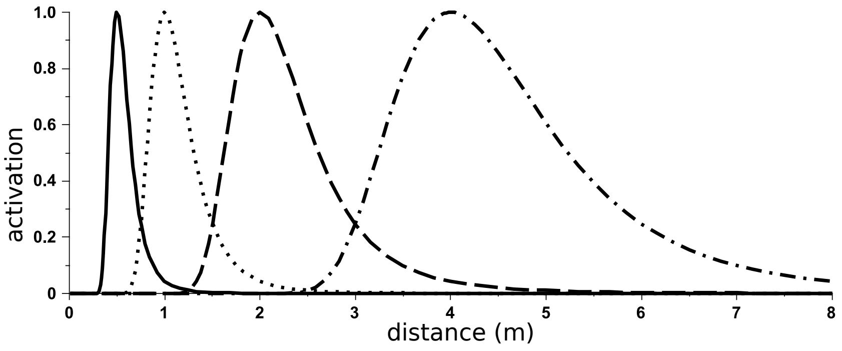


Figure 6. Activation function of the disparity neurons as a function of the distance. They are Gaussian as a function of the disparity, naturally resulting in a non-linear distance scale with higher precision for nearby objects.

Primates seem to use other types of disparity measures such as relative disparity between objects next to absolute disparity [40]. We tried to increase the performance of the visual system by adding disparity neurons with other activation functions (based on Gabor filters), but the quality of the resulting place cells (next Section) did not increase. This is probably due to the fact that enough information is already coded by the other feature neurons to distinguish places in the environment we used and that a non-linear training algorithm is used.

An important part of the second layer of the visual system are the so-called trust neurons. These neurons modulate the output of the second layer to the third layer of the visual system. The idea is to suppress noisy inputs when the visual input is unreliable. This happens when the head of the robot moves too fast, as the neurons in the first (tracking units) and second layer need some time to stabilize. This is easily detected by the odometric system and hence the odometric system is used to modulate (suppress) the connections between the second and third layers when necessary. The faster the head movement, the less reliable the visual information. This prevents the third layer of the visual system from being influenced by unreliable information.

The third level of the visual system integrates the information from the second layer by combining it with odometric information. This results in egocentric panoramic information on the environment.

2.3. Visual place cells

The output from the neural network visual layers is high-dimensional (about 800 neurons). Because simple rate-coding neurons were used, the output can be seen as a vector representing egocentric visual information integration. To construct non-directional place cells, such output vectors were summed over all orientations to activate the same neuron (i.e. a place cell). The problem was therefore reduced to a dimensionality reduction or clustering problem. This subsystem is indicated as *PC* on Fig. A1 in the Appendix.

In a first version of the simulation model [33], ad hoc place cells were used, and thus the dimensionality reduction / clustering problem was not addressed. In [41], a model of the hippocampus [38] was used to autonomously create the place cells. It is based on a competitive Hebbian-like learning rule: a number of random place cells are created, during the learning phase, the place cells specialize for particular input patterns using a sparseness-based Hebbian rule, which only allows the most active input neurons to reinforce their connections.

Such an approach works very well when the number of input neurons and distinct patterns is not too high and the patterns are well characterized by their most active neurons. In our case however, the input can be noisy with typically large, but meaningless values for a few neurons in the input. When the sparseness function from [41] is applied to such an input, the noise is reinforced, while useful neurons will be ignored.

We therefore needed a technique that learns the input patterns by evaluating the whole set of input neurons instead of only the most active ones. We initially tried linear approaches such as Principal Component Analysis [42] to check if the inputs were linearly separable. At most 4 to 5 regions could be consistently separated. This is insufficient for good performance, as the place fields of the place cells would be too large (only 4 to 5 distinct zones). Indeed, the gating network takes input from the place cells and hence its precision is limited by the place cells.

2.3.1. Implementing a Self-Organizing Map A popular non-linear alternative for clustering are Self-Organizing Maps (SOM)[43]. The goal of the SOM is to move the neurons in the high-dimensional input space to approach the topology of the input. For each input the Euclidean distance between the input and each neuron of the SOM is computed. The closest neuron is called the best-matching unit (BMU). The SOM is then updated by moving the BMU closer to the input (weighted sum) as well as its neighbors. In order to avoid using a fixed number of neurons in the SOM, we used the Growing Neural Gas (GNG) algorithm [44]. GNGs are created incrementally by inserting a new neuron after a number of input samples by splitting the neuron with the largest accumulated error (sum of distances) into two new neurons. The topology itself is also learned by keeping the neighborhood of the neurons up to date.

We used the GNG algorithm to train the weights of an artifical neural network (see Appendix for a detailed description of the implementation). While we do not assume that there is a direct biological equivalent of this training algorithm, nor the activation function (which is based on the Euclidean distance), we do not think that our model makes unrealistic assumptions about the role of the hippocampus in categorizing different places. As [44] indicates, the GNG algorithm can be seen as a form of (non-linear) competitive Hebbian learning, which is the main reason why we chose this algorithm. Because the main interest of this work lies in the strategy and context switching mechanisms, we did not investigate how exactly one might implement the GNG algorithm with a biologically plausible neural network. However, this should not be a problem, as the algorithm is straight-forward and only depends on the computation of the Euclidean distance (or another distance measure), the creation/removal of edges and updating a local error measure. A further argument for using a non-linear approach is that non-linear algorithms can often be cast as a linear technique working in a larger (possibly infinite) feature space by simply replacing inner products with a kernel function (i.e. the kernel function computes the inner product in a different space)[45]. For example, the kernel trick is often applied to PCA (Kernel PCA) [46] for which a Hebbian version already exists [47].

We found that the GNG technique is very flexible. When one forces the GNG to use only a small number of neurons, the GNG creates large continuous place fields (Fig. 7). When more neurons are available, the place fields are smaller.

The neurons from the GNG layer project onto the planning strategy and the gating network. The resulting place cells for a GNG layer with 12 neurons are shown in Fig.

7. One can see that the best (i.e. most restricted to a particular zone) place cells are found near the borders (similar to Fig. 9 (D) of [9]). This is due to the fact that there are no intramaze cues, which causes observations near the center to be more similar and thus less place cells are created in this region. We used a higher number of place cells in our experiments to increase the precision of the planning strategy (Section 3.1) and the gating network (Section 4.2). However this phenomenon still occurs (e.g. Fig. 8(b)), causing topological maps to be less dense near the center.

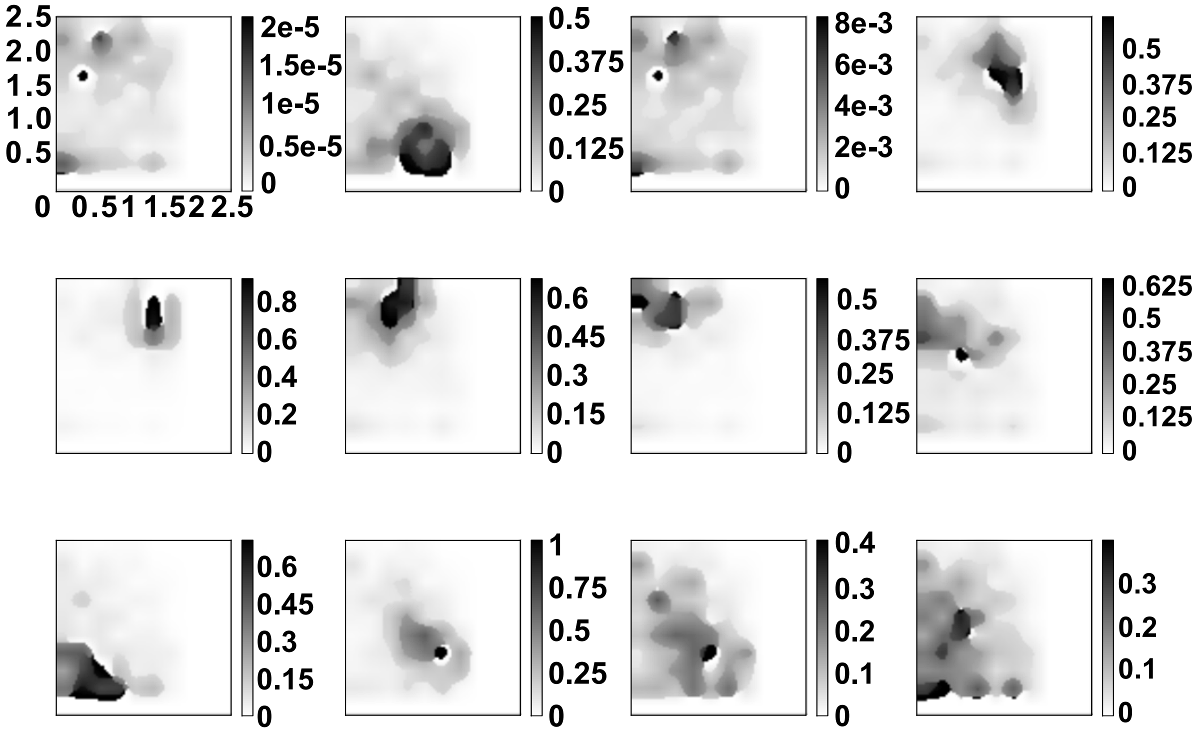


Figure 7. Heat map of 12 place cells (GNG with 12 output neurons), with smooth activation (Eq. A.4). Note that there are 2 place cells (top left and first row third from left) which only have very weak activations but large receptive fields. These cells will thus not be important as their activation will be negligible (the topological map will discard them). The axes of each of the images give the position in meters of the robot's head as recorded by a ceiling camera. Note that there are more and more precise place cells coding for locations near the borders of the environment.

2.3.2. Experimental testing of the place cell system To get a rough estimate of the usefulness of the place cell system, the robot was put at 40 random places in the environment and indicated the activation of its place cells (for this experiment, we trained 100 place cells). The Euclidean distance between the real position of the robot and the center of activation (computed by averaging over a large training set using a ceiling camera) of the most active place cell (binary activation) was computed to estimate the precision of the place cell coding. The mean distance was found to be 16.5 cm with a standard deviation of 8.5 cm. When near the border of the environment, the mean lies around 11 cm, which is very good, but when approaching the center of the environment the mean distance becomes considerably larger (between 20 and 30 cm). In [9] the authors found a mean error of 6 cm but in a smaller (0.8 m by 0.8 m) environment.

In the ideal case, one would expect the place fields to be evenly distributed. Hence, to evaluate our place cells mechanism, we consider 100 points picked randomly from a uniform distribution on a rectangle of 2.5 m by 2.0 m. The expected distance from any such point within the rectangle to the nearest point out of the 100 randomly chosen points is about 12 cm. The expected distance to the second closest point is about 18 cm and 22.5 cm for the third closest point.

This indicates that the presented place cell system performs reasonably well, compared to the ideal, uniformly distributed case.

We tested the system with different number of place cells, splitting the data in a train and test set. We found that 100 place cells is about the maximum one can obtain in our environment without overfitting the training data. Moreover, for higher numbers of cells, the classification results on the test set did not increase. Thus for all the next experiments with the multiple navigation strategy model, the number of place cells was fixed at 100. In the next section we present the navigation strategy which is based on information from the place cells layer.

3. Navigation strategies

3.1. Planning expert

The [33] model best replicated experimental results using a planning algorithm for the place strategy, rather than a place recognition triggered response one. It is organized as follows: a place cell module, simulating the hippocampus, is in charge of learning internal representations of places in the environment using sensory inputs; a graph module (topological map), simulating the prefrontal cortex, learns by means of a Hebbian rule the directions of movement, which are used to go from one place to another. When the goal has been found at least once, a diffusion of activity in this graph originating from the goal node generates a gradient which, when followed, leads to the goal with the shortest path [48, 49].

The simulation model from [33] uses distinct representations for place cells and nodes in the graph module, because they differ in function and precision. More precisely, a simple single-layer network trained with a competitive Hebbian-like learning rule is used to activate the topological map nodes based on the place cells' activation. Because of this layer, the number of topological map nodes (around 100) was typically a factor 10 to 20 lower than the number of place cells.

Because of the encouraging results from this simulation model, we chose to adapt it to the physical platform. Several modifications needed to be made to make this feasible, which will be explained in this and the next sections.

The maximum (useful) number of place cells on the physical robot is limited by the quality of the sensory input (see previous section). Because the goal is relatively small and a high level of detail in both the planning strategy and the gating network is advisable to get precise results, we mapped the place cells directly onto the nodes in the topological map to get the maximum resolution. This means that there is no additional training to map the place cells onto the topological map nodes (1-to-1 connections).

We initially also tested (not shown) the system with a coarser representation of space (using an additional layer trained with competitive Hebbian-learning) for the gating network, yielding similar, but less detailed results than the ones presented in this article. The added benefit of the direct mapping from place cells to nodes in the topological map, is that analyzing the results is easier as the space representation is the same throughout the system (gating network, place cells and topological map). To underline the functional difference between nodes in the topological map and place cells, we use the notation n^{PFC} for nodes in the map (referring to the prefrontal cortex) and n^{PC} for place cells.

3.1.1. Learning the topological map

During an exploration phase, the topological map learns connections between nodes by Hebbian learning. For this, two types of information need to be stored, the relative angle between two nodes and their mutual distance (Fig. 8(a)). To store this information, we use two sets of transition neurons for each connection between nodes [50, 51]. There are N_{ANG} transition neurons (per set) for directional information and N_{DIST} for distance information. Each node initially has connections (transition neurons) to every other node with zero weights. The transition neurons are stored in a vector $v_{k,l} = [v^1_{k,l}, \dots, v^{N_{ANG}}_{k,l}, v^{N_{ANG}+1}_{k,l}, \dots, v^{N_{DIST}+N_{ANG}}_{k,l}]^T$, i.e. the subscript k, l indicates the transition from node k to l and the superscript is the affected transition neuron (orientation or distance information).

As for the distance information, the angular information is stored in a set of N_{ANG} neurons with Gaussian activation functions centered around a fixed directions. We define the vector $b = [b^1, \dots, b^{N_{ANG}}, b^{N_{ANG}+1}, \dots, b^{N_{DIST}+N_{ANG}}]^T$ similar to $v_{k,l}$. The first N_{ANG} elements of b code the angular information between locations. The last N_{DIST} neurons contain the distance information between 2 locations. I.e. a vector $b(t_0, t_1)$ contains the activation of the transition neurons to the location where the robot is at time t_1 from the location where the robot was at time t_0.

Now, to update the neurons $v^i_{k,l}$, we iterate over a trajectory of the robot and update the weights to the transition neurons using a simple learning rule:

$$\Delta v^i_{k,l}(t+1) = \begin{cases} (1-\delta_{k,l}) H(n^{\bar{L3}}_{conf} - \beta) b^i(t_k, t) & \text{if k = last} \wedge \text{l = winner} \\ 0 & \text{else} \end{cases}$$

Here $\delta_{k,l}$ is the Kronecker delta, to prohibit connections from a node to itself. $H(x)$ is the Heaviside step function used to prevent updating the graph when the confidence of the place cells ($n^{\bar{L3}}_{conf}$) is below a threshold β. *last* is an index referring to the node in the topological map where the robot was when $n^{\bar{L3}}_{conf}$ was above the threshold for the last time, i.e. the previous location.

For planning in this graph only the shortest transitions between nodes are kept, based on the distance coding neurons. For the results presented here, we fixed the maximum number of transitions per node to 6. A learnt map is shown in Fig. 8(b). While we explained this process as a sequential algorithm (recruitment of nodes, learning of transitions, competition), it can be done online by simply adding an additional transition usage intensity neuron $v^{0}_{k,l}$ increasing in activation when the robot moves from k to l, combined with a decay rate or competition factor (i.e. $-\psi v^{\bar{0}}_{k,l}$) to the previous equation, which also prevents the weights from increasing without bounds.

3.1.2. Using the planning expert

In order to plan in this graph, the model maintains a set of neurons g_i, one for each node corresponding to the reward received at each of the locations. A leak rate is added so that the robot can navigate in an environment with changing reward locations. g_{winner} is the neuron assigned to the node at the robot's current position.

$$g_j(t+1) = g_j(t+1)(1-\tau_{forget}) \tag{1}$$

$$g_{winner}(t+1) = g_{winner}(t)(1-\tau_{learn} n^{PFC}_{winner}(t)) + \tau_{learn} n^{PFC}_{winner}(t) R(t) \tag{2}$$

The second value associated with a node is the diffusion value $d_j(t)$ and this value is used to implement a shortest-path algorithm. The activation from the goals diffuses or spreads out [48, 49] over the other nodes (Fig. 8(a)). To compute the equilibrium state efficiently, we used a modified Floyd-Warshall algorithm [52], where $d_j[iter]$ is used to refer to the value of d_j at iteration $iter$:

Algorithm 1 Computing the diffusion values

```
$iter = 0$
**for** $i = 0; i < N_{PFC}; i = i + 1$ **do**
  $d_i[0] = g_i$
**end for**
**while** $iter < N_{PFC} - 1$ **do**
  **for** $i = 0; i < N_{PFC}; i = i + 1$ **do**
    $d_i[iter+1] = \max(d_i[iter], \max_{j \in neighbors(i)}(d_j[iter])\iota)$
  **end for**
  $iter = iter + 1$
**end while**
**for** $i = 0; i < N_{PFC}; i = i + 1$ **do**
  $d_i = d_i[N_{PFC} - 1]$
**end for**
```

This algorithm is only run when $n^{\bar{L3}}_{conf} > \beta$, i.e. when the robot has a high trust in its current position.

The algorithm finds the shortest path to a maximum goal value (g_j) in terms of the number of intermediate nodes and the goal values. However, multiple maxima can exist as more than one node can code for the goal location due to aliasing. To find a path to the closest maximum (goal), one starts from the current node (n^{PFC}_{winner}) and chooses the neighbor with the highest value as the next node on the path. The topological map then proposes the mean of angles $\Phi^P(t)$ computed from the activation of the direction transition neurons associated with the connection from n^{PFC}_{winner} to its most active neighbor.

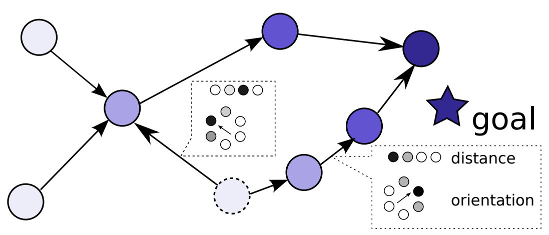
In practice, we add a slight twist by storing a complete path towards the closest maximum. Next the robot computes the relative positions of the nodes of the path in the egocentric frame. It then tries to follow this path by moving sequentially to the position of the nodes based on the odometry. This allows this map-based strategy to persist even in the absence of sensory input and is equivalent to the basic algorithm when sensory input is reliable.

Fig. 8(c) and 8(d) show the same map with two different goal locations, the arrows represent the direction towards the goal from each node in the graph, as computed by the planning algorithm. In the situation shown in Fig. 8(c), the robot faces an aliasing problem: there are multiple optima (high diffusion values) near the goal, and starting from the east part of the environment, the robot may plan trajectories towards different nodes apparently close to the goal. As a consequence and as will illustrate the experiments with the whole model, the planning expert can adapt fast but remains approximative. It can quickly learn trajectories towards the coarse area around a new goal location. But these trajectories may not be precise enough to reach the goal and other experts using different strategies may be more relevant in more precisely attaining the goal location.

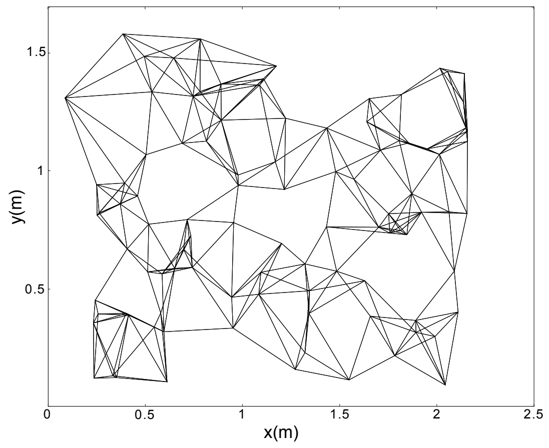
3.2. Taxon strategy

The second strategy is implemented in the so-called taxon expert. It learns to associate proximal visual cues with actions using a standard Q-learning algorithm [53]. While in previous work with noisy continuous state space in complex mazes we employed a multiple-module reinforcement learning approach for the taxon system [54], here we used a simplified taxon in order to test the ability of our meta-controller to switch between complementary strategies. While rodents' hippocampus-dependent place strategies rely exclusively on distal landmarks - which are far and outside the maze and thus are more stable to constitute the anchoring of a cognitive map -, the taxon strategy consists in learning directions of movements in association with intra-maze proximal landmarks [55, 18, 56].

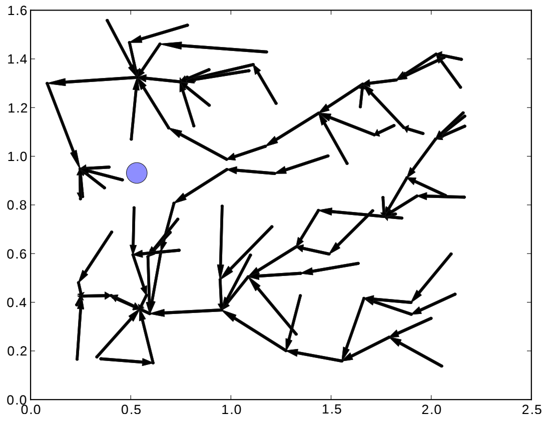
Thus here, the taxon expert can only perceive the goal. In order to prevent the robot from seing the goal when it is far - thus distal -, we make its perception noisy. Thus the relative position of the goal is seen by the taxon as a Gaussian which decreases in height and variance as the distance increases (until it drops below a threshold). While conceptually simple, it is important to note that the taxon expert initially does not know that it should move towards the stimulus it receives. So the taxon (like any other



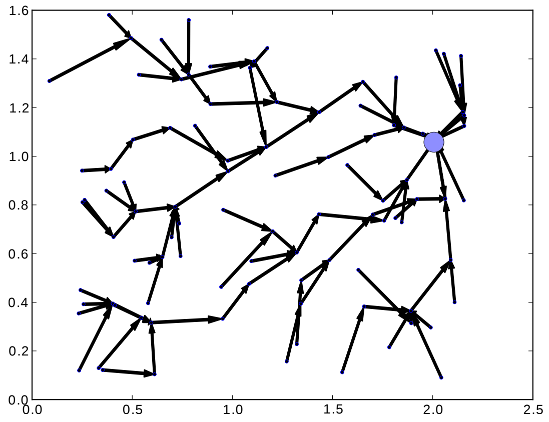
(a) Principle of the topological map. The large circles are the nodes in the map. The goal is on the right and the color of the nodes correspond to the diffusion values. Direction transition neurons (small circular pattern) and distance transition neurons (4 small circles) are shown for two connections. A path planned from the dashed node can be ambiguous for one node due to equal diffusion values. The robot chooses a next node arbitrarily in this case.



(b) Topological map constructed by the robot. Vertices correspond to nodes in the map, edges are paths the robot can use. The maximum number of neighbors per node was fixed at 6.



(c) Paths towards the goal shown from each node in the topological map. In this map there is aliasing around the goal, resulting in very low performance of the planning expert on the left side of the environment, near the goal (circle). Axes are in meters.



(d) Paths towards the goal shown from each node in the topological map. This map predicts good performance of the planning expert with only a few regions with ambiguities.

Figure 8. Overview of the topological map. The map is the same in (b), (c) and (d). (a) Principle of operation (b) Constructed topological map after exploration (c) and (d) Path planning using the topological map. The qualitative difference between (c) and (d) is caused by the change of the goal location. In (c) there is a lot of aliasing around the goal location (multiple high diffusion values), resulting in two disjunct path-planning trees and sub-optimal paths near the goal (e.g. there is an optimum around (0.5, 0.4)). In (d) the topological map works very well (one tree with almost every edge leading the robot closer to the goal). Note also the difference in map quality between the simulation model (e.g. Fig. 8d and 12b in [33]) and the robotic platform. This is caused by the extensive exploration phase in simulation and the noise-free simulation environment.

learning expert) learns at the same time as the gating network.

The update equations of the taxon expert are based on [33], with slight modifications to adapt the strategy to the robot platform.

The possible directions (continuous) the robot can take are coded on the N_{dir} action cells a_i. $w_{i,j}$ are the Q-values, which are used to associate input orientations with output orientations:

$$a_i(t) = \sum_{j=1}^{N_{GP}} r_j^{GP}(t) w_{i,j}(t) \tag{3}$$

By taking the mean of angles Φ^T, we obtain the proposed action of the taxon:

$$\Phi^T(t) = \arctan\left(\frac{\sum_i a_i(t)\sin(2\pi i/N_{dir})}{\sum_i a_i(t)\cos(2\pi i/N_{dir})}\right) \tag{4}$$

$\Delta w_{i,j}$ is the update rule for the Q-values, based on the reward-prediction error δ^{taxon} and the eligibility traces $e_{i,j}^{taxon}$:

$$\Delta w_{i,j}(t) = \eta \delta^{taxon}(t) e_{i,j}^{taxon}(t) \tag{5}$$

The eligibility traces are used to speed up learning by storing previous state-action pairs and by using action generalization. The action generalization is given by r_i^{AC} and is based on the *executed* action, which can be the action proposed by a different strategy (see Section 4.2) §. Thus, the taxon strategy also learns if another strategy performs well. Note that in practice one uses a wrapped Gaussian as the difference between circular values has to be computed.

$$\delta^{taxon}(t) = R(t+1) + \alpha \mathrm{max}_i a_i(t+1) - a(t) \tag{6}$$

$$e_{i,j}^{taxon}(t+1) = r_j^{GP}(t) r_i^{AC}(t) + \kappa e_{i,j}^{taxon}(t) \tag{7}$$

$$r_i^{AC}(t) = \exp\left(\frac{-(\Phi^*(t) - 2\pi i/N_{dir})^2}{2\omega^2}\right) \tag{8}$$

The input of the system are the N_{GP} goal direction neurons r_j^{GP}, which replace the landmark cells r_j^{LC} from [33]. r_j^{GP} are again samples from a wrapped Gaussian at equally spaced angles. The width of the Gaussian increases as the robot approaches the goal as does its amplitude. The amplitude of the Gaussian drops below the threshold between 0.5 and 0.75 m.

Because the goal is the only landmark that can be seen by the taxon expert, the taxon is essentially the same if the input orientations are egocentric or allocentric. In this work, we used an allocentric taxon because the robot approaches the goal with its head pointing in the direction of the goal and hence learning the correct action to take when the goal is behind the robot slower (e.g. Fig. 15 near $-\pi/2$).

3.3. Exploration strategy

This is a simple strategy that proposes random directions and which serves two purposes in our model. We do not however consider this strategy equivalent to the much more

§ Note that there is a small error in Eq. 7 of [33]. It is indeed $\Phi^*(t)$ instead of $\Phi^{\mathrm{T}}(t)$.

complex exploratory behavior in rodents. It has been shown that rats, in a new environment, observe recurrent patterns such as spending the first minutes establishing a home base [57], then moving out slowly in a zig-zag way, and coming back home in a straight line. We just use a simple random exploration strategy in order to make sure the robot would cover most regions of the environment. Because each action proposed by this strategy persists for 5 steps, it allows for some exploration and to break out of looping behavior. Secondly, this strategy allows us to evaluate the performance of an expert with respect to chance level. Because its behavior is random, we expect every other strategy to perform at least as well in most regions of the environment. As we shall see, this is not generally valid, as a random strategy might outperform a more elaborate strategy in certain situations.

4. Strategy selection

4.1. General framework

The strategy selection model of [33] is based on the premises that rodents have multiple navigation strategies at their disposition to reach a goal and that they are capable of switching between them. These strategies coexist and are learned in parallel and independently, while a strategy selection mechanism learns to associate perceptions (or situations) with a preferred strategy by means of a Q-learning algorithm.

Accumulating evidence support the hypothesis that Q-learning is a plausible mechanism by which part of mammals brain learn by reinforcement [58]. Neural correlates of action values (similar to Q-values) have been found in the basal ganglia [59]. And correlates of action-dependent reward prediction errors (consistent with Q-learning) have been found in dopaminergic neurons [60] as well as in the medial prefrontal cortex [61].

A global overview of the system is shown in Fig. 9. The basic idea is that we have a number of strategies or experts providing the next action to take (following the given strategy) at each time step, while the action selection network selects one of these actions, based on the current situation. A Q-learning algorithm based on the simulation model from Dollé et al. [33] is implemented to allow the robot to associate a state with an optimal strategy. The so-called landmark cells of the simulation model (Fig. 2) necessitate a mechanism to identify and track distal landmarks in the environment, while here, due to perceptual aliasing and noise in physical experiments with the robot, we used the global configuration of distal cues to build place cells without requiring the recognition of individual cues (2.2). Thus the connection of the visual system to the gating network is left as future work (see Section 7) and in our current model we only use the place cells as input to the gating network. This has the benefit of allowing us to visualize the relationship between locations and preferred strategies as the Q-matrix associates place cells with experts.

The actions proposed by the strategies are the *common currency* in the model

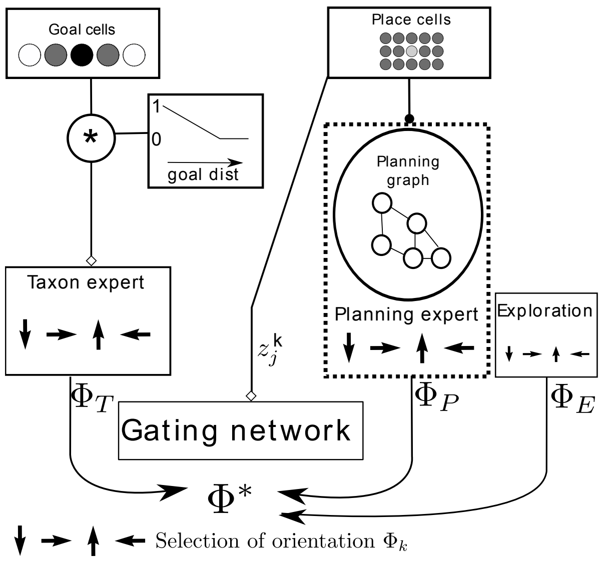


Figure 9. Overview of the gating network with taxon, topological map and exploration experts.

as they are the generic information used to evaluate the performance of each strategy [33]. In our model, these actions are the next (egocentric) direction to follow. This way, the robot selects a new orientation provided by the strategy that was deemed the winner among the strategies, turns and moves forward for a fixed distance, observes its new situation (state) and the reward (if any) observed and finally updates its strategy selection network.

The neural network used to implement the strategy selection is a single-layer network, with the neurons coding for the current situation or perception fully connected to each of the output neurons, one for each available strategy. The layer of output neurons with the connections to the input neurons is called the *gating network*, as it only lets one action through at each time step. One can of course consider the winner of a lower-level gating network as the input of the current network to create a hierarchical selection mechanism.

4.2. Gating network

The gating network computes the so-called gating values $g^k(t)$, one for each strategy k. The Q-values are stored in a matrix $z_j^k(t)$, associating inputs from the place cells with gating values:

$$g^k(t) = \sum_j^{N_{PC}} z_j^k(t) n_j^{PC}(t) \tag{9}$$

Instead of adopting the winner-take-all policy ($\Phi^*(t) = \Phi^{\mathrm{argmax}_k(g^k(t))}(t)$) from the simulation model to select the winning strategy for the next action, we generalize such principle so that the selection probability of an expert increases with its relative gating value:

$$\mathrm{P}(\Phi^*(t) = \Phi^k(t)) = \frac{g^k(t)^\zeta}{\sum_i g^i(t)^\zeta} \tag{10}$$

Here $\Phi^k(t)$ is the action proposed by expert k at time t. $\Phi^*(t)$ is the final action proposed by the gating network. Note that this action is not always the executed action, as higher priority mechanisms can override the gating network (i.e. obstacle avoidance or guiding). For $\zeta = \infty$ our action selection mechanism is equivalent to the one from [33]. For $\zeta = 1$, one obtains the action selection mechanism from [62]. For all experiments in this paper we set $\zeta = 1$, except for Fig. 30(a) and 30(b) for which $\zeta = \infty$ to show the behavior of the robot when it follows its learned optimal policy.

The advantage of introducing some randomness in the action selection is that slower learning strategies can catch up with fast learning strategies when they start to perform better only after a long time. With a winner-take-all strategy, one might have to wait for convergence before a slower learning, but an optimal strategy can increase its weights beyond these of a faster learning, but suboptimal strategy. This is also biologically relevant since choosing a suboptimal strategy from time to time allows for exploration of unfamiliar alternatives [63].

Learning is sped up by using action generalization and eligibility traces. The equations for these techniques were taken from [33]. However, a substantial difference lies in the equation to update the eligibility traces. Whereas sensory input is always reliable in the simulation model, it is not true in general with the real robot. To incorporate this fact in our system, the eligibility traces are modulated by the trust neurons introduced in 3.1:

$$e_j^k(t+1) = n_{conf}^{\bar{L3}}(t)\Psi(\Phi^*(t) - \Phi^k(t))r_j^{PC}(t) + \lambda e_j^k(t) \tag{11}$$

To modify the Q-values, a modified Q-learning algorithm [64, 53] is applied:

$$\Delta z_j^k(t) \quad = \xi\delta(t)e_j^k(t+1) \tag{12}$$

$$\delta(t) \quad = R(t+1) + \gamma \max_k(g^k(t+1)) - g^{k^*}(t) \tag{13}$$

$$e_j^k(t+1) = \Psi(\Phi^*(t) - \Phi^k(t))r_j^{PC}(t) + \lambda e_j^k(t) \tag{14}$$

where ξ is the learning rate of the algorithm and δ the reward prediction error.

The reward prediction error δ is based on the observed reward when performing action $\Phi*$ and the future expected reward (g^{k^*} is the activation of the winning output neuron and γ the future reward discount factor). The eligibility trace e_j^k reinforces previously selected strategies and the strategies proposing a direction close to the one proposed by the winning strategy [53]. Here λ is the decay factor for previous winning strategies and Ψ a Gaussian function:

$$\Psi(x) = \exp(-x^2) - \exp(-\frac{\pi}{2}) \tag{15}$$

The gating network is a simple but effective way to combine competition and cooperation between strategies. While the gating network itself only directly provides competition, the strategies cooperate by sharing rewards and their actions (e.g. the taxon uses the executed action for learning, instead of its proposed action (Eq. 8)). Hence the gating network is advantageous for strategies as they can learn from each other, while at the global level the performance can also increase because the best performing strategy can be used in each situation.

4.3. Context-switching meta-controller

The advantage of using a gating-network for strategy shifting is the possibility to memorize that strategy A is efficient in a subpart X of the environment while strategy B is relevant in subpart Y [62, 33]. However, since this is learned through Q-learning, noisy information perceived by a physical robot may render this process very slow. In addition, a change in the environment requires to unlearn A-X and B-Y associations (which can also be very slow). As a consequence, these associations cannot be used again if the environment comes back to its previous state (i.e. the animat cannot recall what is has previously learned).

To overcome this limitation, we implemented a simple context switching mechanism. The idea is that a change in the environment (e.g. a change in goal location) will be quickly reflected in the profile of diffusion of goal information in the topological map once the robot found the new goal. Such profile can be identified as a context, and the system can recognize a previously experienced context when the goal is set back to its initial position and the model diffuses such goal location in the topological map. Each time the model detects a new context, it will create a new memory component to store values of the gating network in the new context, without erasing values of the gating network associated to the previous context. This part of the model may be viewed as a primitive prefrontal cortex-based cognitive control mechanism allowing to associate different contexts to different tasksets [65] (see Section 7 for a detailed discussion).

In practice, before every step, the gating network decides upon the context it is working in. For this it uses the current vector of diffusion values d from the planning strategy. The gating network now stores a set of Q-value matrices $z^i_{i,j}$, and associates a diffusion vector u^i with each matrix. The current context is now chosen as follows (d is the current diffusion vector):

$$v^i = \frac{d \cdot u^i}{\|d\|\|u^i\|} \tag{16}$$

$$z^*_{k,j} = z^{\mathrm{argmax}_i v^i}_{k,j} \tag{17}$$

When $\max_i v^i$ is below a threshold, a new context is recruited.

5. Experimental setup

5.1. Introduction

The robot is allowed to move in an open 2 m by 2.5 m environment (Fig. 18). There are only extra-maze cues (we tested with 10, 13 and 18 cues) and the position of the robot is defined by the position of its neck. The egocentric reference frame has the neck of the robot as its origin and the orientation is defined by the direction of the head.

5.2. Action selection

The robot makes discrete movements, moving 10 cm at each time step. The action selection mechanism is a simple finite state machine that waits for all strategies to propose an action (an egocentric direction) and then activates the gating network to find the winning strategy. The action proposed by the winning strategy is then executed, except if a higher than normal priority mechanism proposes an action (guiding/obstacle avoidance). After an action is performed the reward is used by the gating network and all strategies to update their learning parameters.

5.3. Reward

The reward node is a simple node processing information from the ceiling camera. When the robot's head passes through the zone defined as the goal, the reward is 1, else the reward is 0. In all experiments the goal diameter was set to 20 cm (314 cm^2, or 1/160th of the environment).

This is a global reward signal, shared by all strategies and the gating network. Strategies learn by updating their parameters using the global reward.

5.4. Experiments

All experiments consist of two phases. During the first phase the robot explores the environment. In this phase we force the robot to visit a number of locations in the environment so that enough information is available to construct place cells and a topological map (but no diffusion values nor goal values). The gating network and the navigation strategies are not active and there is no reward in the environment. This phase can be common to multiple experiments. During the second phase, the robot is put at a random location in the environment and the gating network and navigation strategies are turned on. In this phase the strategies and the gating network use the place cells and topological map to learn to navigate towards a goal.

A goal location is chosen and the robot learns to appropriately coordinate navigation strategies to reach it. During an experiment, after each failed trial (i.e. the robot does not find the goal after 5 meters of movement), the guiding procedure forces the robot towards the goal to show the goal location and thus speed up learning, similar to the procedure used by experimenters in rodent laboratory tasks. This is achieved by using the ceiling camera that tracks the robot's position and provides actions leading towards the goal in a straight line. The robot learns the result of this movement as if it had decided itself to perform it. Similarly, when the robot has received a reward (i.e. succesful trial), the guiding procedure guides the robot away from the goal to a new starting location at least 0.5 m away from the goal, mimicking the beginning of a new *trial* in rodent laboratory tasks.

All strategies and the gating network perceive the actions proposed by the guiding mechanism (there is no difference between an ordinary action and a forced action), so learning is performed in the usual way.

In the last experiments presented in this paper, once the robot has learned to reach the goal directly from its different starting position, the goal location is changed and the robot shall adapt its behavioral policy to this new condition. Finally, once the robot has learned the new condition, the goal is moved back to its initial location in order to test the ability of the robot to restore previously acquired behavioral policy.

6. Results

In this section we discuss the results we obtained on the Psikharpax platform. We will empirically show that our model can easily learn to associate a state with the best performing strategy in that state. The experiments were chosen to clearly show that the system works correctly (i.e. many of the results are predictable), instead of reproducing a complex protocol for which the evaluation of the quality of the model is inherently much harder to evaluate.

However, the experiments with multiple parallel strategies are in no way simple or unrealistic, given the limited and very noisy sensory input. Our results prove that with some small modifications, the Q-learning mechanism for the gating network from [33] works well on a real robot platform.

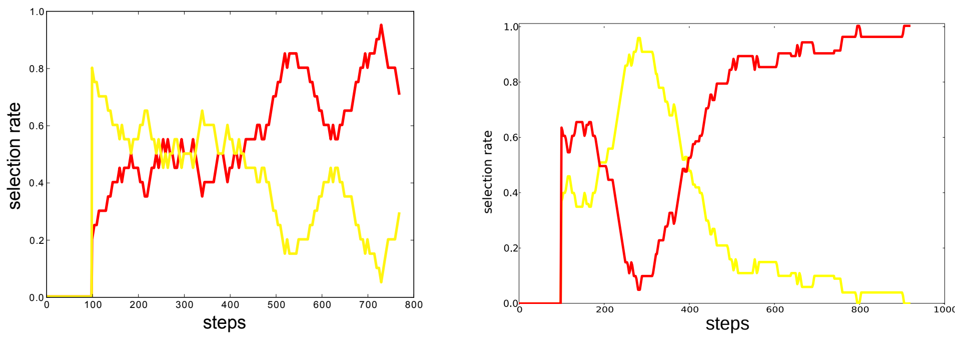
We first make a series of experiments to test the capacities of individual strategies (first planning, then taxon) to learn the goal location and lead the robot towards it. In these experiments, the studied strategy is combined with the exploration (random) strategy in order to compare its performance with random movements of the robot, and to test the ability of the gating network to learn to stop selecting the exploration strategy when another strategy can lead the robot to the goal. In the next experiments, the taxon and planning strategies work in parallel and the gating network successfully learns which strategy is the most efficient in each subpart of the environment. In response to an environmental change (i.e. change of the goal location), the gating network manages to unlearn previous associations of strategies to subparts of the environment and to learn new ones, but this process is very slow. In a last experiment, we add the context-switching meta-controller to the system and show that it manages to adapt faster to environmental changes and to restore previously learned associations when a previously experienced context is again presented. Finally, we analyze in more details a set of examples of cooperation between strategies produced by the system and allowing the robot to execute successful trajectories towards the goal.

6.1. Planning expert and exploration strategy

In this experiment, we connected two strategies to the gating network. The first one is the planning expert as introduced in the previous chapter. The other one is the random exploration strategy.

The goal of this experiment was to verify if the gating network could effectively learn to suppress a suboptimal strategy (the exploration strategy). Because the goal was relatively small and does not fall precisely on the center of activation of a place cell, the planning expert was only capable of efficiently guiding the robot to a zone around the goal. In other words, when the planning expert had reached a node in its map with higher diffusion values than any of its neighbors (given that the goal values are meaningful), the robot was not necessarily at a location where the reward is given, but only in the neighborhood. As a consequence, the robot's behavior remained random near the goal location - produced by a combination of the exploration expert and the planning expert which proposed random actions when it had attained the node in the diffusion vector with the highest rate.

Fig. 10 shows the selection rate for the two experts as a function of the number of steps taken for two experiments (different environments and goal locations).



(a) Less precise topological map, resulting in an overall higher selection rate of the exploration strategy

(b) Very precise topological map, the planning strategy has very good performance

Figure 10. Selection rate of both strategies during learning. The dark line represents the planning expert, the light line the exploration strategy. The horizontal axis indicates the current step (time). The transient phase ends for both experiments after about 300 to 400 actions.

After an initial transient phase during which most of the Q-values were still small and meaningless, the system quickly converged to a regime in which the planning strategy was selected almost all the time. During this transient phase both strategies had similar performances because the diffusion values were not meaningful yet. We see that this happened for both experiments, but with a different (random) initial transient phase. However the experiment from Fig. 10(a) which was slightly shorter than Fig. 10(b) and not yet fully converged had a higher selection probability for the exploration expert even after several hundred actions. After 300 to 400 actions (goal reached approx.

5 times) the gating network started to move towards its steady state. The Q-values would continue to increase but their relative values stayed stable.

To explain the observed difference in selection rates between experiments, we analyzed the locations in which each expert is the preferred strategy. This is illustrated by Fig. 11 and 12 where the size of the dots corresponds to the difference between the Q-values of each strategy at the mean position of each place cell: $|z_j^{explr} - z_j^{planning}|$. This corresponds to the selection probability in the gating network at each location. Larger differences indicate that the weights have differentiated more. The color/shape of each dot is determined by the strategy with the highest Q-values at that location (the winning strategy when following an optimal policy).

The planning strategy normally only needed one visit to the goal in a new environment to learn (i.e. to determine) good diffusion values. Fig. 11 shows the evolution of the Q-values of the gating network for an experiment planning strategy vs. exploration strategy. After 300 steps the global structure did not change significantly anymore. Close to the goal, exploration was often the preferred strategy, due to the coarseness of the planning strategy. Farther away, we see that the planning strategy was gaining terrain, because there the planning strategy performed well. The relative weights were still increasing (they diffused away from the goal due to the Q-learning algorithm), indicating that learning had not yet fully converged. In particular, the region at the lower left corner still needed to be visited more to learn the Q-values in this region.

Note that in Fig. 11 there is a region around (1.5,0.8) where the exploration strategy remained competitive with the planning strategy for a long time. This is again a consequence of the topological map (shown in Fig. 8(d)). Such regions differred between experiments (different maps) and often indicated regions in which the topological map contained a detour. In such a case it could indeed be a good strategy to transiently follow a random direction - as suggested by the *exploration* expert - to get onto another path where the planning strategy would be used again.

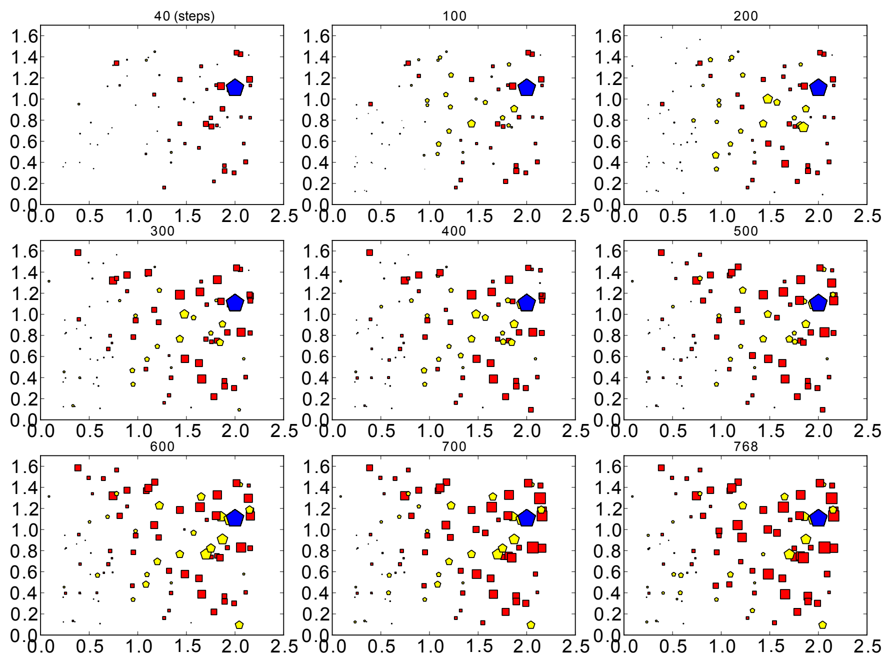


Figure 11. Evolution through time of the relative Q-values of the Planning expert (dark squares) vs. random strategy (light-colored pentagons) corresponding to Fig. 10(a). The relative Q-values $|z_j^{explr} - z_j^{planning}|$ are shown at the center of activation of each place cell. The dot takes the shape (square/pentagon) of the strategy with the highest Q-value (highest selection probability). The larger the dot, the more the weights have differentiated and the more likely the strategy with the highest Q-value is to be selected. The goal location is shown in blue. Competition between the strategies is still going on after 768 steps, but the structure becomes apparent. Near the goal the random exploration strategy often performs equally well as the topological map.

Fig. 12 shows the result of another experiment with the same system (i.e. planning and exploration experts) but with different distal landmarks with less aliasing of the place cells around the goal and after a longer time of experimentation. In this case, the gating network had stabilized, and the robot preferred the topological map strategy almost everywhere, except very close to the goal where the planning strategy was still unprecise and the robot's behavior was random. In the region near the goal the weights of the gating network had differentiated less, indicating that the strategies are still competitive in this region.

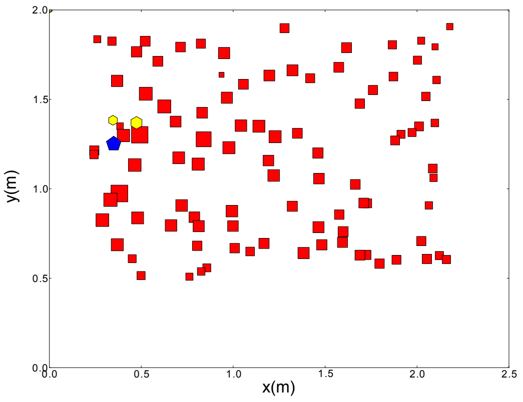


Figure 12. Planning expert vs. random strategy corresponding to Fig. 10(b). Colors as in Fig. 11. The planning expert is almost always preferred, except very near the goal. Note the lower weight differences very near the goal.

6.2. Taxon expert and exploration

We now verify the taxon strategy by having it compete with the exploration expert. The setup is the same as in the previous experiment. However because the taxon can only sense the goal within a certain range, the outcome we expected is different.

Once the taxon expert has learned to move towards the stimulus, its performance should be better than the exploration expert within a region around the goal. Farther away from the goal, the taxon does not receive sensory input and becomes equivalent to the exploration expert.

Fig. 13 shows the evolution of the relative Q-values through time. It is clear that the taxon expert had significantly been reinforced around the goal and a large region around it. However the taxon was still the dominant strategy relatively far away from the goal, which is against our expectations. We expected to see a more or less circular region around the goal in which the taxon was the dominant strategy, while the rest of the environment would be randomly assigned to one of both strategies, resulting in very small weights on the plot at these locations (we plot the weight difference). While we indeed see this phenomenon at many locations (predominantly at the lower right side of the environment), there are regions far away from the goal with non-negligible weights for the taxon. This effect stems from the aliasing in the place cells. If some of the place cells far away from the goal were activated even slightly when the robot was close to the goal, then the optimal strategy around the goal (the taxon) was also slightly reinforced at these aliased locations. This was only of importance when none of the strategies performed well at the aliased locations farther from the goal, because otherwhise this effect was dominated by reinforcements of these strategies. This indicates that the gating network gave a slight advantage to strategies having good performance around the goal when there was aliasing in the sensory input. One way to reduce this effect would be to only update the Q-values of the most active place cell in the gating network.

This would increase the learning time significantly and the system would not use the similarity between nearby locations anymore.

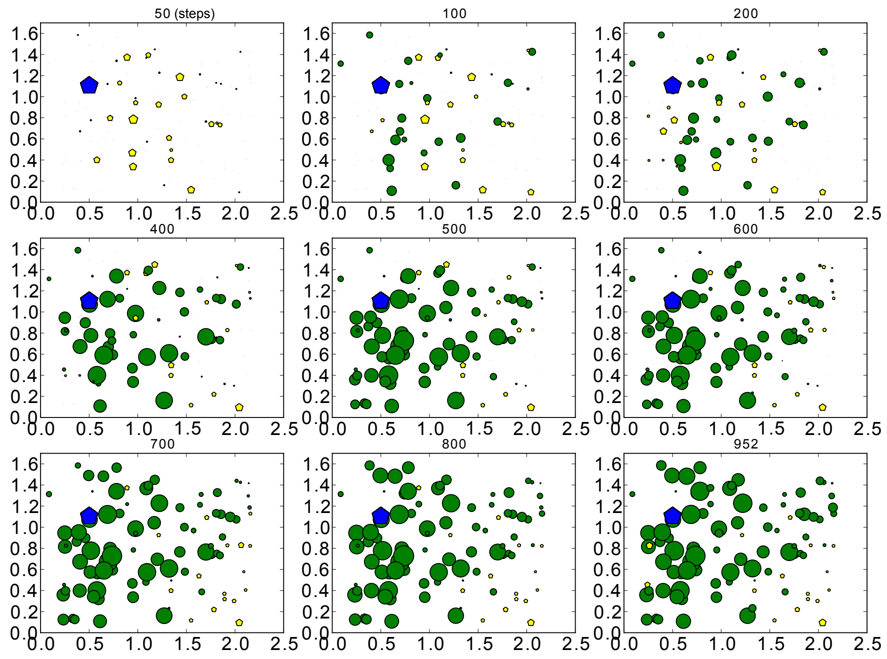


Figure 13. Taxon (dark circles) vs exploration expert (light pentagons). The goal is on the left (large dark pentagon). The gating network learned to select the taxon strategy near the goal, and preferred the random exploration strategy in some areas far away from the goal.

Despite such limitation, the system appropriately learned to priviledge the taxon expert near the goal and to select the exploration expert mostly in the right side of the maze, far away from the goal. Fig. 14 shows the global selection rate of both strategies as a function of the number of actions. Although small changes continued to occur - due to the equal performance of both strategies farther away from the goal - the learning had mostly converged.

Fig. 15 shows the learned output direction of the taxon Φ^T for each input direction (see Eq. 4). Here, we used a binary input vector r_j^{GP} to code the goal direction (the center of the Gaussian). The figure shows that the taxon learned to orientate towards the direction of the stimulus.

6.3. Planning expert and taxon (and exploration)

We then performed an experiment where we combined all experts and tested the ability of the gating network to select the right strategy in the right location. Based on the previous results, we expected the animat to learn to prefer the taxon strategy when close to the goal, while using the planning expert farther away. The exploration strategy

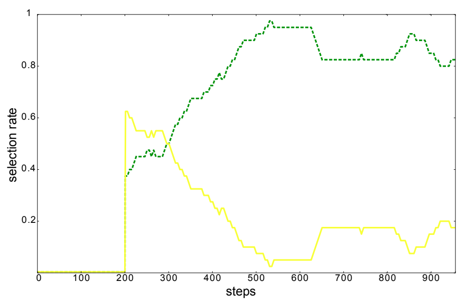


Figure 14. Selection rate of the taxon (dashed dark line) vs. exploration strategy (light line) computed over the last 200 actions. The gating network clearly learned to priviledge the taxon strategy.

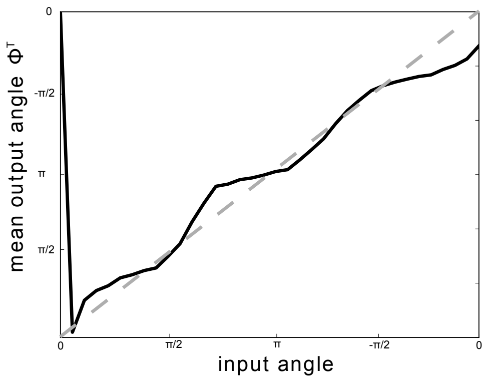


Figure 15. Output directions (weighted mean of angles) learned by the taxon in response to the input angle between the robot's head orientation and the goal. After learning, the taxon approximated a diagonal line and thus learned to orientate towards the direction of the stimulus.

should normally be used at the beginning of the experiments, and then progressively excluded in most location, unless if there is doubt or when there is indeed an advantage in following an arbitrary direction.

This experiment consisted of two parts. In the first part the goal location from the previous experiment was left unchanged and the robot started its exploration until converging to an appropriate coordination of strategies and adopting a satisfying behavioral policy to reach the goal. In the second part of the experiment, the beginning is similar to part 1, but the goal is moved to a new location at the opposite side of the environment after some time. This was done to evaluate the time required by the gating network to learn new associations of strategies with maze areas.

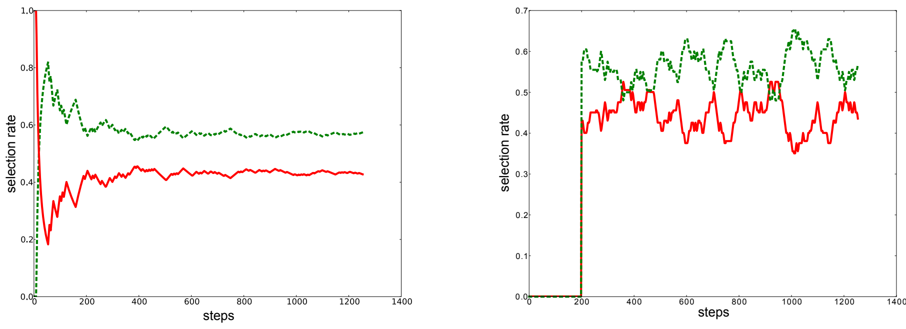
We performed two versions of this experiment. In the former we connected only the taxon and the topological map strategy to the gating network. Furthermore we pre-trained the taxon in a separate experiment (as in Section 6.2) to study the complementarity of the two strategies once stabilized before testing the whole system. In the latter we connected all three experts to the gating network. Each version used a different topological map, place cells and goal locations.

The experiments consisted of two phases. First we proceed as in the previous experiments. In the second phase the goal location was moved to test the context switching mechanism. We tested the second phase with and without the context switching enabled to illustrate its benefits.

6.3.1. Part 1: fixed goal We first discuss the results for the first version of the experiment: taxon and pre-trained taxon.

Fig. 17 shows the result of the experiment after 30 trials of learning - the robot had reached the goal 30 times. The gating network had learned to choose the taxon strategy around the goal, as indicated by large weight differences in this area. Farther away from the goal, the weight differences were smaller and the planning strategy was preferred most of the time.

Fig. 16 shows the selection rates of both strategies through time. We see that on short term, the selection rates varied (depending on the robot's trajectories) while the global selection rate had converged. Convergence was fast because each strategy performed well in a specific region and there was not much competition. Furthermore, the transitional phase was short because the taxon had been pre-trained (the taxon already had good performance around the goal.



(a) Selection rate computed over all previous actions

(b) Selection rate computed over the last 200 actions (moving average)

Figure 16. Evolution of the selection rate of both strategies (version 1) during learning (as a function of the number of actions). The full line represents the planning expert, the dashed line the (pre-trained) taxon strategy. The horizontal axis indicates the current step (time). The gating network globally converged to a stable repartition of selection of the two strategies, with the taxon being selected most often.

Fig. 18 illustrates the result of a session of the same experiment overlayed on an image of the environment to show the physical location corresponding to the different areas on Fig. 17.

For the second version of the experiment, we connected the taxon (this time not pre-trained), the planning and the exploration strategies to the gating network. Fig. 19 shows the evolution of the relative Q-values through time and Fig. 20 contains the selection rates for the different strategies. This gives (as in the previous experiments)

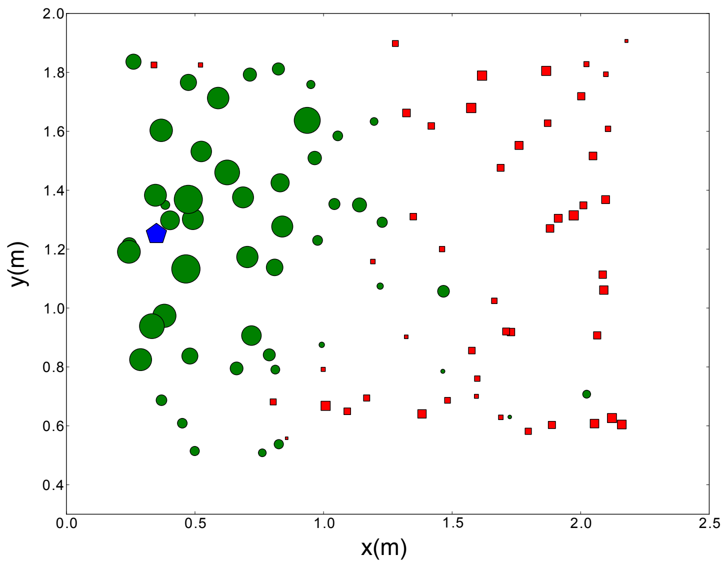


Figure 17. Planning expert (squares) vs. pre-trained taxon (circles) (version 1). The goal location is shown in as a dark pentagon on the left. The robot has learned to prefer the taxon strategy over the planning expert when close to the goal.

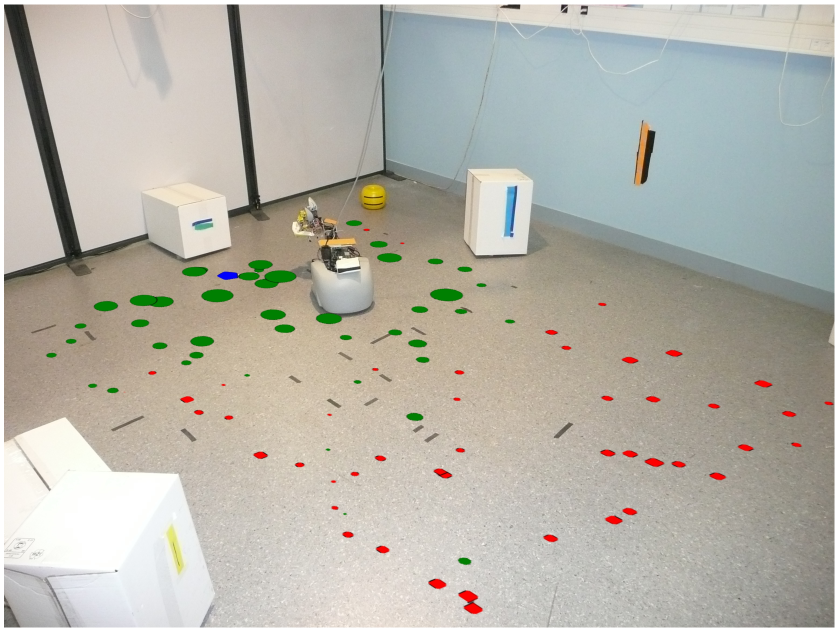


Figure 18. Result of the first version of the experiment illustrated in Fig. 17, this time overlayed on the environment.

an indication of how likely a strategy was to be selected at a certain location and of how much the weights had differentiated. To show such differentiation between the three competing strategies, we plotted $|z_j^1 - (z_j^2 - z_j^3)\frac{1}{2}|$, where the indices $\{1, 2, 3\}$ correspond to the strategies ordered by Q-value for place cell j by descending order. In other words, we plotted the relative Q-value of the most likely strategy at each location compared to the average of the other strategies.

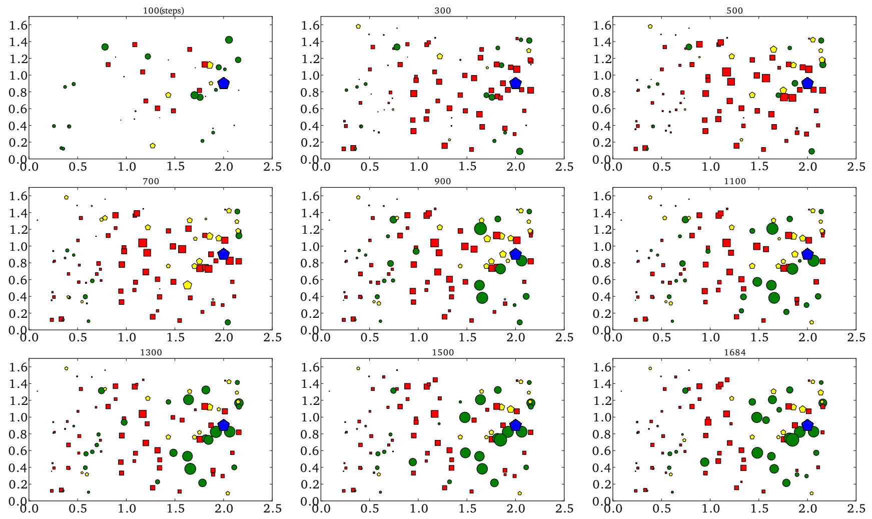


Figure 19. Evolution of the relative Q-values (see text) for the second version of the experiment with taxon, topological map and exploration experts. The taxon strategy is preferred around the goal, while the planning strategy has higher Q-values farther from the goal (as in the simplified version). The exploration expert only wins in a small area (see text for explanation). Taxon: dark circles, topological map: dark squares, exploration: light pentagons and goal: dark pentagon on the right. Axes are in meters.

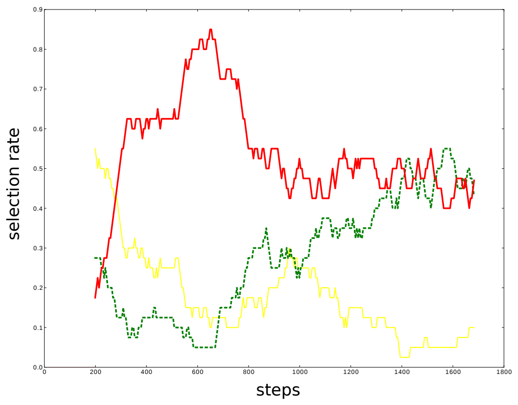


Figure 20. Selection rates computed over the past 200 actions (moving average) for all three experts. The taxon is shown as a dashed dark line, the topological map as a dark full line and the exploration expert as a light line. After an initial learning period, the taxon increases in importance (approx. 700 actions) and finally settles in a similar regime as in Fig. 16(b). The exploration expert has a very low selection rate as expected.

The figure shows that between 700 and 900 actions, the taxon took over the region around the goal, as expected. This was slightly longer than in the taxon-exploration experiment because the taxon needed to learn to move towards the stimulus and the gating network needed to learn the best out of three strategies. The topological map seemed to have performed very well in this experiment as even in a small region around the goal, this strategy was the winning strategy, although we expected the taxon strategy to win here. This effect is shown in detail in Fig. 21 which shows the relative Q-values at the end of the experiment for each expert separately (at the location where each expert was the most reinforced one). Clearly, the same structure as in Fig. 17 is found (without the exploration).

The exploration strategy's selection rate droped to a significantly lower level than the two other strategies. The taxon and topological map strategies' rates approached their final state after 1400 actions. They continued to oscillate (depending on the robot's trajectory), similar to Fig. 16(b), but remained globally stable.

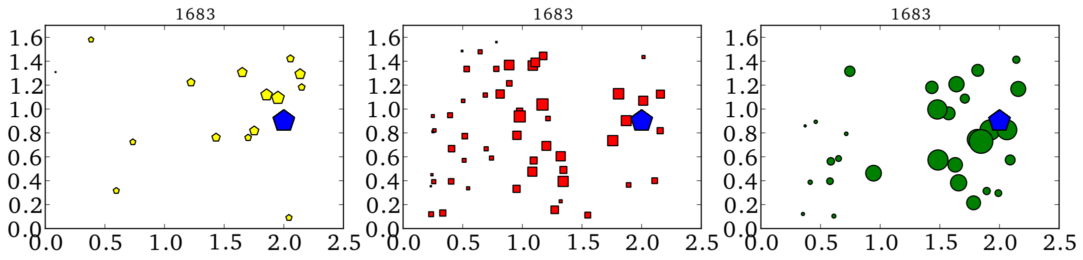


Figure 21. Left: exploration, center: topological map, right: taxon. Relative Q-values for each expert at the end of the experiment at locations where each strategy has the highest Q-value (highest selection probability). Axes are in meters.

Now we still have to explain why the taxon performed badly in this small region mainly above the goal. Supposing that the gating network indeed worked correctly, this could only be due to the fact that the taxon indeed performed worse than the two competing strategies in this region. Now if we look at the learned directions of the taxon (Fig. 22), we indeed find that the taxon had not correctly learned the directions when the goal was located in the south ($-\pi/2$) of the robot.

Fig. 23 shows the visited locations of the robot. It is clear that the robot had approached the goal (located at (2,0.9)) from the north very few times. Hence the taxon did not have the possibility to learn to move towards the goal from above. The gating network learned that it was better to follow a random strategy (exploration or the planning when very near the goal) here, which was the best available choice.

Finally, there was a region around (0.5,0.5) where the topological map was not the preferred strategy in all cases. Fig. 24 shows the topological map at the end of the experiment (step 1683). At the right of this region there was a large gap in the map (two disjoint trajectories). There were multiple nearby locations which pointed towards almost opposite directions (because the map tried to find a path with the least number of nodes). So if there was some aliasing in the place cells in this region, the

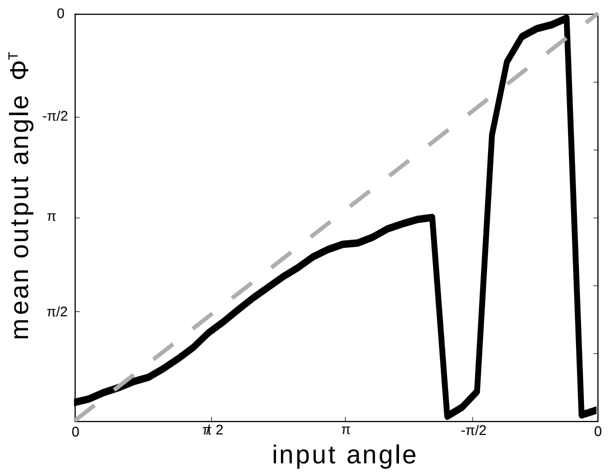


Figure 22. Learned orientations of the taxon. Each time the goal was located at the south of the robot ($-\pi/2$), the taxon did not learn to move towards the stimulus.

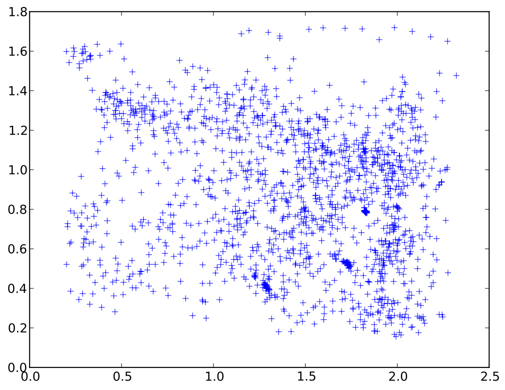


Figure 23. Visited locations of the robot. The goal was located at (2.0,0.9). The region right above the goal (the goal extends up to (2.0,1.0) had been visited much less by the robot, which explains why the taxon had not learned the correct response when the goal was situated at the south of the robot. Axes are in meters.

topological map could oscillate here and thus have low performance, as we explained for the experiment with the topological map and exploration expert connected to the gating network. Fig. 21 shows that the gating network learned to also select the two other strategies in this region to compensate the limitations of the planning strategy in this zone.

Globally, we got very similar results for both versions of the experiments. Once the robot had experienced the goal, the exploration expert's selection rate progressively decreased to a very low level. The main conclusion is that the gating network indeed worked as proposed and could easily learn to use the topological map when farther away from the goal and the taxon when closer. Interestingly, the system also solved the more complex situations in which aliasing of the map caused locally bad performance.

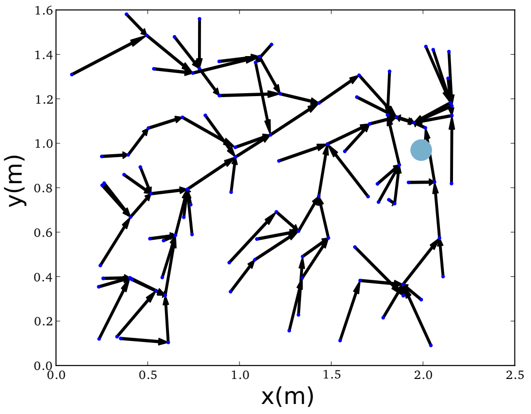


Figure 24. Topological map after 1683 steps. The goal is shown as a circle on the right.

6.3.2. Part 2: changing goal location In the second part of the experiment, after 1000 steps the goal was moved to the opposite side of the environment to test the ability of the system to adapt to a new situation. The context switching mechanism was disabled.

The experiment was relatively hard as the only visible change to the gating network is the reward. The result from a session (first version) is shown in Fig. 25. The gating network appropriately learned to stop selecting the taxon expert on the left side of the maze and to rather select it on the right side of the maze (near the new goal location). Thus the robot adapted to the new situation. However, this process was very slow and took approximately 8000 steps (approx. 180 trials). Even then the taxon remained the preferred strategy in the area around the old goal. This is because the robot needed to unlearn the previous associations between strategies and maze areas before learning the new ones.

Rats typically notice drastic changes in the environment and can adapt their behavior fast when they are not overtrained - in which case they build unflexible habits [17]. In our own previous strategy shifting experiments with real rats, in response to task changes, animals abandoned the previously performed strategy after an average of 10 error trials and learned to select the new appropriate strategy in about 100 trials [8, 66]. Thus the gating network alone is not sufficient to produce behavioral performance and adaptatibility comparable to real rats.

6.4. Context switching

To overcome this limitation (inherent to the Q-learning algorithm), we implemented a simple context switching mechanism which associates to each different goal location a different context and thus a different instance of the gating network (see Section 4.3). The idea was to anchor the detection of new contexts in response to a change in goal location on the diffusion values in the topological map of the planning expert. While the place cells activation (i.e. the input of the gating network) does not change

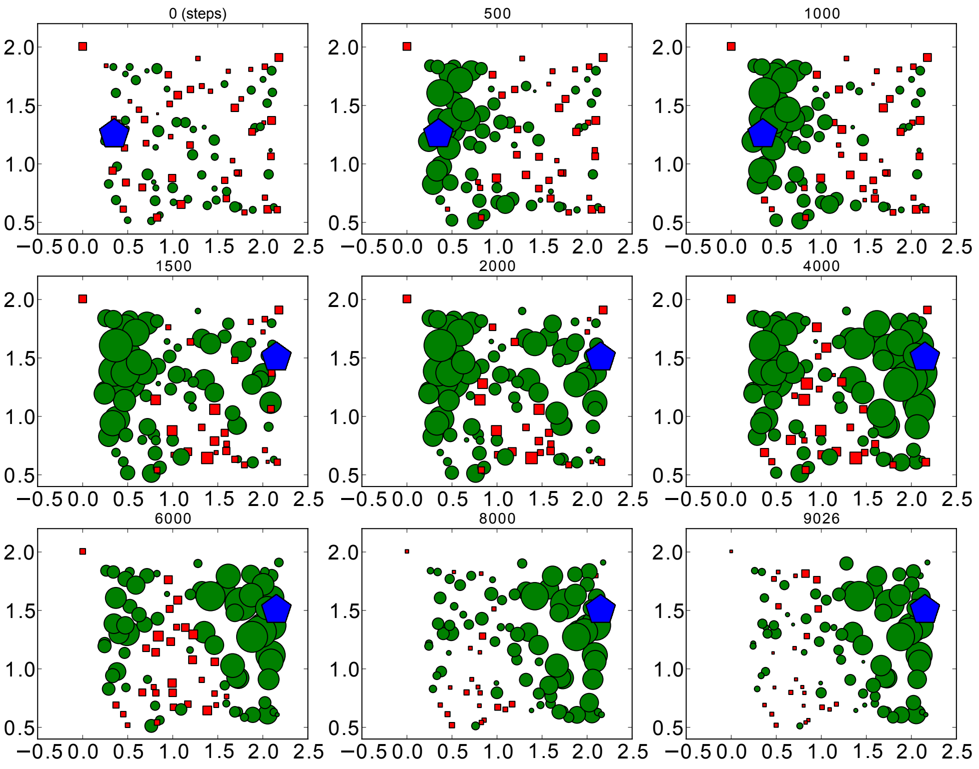


Figure 25. Planning expert (squares) vs. taxon (circles) (first version of the experiment). The goal location is shown in blue. The location of the goal is changed after 1000 steps. The gating network appropriately learned to stop selecting the taxon expert on the left side of the maze and to rather select it on the right side of the maze (near the new goal location). However, this process was very slow and took approximately 8000 steps, corresponding to around 180 trials. Axes in meters.

when the goal is moved, the goal values g_j in the topological map do. However the goal values can fluctuate heavily and it is better to use the diffusion values d_j. The topological structure was thus used to compute the diffusion values, which gave them a much smoother activation.

As in the Part 1 of the experiment, we first discuss the result of the first version (no exploration expert).

The previous experiment was repeated with this mechanism (Fig. 26). In total 4 contexts were recruited (two transitional). The goal was moved twice, to verify that the robot had stored the initial context and that it could appropriately restore it. After 900 steps, the goal was moved from the initial location on the left to the new location on the right. Once the robot found the new goal location, the topological map automatically produced different diffusion values and the system could create a new context (illustrated by the abrupt vanishing of most circles and squares at step #1100 on Fig. 26). After 200 more steps, the gating network had learned to select the taxon expert near the new goal location and the planning expert in most of the rest of the maze. At step #1450 the goal was moved back to the initial location. Because the contexts had been stored, the robot recalled the previously learned weights when the goal moved back to its initial location. This resulted in instantaneously restoring previously learned weights of associations of the gating network (illustrated at step #1770 on Fig. 26).

In total the robot had constructed four contexts. The first one corresponded to the initial phase in which the robot had yet to find the goal for the first time. The second one was the context used to learn when the goal was on the left and when the goal was moved back to the left (at the end of the experiment). The third one was a transitional context used when the goal had only recently changed sides. This is probably due to the change in diffusion values of the topological map once the robot did not get reward anymore at the initial goal location, although the robot had not experienced the new goal location yet. The last one was used when the goal was found on the right side of the maze.

With this simple extension, learning was much faster and the robot could recall previous contexts which made the navigation system much more useful in practice. Another small but useful extension would be to associate a ζ^i with each context. This way the robot could approach the winner-take-all strategy for contexts that had been learned for a longer period.

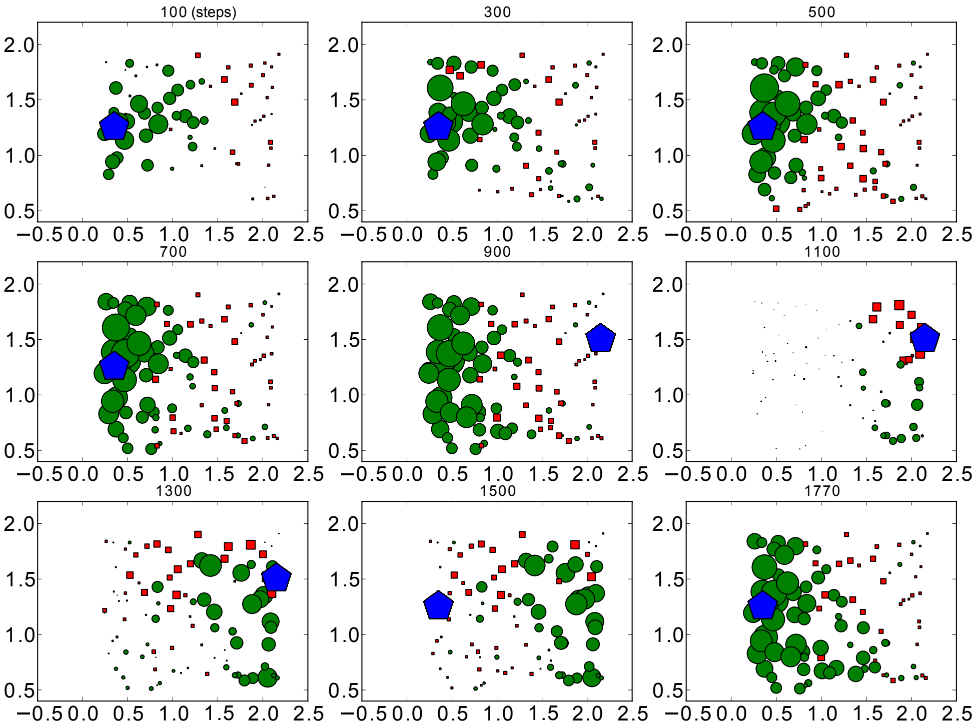


Figure 26. Planning expert (squares) vs. taxon (circles) with the context switching mechanism. The goal location is shown in blue. Learning is now faster as the robot recalls previously learned contexts and new ones. The number of steps is shown at the top. With the context switching meta-controller, the robot could quickly adapt its behavioral policy to a change in goal location, and could fastly restore its initial policy when the goal was moved back to its first location. Axes in meters.

We repeated this experiment (moving goal with context switching) with the second version (all three experts). As the gating network learns more slowly when three strategies are present, we moved the goal after 1683 steps (we continued the experiment from Part 1).

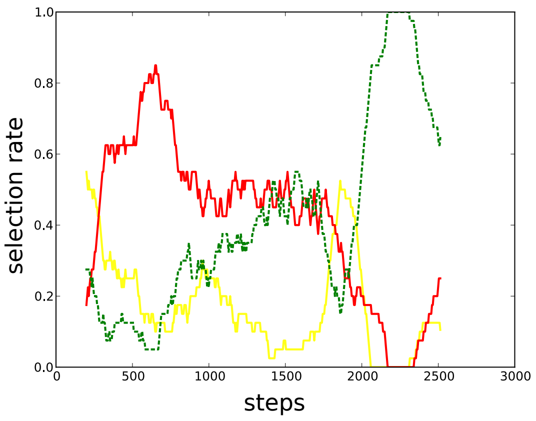


Figure 27. Selection rates computed over the past 200 actions for all three experts. The taxon is shown as a dashed dark line, the topological map as a dark full line and the exploration expert as a light line. The goal was moved after 1683 steps and shortly afterwards we see a new transient phase in the selection rates because a new context was created.

Fig. 27 shows the selection rate of the three strategies. Because the context switching mechanism recrutes a new set of Q-values when the change is detected, learning in the gating network can restart from scratch, which is faster than when the same context is used.

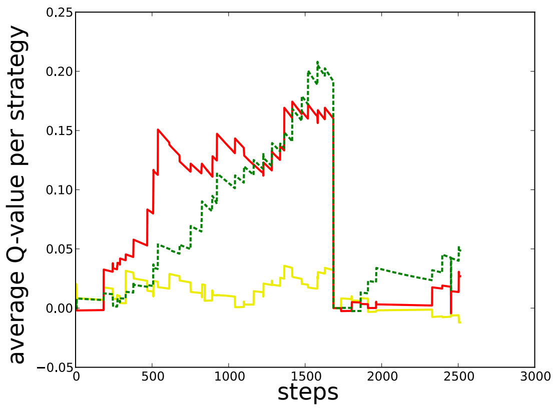


Figure 28. Average Q-values (over all locations) of the current context for each of the strategies as a function of the number of actions. Shortly after the goal is moved, the weights become zero, because a new context is activated. Learning restarts in this new context. Colors as in Fig. 27.

In Fig. 28 the average Q-values for each expert are shown (averaged over all locations). When the context switch occurs, the weights become zero (new context). Finally, Fig. 29 shows the selected context at each step. In total four contexts were created, of which two transitional. The context selection mechanism is stable (this depends on the threshold).

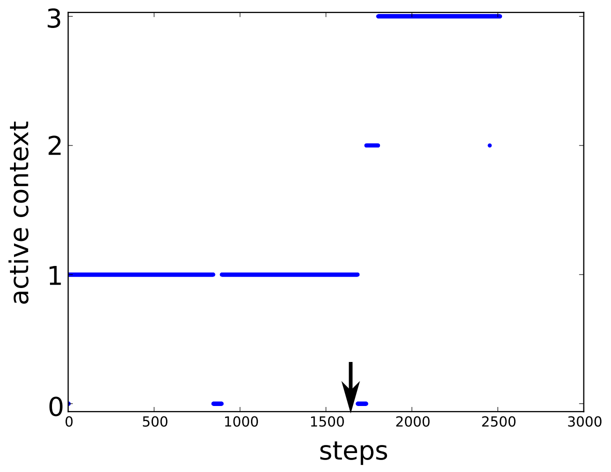


Figure 29. Active context through time. When the goal changes location (indicated by the arrow) a transitional period starts and until the diffusion values have adapted to the new goal location (context 3).

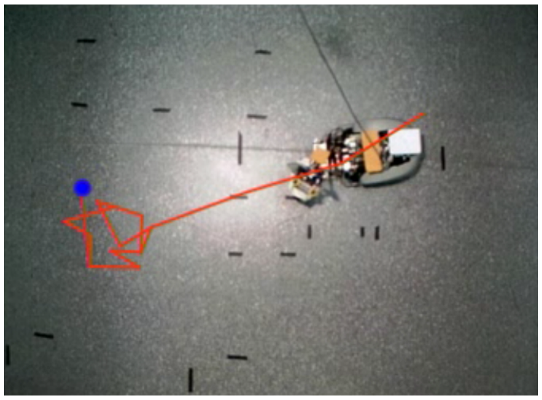
6.5. Cooperation: typical trajectories

To prove that the robot could learn to make strategies cooperate, we compared a number of trajectories of the robot equipped with only the planning strategy or with both the planning strategy and the taxon strategy. The goal was not moved during the experiment.

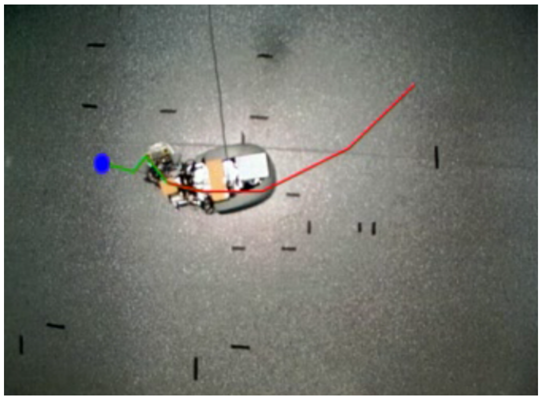
To illustrate abrupt switching between strategies when both strategies were used, the gating network was used with ζ initially set to 1 and to ∞ after learning (the optimal strategy always wins).

Fig. 30(a) and 30(b) show two typical paths taken by the robot to reach the goal. Fig. 30(a) is a trajectory obtained when only the planning strategy was used. Due to the limited precision of the topological map (place cells), the robot took pseudorandom actions near the goal. It could not plan a path leading closer to the goal. Fig. 30(b) shows the cooperation between the taxon and the planning strategy. The robot had learned that when it was near the goal, it should rely upon the taxon strategy as it could guide the robot when the goal was close enough‖. When only the taxon strategy was used (not shown) and the robot was placed far away from the goal, the robot did not approach the goal and instead moved randomly.

‖ A video illustrating such coordination of navigation strategies in the robot in a simple example is shown here: http://elis.ugent.be/˜kcaluwae/VideoCaluwaerts2010.mov



(a) Planning strategy only. The robot approaches the goal fast, but cannot reach it efficiently due to the limited precision of its topological map.



(b) Planning and taxon strategies. The robot approaches the goal with the planning strategy as before and switches to the taxon strategy at the end.

Figure 30. Sample trajectories of the robot: (a) only planning strategy enabled, (b) planning and taxon strategies enabled

7. Discussion

We presented the implementation of a novel strategy selection meta-controller allowing an autonomous robot to navigate an initially unknown environment. The model selected among two parallelly learned navigation strategies: a response strategy learning directions of movements in response to perceived cues; a place strategy building a map of place cells and planning trajectories between different maze areas. The strategy selection meta-controller was added to a previously published model of multiple navigation strategies [33] which was tested in simulation to replicate a series of rat behavioral experimental results [34, 18].

It was shown that the animat can learn to ignore useless strategies very fast (e.g. an exploration expert after the goal was found). Furthermore, the robot learned to associate states with optimal strategies even in more complex cases, when for example a local taxon strategy was combined with a global but coarse path planning strategy. By introducing a simple context switching mechanism, the robot could adapt quickly to changes in the environment.

The results are encouraging in two ways. First, the simulation model was adapted and validated on a real robotic platform, which was the main goal of this work. For the model to work on a robotic platform, the sensor input and model parameters had to be adapted. Because of these differences, the results presented here cannot be compared directly with the simulation model in the quantitative sense. However the general principle of *common currency* was shown to be flexible and allows us to investigate the behavior of our robotic rat with different combinations of strategies in a realistic setting.

Secondly, the robotic experiments revealed that a simple context switching mechanism can drastically increase the performance. Such a mechanism was absent from the simulation model as the robot could learn different situations with a single gating network because more sensory inputs about perfectly distinguishable landmark cues were avalaible. Hence our context switching shows that good performance can be obtained with less sensory input (only place cells feed into the gating network) and it has the added benefit of long-term storage of situations.

In terms of neural substrates, such meta-controller may constitute a model of rat prefrontal functions during strategy-shifting. Indeed, although less differentiated than primates' prefrontal cortex (PFC), the rat PFC is known to have strong functional homologies with the lateral PFC in primates [67]. It is important to achieve high-level cognitive processes, usually referred to as executive functions, that is "complex cognitive processes required to perform flexible and voluntary goal-directed behaviors based on stored information in accordance with the context" [68]. Responses of rat PFC neurons to working-memory components [69], spatial goals [70] and action-outcome contingencies [71, 72] initially suggested to several authors that the rat PFC could be the neural substrate for a particular behavioral strategy, the planning system, or more generally for model-based learning processes - that is decision-making based on the learning of transitions within the environment by means of action-outcome contingencies [50, 48, 49], see [8] for a review. However, lesions of the rat PFC only impair the acquisition of goal-directed behaviors - that is, model-based strategies such as planning [73] -, but not their expression [74]. Besides, lesions of the rat PFC impair working-memory processes only when combined with other factors such as the difficulty of the task, attentional mechanisms or the requirement for flexible behaviors [68]. This suggests that the neural substrate for the planning navigation strategy is located elsewhere in the brain, and that the rat PFC might be involved in a higher-level of decision-making and adaptation [8]. Consistent with our interpretation, on the one hand, neural correlates of forward planning have been found in the hippocampal system [75] and projections from the hippocampal system to the ventral and dorsomedial striatum appear important for model-based learning such as the quick adaptation to changes in the association between places and rewards [23, 76, 8, 77, 78]. On the other hand, PFC was found to be crucial for switching between behavioral strategies in response to task-rule changes [35, 36, 37, 79, 80]. Neural responses of the rat PFC show abrupt changes when the animal switches its navigation strategy [81, 82, 66], and only neurons responding for the correct strategy - i.e. the rewarded one - are reactivated during sleep in interaction with the hippocampus, therefore contributing to the consolidation of the association of the right strategy with the right context [66]. Our present meta-controller - combination of a gating network that learns to associate strategies to subparts of the task and a context-switching detector - constitutes a proposition of how such strategy shifting functions may be modelled. We found that, during robotics tests in the real world as opposed to previous simulations of the model, such system was required on top of the planning and taxon strategies to produce fast adaptation to task changes. Such meta-controller may correspond to a minimal form of cognitive control models, which are used to model primates PFC's role in memorizing which tasksets are relevant in which contexts [65]. In future work, we plan to test the robot for artificial lesions of this meta-controller, as compared to lesions of the planning system only, to compare with rodents behavioral data previously obtained during various maze tasks commonly used in the neuroscience community.

Finally, this work has also the potential of contributing to mobile robotics. Indeed, the bio-inspired ability to rapidly switch between several behavioral strategies, and to memorize which strategy is the most efficient and appropriate in each subzone of the environment could help improve current control architectures for robots. Multi-layered control architectures with different levels of decisions have become more and more popular in robotics and are now widely used [83, 84, 85]. Such architectures open issues such as managing the interactions between submodules, coordinating multiple competing learning processes and providing alternative solutions to motion planning in situations where such strategy is limited [86, 87]. Indeed the planning strategy can be approximative when coping with uncertainties, e.g. when there is perceptual aliasing as we have seen here, and can also require high computational costs and long time to propagate possible trajectories through mental maps [84]. In contrast, in situations where animals have developed habits under the form of cue-guided taxon or response strategies to solve a particular task, they can perform quick and accurate decision-making. Moreover, in the case of a sudden change in the environment, they can adaptively abandon habits in favor of planning new routes towards their current goal [88]. Taking inspiration from computational models of how the mammalian brain learns to select appropriate strategies and to switch between strategies as a function of a speed-accuracy trade-off may constitute the basis of great future advances in robotics [73, 89].

Acknowledgments

This research was funded by the EC FP6 IST 027819 ICEA Project and a Ph. D. fellowship of the Research Foundation - Flanders (FWO).

References

[1] J.-A. Meyer, A. Guillot, B. Girard, M. Khamassi, P. Pirim, and A. Berthoz. The Psikharpax project: towards building an artificial rat. *Robotics and Autonomous Systems*, 50(4):211–223, 2005.

[2] S. N'Guyen, P. Pirim, and J.-A. Meyer. Tactile texture discrimination in the robot-rat Psikharpax. In *BIOSIGNALS 2010, Third International Conference on Bio-Inspired Systems and Signal Processing*, Valencia, Spain, 2010.

[3] M. Bernard, S. N'Guyen, P. Pirim, B. Gas, and J.-A Meyer. Phonotaxis behavior in the artificial rat Psikharpax. In *International Symposium on Robotics and Intelligent Sensors, IRIS2010*, Nagoya, Japon, 2010.

[4] S. N'Guyen. *Mise au point du système vibrissal du robot-rat Psikharpax et contribution à la fusion de ses capacités visuelle, auditive et tactile.* PhD thesis, Université Pierre et Marie Curie, 2010.

[5] S. N'Guyen, P. Pirim, J.-A. Meyer, and B. Girard. An integrated neuromimetic model of the saccadic eye movements for the psikharpax robot. In J.-A. Meyer, A. Guillot, and J. Hallam, editors, *From Animals to Animats*, LNAI. Springer, 2010.

[6] O. Trullier, S. Wiener, A. Berthoz, and J.-A. Meyer. Biologically-based artificial navigation systems: Review and prospects. *Progress in Neurobiology*, 51:483–544, 1997.

[7] A.D. Redish. *Beyond the Cognitive Map: From Place Cells to Episodic Memory.* MIT Press, Cambridge, 1999.

[8] M. Khamassi. *Complementary roles of the rat prefrontal cortex and striatum in reward-based learning and shifting navigation strategies.* PhD thesis, Université Pierre et Marie Curie, 2007.

[9] A. Arleo and L. Rondi-Reig. Multimodal sensory integration and concurrent navigation strategies for spatial cognition in real and artificial organisms. *Journal of Integrative Neuroscience*, 6(3):327–366, 2007.

[10] R. Morris. Developments of a water-maze procedure for studying spatial learning in the rat. *Journal of Neuroscience Methods*, 11(1):47–60, 1984.

[11] M. G. Packard, R. Hirsh, and N. M. White. Differential effects of fornix and caudate nucleus lesions on two radial maze tasks: evidence for multiple memory systems. *Journal of Neuroscience*, 9(5):1465–1472, 1989.

[12] N. Burgess. Spatial cognition and the brain. *Year In Cognitive Neuroscience 2008*, 1124:77–97, 2008.

[13] G. E. Alexander, M. R. DeLong, and P. L. Strick. Parallel organization of functionally segregated circuits linking basal ganglia and cortex. *Annual Review of Neuroscience*, 9:357–381, 1986.

[14] J. W. Mink. The basal ganglia: Focused selection and inhibition of competing motor programs. *Progress in Neurobiology*, 50(4):381–425, 1996.

[15] J. C. Houk, J. L. Adams, and A. G. Barto. A model of how the basal ganglia generate and use neural signals that predict reinforcement. In J. C. Houk, J. L. Davis, and D. G. Beiser, editors, *Models of Information Processing in the Basal Ganglia*, pages 249–271. The MIT Press, Cambridge, MA, 1995.

[16] A.M. Graybiel. The basal ganglia and chunking of action repertoires. *Neurobiology of Learning and Memory*, 70(1-2):119–36, 1998.

[17] H.H. Yin and B.J. Knowlton. The role of the basal ganglia in habit formation. *Nature Reviews Neuroscience*, 7(6):464–76, 2006.

[18] B.D. Devan and N.M. White. Parallel information processing in the dorsal striatum: Relation to hippocampal function. *Journal of Neuroscience*, 19(7):2789–2798, 1999.

[19] M. G. Packard and B. J. Knowlton. Learning and memory functions of the basal ganglia. *Annual Review of Neuroscience*, 25:563–593, 2002.

[20] Richard G. M. Morris. Spatial localization does not require the presence of local cues. *Learning and Motivation*, 12(2):239–260, 5 1981.

[21] J. O'Keefe and L. Nadel. *The Hippocampus as a Cognitive Map.* Clarendon Press, Oxford, 1978.

[22] H.H. Yin and B.J. Knowlton. Contributions of striatal subregions to place and response learning. *Learning & Memory*, 11(4):459–63, 2004.

[23] S.V. Albertin, A.B. Mulder, E. Tabuchi, M.B. Zugaro, and S.I. Wiener. Lesions of the medial shell of the nucleus accumbens impair rats in finding larger rewards, but spare reward-seeking behavior. *Behavioral Brain Research*, 117(1-2):173–183, 2000.

[24] R. Pfeifer, M. Lungarella, and F. Iida. Self-organization, embodiment, and biologically inspired robotics. *Science*, 318:1088–1093, 2007.

[25] M. Arbib, G. Metta, and P. van der Smagt. *Handbook of robotics*, chapter Neurorobotics: From vision to action, pages 1453–1480. Springer-Verlag, Berlin, 2008.

[26] J.-A. Meyer and A. Guillot. *Handbook of robotics*, chapter Biologically-inspired robots, pages 1395–1422. Springer-Verlag, Berlin, 2008.

[27] A. Arleo and W. Gerstner. Spatial cognition and neuro-mimetic navigation: a model of hippocampal place cell activity. *Biological Cybernetics*, 83(3):287–299, 2000.

[28] J.L. Krichmar, A.K. Seth, D.A. Nitz, J.G. Fleischer, and G.M. Edelman. Spatial navigation and causal analysis in a brain-based device modeling corticalhippocampal interactions. *Neuroinformatics*, 3(3):147–169, 2005.

[29] J.G. Fleischer, J.A. Gally, G.M. Edelman, and J.L. Krichmar. Retrospective and prospective responses arising in a modeled hippocampus during maze navigation by a brain-based device. *Proceedings of the National Academy of Science*, 104(9):3556–3561, 2007.

[30] A. Barrera and A. Weitzenfeld. Biologically-inspired robot spatial cognition based on rat neurophysiological studies. *Autonomous Robots*, 25:147–169, 2008.

[31] C. Giovannangeli and P. Gaussier. Autonomous vision-based navigation: Goal-oriented action planning by transient states prediction, cognitive map building, and sensory-motor learning. In *Proceedings of the International Conference on Intelligent Robots and Systems*, volume 1, pages 281–297. University of California Press, 2008.

[32] M. Milford and G. Wyeth. Persistent navigation and mapping using a biologically inspired slam system. *The International Journal of Robotics Research*, 29(9):1131–1153, 2010.

[33] L. Dollé, D. Sheynikhovich, B. Girard, R. Chavarriaga, and A. Guillot. Path planning versus cue responding: a bioinspired model of switching between navigation strategies. *Biological Cybernetics*, 103(4):299–317, 2010.

[34] J.M. Pearce, A.D. Roberts, and M. Good. Hippocampal lesions disrupt navigation based on cognitive maps but not heading vectors. *Nature*, 396(6706):75–77, 1998.

[35] M.E. Ragozzino, S. Detrick, and R.P. Kesner. Involvement of the prelimbic-infralimbic areas of the rodent prefrontal cortex in behavioral flexibility for place and response learning. *Journal of Neuroscience*, 19(11):4585–94, 1999.

[36] J.M. Birrell and V.J. Brown. Medial frontal cortex mediates perceptual attentional set shifting in the rat. *Journal of Neuroscience*, 20(11):4320–4324, 2000.

[37] S. Killcross and E. Coutureau. Coordination of actions and habits in the medial prefrontal cortex of rats. *Cerebral Cortex*, 13(4):400–8, 2003.

[38] B. Ujfalussy, P. Erős, Z. Somogyvári, and T. Kiss. Episodes in space: A modeling study of hippocampal place representation. In *From Animals to Animats*, pages 123–136, 2008.

[39] M.T. Block. A note on refraction and image formation of rats eye. *Vision Research*, 9(6):705–711, 1969.

[40] A. J. Parker. Binocular depth perception and the cerebral cortex. *Nature Reviews Neuroscience*, 8(5):379–391, 2007.

[41] L. Dollé, D. Sheynikhovich, B. Girard, B. Ujfalussy, R. Chavariagga, and A. Guillot. Analyzing interactions between cue-guided and place-based navigation with a computational model of action selection: Influence of sensory cues and training. In *From Animals to Animats*, LNAI. Springer, 2010.

[42] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

[43] T. Kohonen, M. R. Schroeder, and T. S. Huang, editors. *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

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

[45] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, October 2007.

[46] B. Schölkopf, A. Smola, and K.-R. Muller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319, July 1998.

[47] K.I. Kim, M. O. Franz, and B. Schölkopf. Kernel hebbian algorithm for iterative kernel principal component analysis. Technical report, Max Planck Institute for Biological Cybernetics, 2003.

[48] M. E. Hasselmo. A model of prefrontal cortical mechanisms for goal-directed behavior. *Journal of Cognitive Neuroscience*, 17(7):1115–1129, 2005.

[49] L.-E. Martinet, D. Sheynikhovich, K. Benchenane, and A. Arleo. Spatial learning and action planning in a prefrontal cortical network model. *PLoS Computational Biology*, 7(5):e1002045,

2011.

[50] J.P. Banquet, P. Gaussier, M. Quoy, A. Revel, and Y. Burnod. A hierarchy of associations in hippocampo-cortical systems: cognitive maps and navigation strategies. *Neural Computation*, 17(6):1339–1384, 2005.

[51] N. Cuperlier, M. Quoy, and P. Gaussier. Neurobiologically inspired mobile robot navigation and planning. *Frontiers in Neurorobotics*, 2007.

[52] R. W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962.

[53] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[54] M. Khamassi, L.E. Martinet, and A. Guillot. Combining self-organizing maps with mixture of epxerts: Application to an Actor-Critic model of reinforcement learning in the basal ganglia. In *From Animals to Animats*, pages 394–405, Rome, Italy, 2006. Springer-Verlag.

[55] M. Packard and J. McGaugh. Inactivation of hippocampus or caudate nucleus with lidocaine differentially affects the expression of place and response learning. *Neurobiology of Learning and Memory*, 65(1):65–72, 1996.

[56] E. Save and B. Poucet. Hippocampal-parietal cortical interactions in spatial cognition. *Hippocampus*, 10(4):491–499, 2000.

[57] D. Eilam and I. Golani. Home base behavior of rats (rattus norvegicus) exploring a novel environment. *Behavioural Brain Research*, 34(3):199–211, 1989.

[58] K. Doya. Complementary roles of basal ganglia and cerebellum in learning and motor control. *Current Opinion in Neurobiology*, 10(6):732–739, 2000.

[59] K. Samejima, Y. Ueda, K. Doya, and M. Kimura. Representation of action-specific reward values in the striatum. *Science*, 310:1337–1340, 2005.

[60] Matthew R. Roesch, Donna J. Calu, and Geoffrey Schoenbaum. Dopamine neurons encode the better option in rats deciding between differently delayed or sized rewards. *Nature Neuroscience*, 10(12):1615–1624, 2007.

[61] M. Matsumoto, K. Matsumoto, H. Abe, and K. Tanaka. Medial prefrontal cell activity signaling prediction errors of action values. *Nature Neuroscience*, 10:647–656, 2007.

[62] R. Chavarriaga, T. Strösslin, D. Sheynikhovich, and W. Gerstner. A computational model of parallel navigation systems in rodents. *Neuroinformatics*, 3(3):223–241, 2005.

[63] N.D. Daw, J.P. O'Doherty, P. Dayan, B. Seymour, and R.J. Dolan. Cortical substrates for exploratory decisions in humans. *Nature*, 441(7095):876–879, 2006.

[64] C. J. C. H. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[65] E.K. Miller and J.D. Cohen. An integrative theory of prefrontal cortex function. *Annual Review of Neuroscience*, 24:167–202, 2001.

[66] A. Peyrache, M. Khamassi, K. Benchenane, S.I. Wiener, and F.P. Battaglia. Replay of rule-learning related neural patterns in the prefrontal cortex during sleep. *Nature Neuroscience*, 12(7):919–926, 2009.

[67] H.B. Uylings, H.J. Groenewegen, and B. Kolb. Do rats have a prefrontal cortex? *Behavioral Brain Research*, 146(1-2):3–17, 2003.

[68] S. Granon and B. Poucet. Involvment of the rat prefrontal cortex in cognitive functions: A central role for the prelimbic area. *Psychobiology*, 28(2):229–237, 2000.

[69] E.H. Baeg, Y.B. Kim, K. Huh, I. Mook-Jung, H.T. Kim, and M.W. Jung. Dynamics of population code for working memory in the prefrontal cortex. *Neuron*, 40(1):177–188, 2003.

[70] V. Hok, E. Save, P.P. Lenck-Santini, and B. Poucet. Coding for spatial goals in the prelimbic/infralimbic area of the rat frontal cortex. *Proceedings of the National Academy of Sciences of the U.S.A.*, 102(12):4602–4607, 2005.

[71] A.B. Mulder, R.E. Nordquist, O. Orgut, and C.M. Pennartz. Learning-related changes in response patterns of prefrontal neurons during instrumental conditioning. *Behavioral Brain Research*, 146(1-2):77–88, 2003.

[72] W.J. Kargo, B. Szatmary, and D.A. Nitz. Adaptation of prefrontal cortical firing patterns and their

fidelity to changes in action-reward contingencies. *Journal of Neuroscience*, 27(13):3548–3559, 2007.

[73] N.D. Daw, Y. Niv, and P. Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8(12):1704–1711, 2005.

[74] S.B. Ostlund and B.W. Balleine. Lesions of medial prefrontal cortex disrupt the acquisition but not the expression of goal-directed learning. *Journal of Neuroscience*, 25(34):7763–7770, 2005.

[75] A. Johnson and A.D. Redish. Neural ensembles in CA3 transiently encode paths forward of the animal at a decision point. *Journal of Neuroscience*, 27(45):12176–12189, 2007.

[76] A. Johnson, M.A.A. van der Meer, and A.D. Redish. Integrating hippocampus and striatum in decision-making. *Current Opinion in Neurobiology*, 17(6):692–697, 2007.

[77] M.A.A. van der Meer and A.D. Redish. Theta phase precession in rat ventral striatum links place and reward information. *Journal of Neuroscience*, 31(8):2843–2854, 2011.

[78] M.A.A. van der Meer and A.D. Redish. Ventral striatum: a critical look at models o learning and evaluation. *Current Opinion in Neurobiology*, 21(3):387–392, 2011.

[79] R.F. Salazar, W. White, L. Lacroix, J. Feldon, and I.M. White. NMDA lesions in the medial prefrontal cortex impair the ability to inhibit responses during reversal of a simple spatial discrimination. *Behavioral Brain Research*, 152(2):413–424, 2004.

[80] F. Naneix, A.R. Marchand, G. DiScala, J.R. Pape, and E. Coutureau. A role of the medial prefrontal cortex dopaminergic innervation in instrumental conditioning. *Journal of Neuroscience*, 29(20):6599–6606, 2009.

[81] F.P. Battaglia, A. Peyrache, M. Khamassi, and S.I. Wiener. Spatial decisions and neuronal activity in hippocampal projection zones in prefrontal cortex and striatum. In S. Mizumori, editor, *Hippocampal Place Fields: Relevance to Learning and Memory*, chapter 18, pages 289–311. Oxford University Press, 2008.

[82] E.L. Rich and M. Shapiro. Rat prefrontal cortical neurons selectively code strategy switches. *Journal of Neuroscience*, 29(22):7208–7219, 2009.

[83] P. Bonasso and T. Dean. A retrospective of the aaai robot competions. *AI Magazine*, 18(1):11–23, 1997.

[84] E. Gat. On three-layer architectures. In D. Kortenkamp, R.P. Bonnasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems.*, pages 195–210. AAAI Press, 1998.

[85] D. Kortenkamp and R. Simmons. Robotic systems architectures and programming. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics.*, pages 187–206. Springer-Verlag, 2008.

[86] J. Minguez, F. Lamiraux, and J.P. Laumond. Motion planning and obstacle avoidance. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics.*, pages 827–852. Springer-Verlag, 2008.

[87] M. Yarmohamadi, H.H.S. Javadi, and H. Erfani. Improvement of robot path planning using particle swarm optimization in dynamic environments with mobile obstacles and target. *Advanced Studies in Biology*, 3(1):43–53, 2011.

[88] A. Dickinson. Actions and habits: The development of behavioural autonomy. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 308(1135):67–78, 1985.

[89] M. Keramati, A. Dezfouli, and P. Piray. Speed/accuracy trade-off between the habitual and goal-directed processes. *PLoS Computational Biology*, 7(5):1–25, 2011.

[90] P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

[91] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[92] T. Zito, N. Wilbert, L. Wiskott, and P. Berkes. Modular toolkit for data processing (mdp): a

python data processing framework. *Frontiers in Neuroinformatics*, 2(0), 2009.
[93] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, April 1936.

Appendix A. Visual processing and place cells

Appendix A.1. Visual system

An overview of the complete visual system is given in Fig. A1. At the top, the BIPS hardware - Bio-Inspired Perception System hardware processors [4] - tracks objects within the field of view of both cameras. Fig. A2 gives an overview of the BIPS hardware. Visual input (15 fps) is passed through a set of elementary filters, similar to those in the primary visual cortex and prestriate cortex. Next the so-called tracking units compete through inhibitory connections to track objects based on feature coherency, similar to the extrastriate cortex. A simple matching procedure is used for stereoscopic vision. Objects tracked by the dominant eye (camera) are matched with objects seen by the other camera based on their location.

The elementary features such as shape, orientation and size are coded using neurons with Gaussian activation functions for each direction within the field of view in L2 of A1. As explained in Section 2.2, the combination of this information with the odometric system results in panoramic information of visual cues around the robot. It is important to note that the robot does not identify landmarks in this system, it only uses a constellation of detected objects.

Appendix A.2. Visual data clustering for place cells building

Different techniques with biologically inspired equivalents are available: Hebbian-like learning rules, Principal Component Analysis [42], Independent Component Analysis [90], Self-Organizing Maps [43] etc.

We initially tried linear approaches such as Principal Component Analysis [42] to check if the inputs were linearly separable, but the number of regions (place fields) that could be recognized was too low (the precision of the place cells would be too low to be usable).

A popular non-linear alternative for clustering are Self-Organizing Maps (SOM)[43]. In a SOM, a fixed number of neurons is used with a predefined topology. Normally a two-dimensional topology is used, so the SOM performs an N-to-2 dimensional mapping.

The goal of the SOM is to move the neurons in the high-dimensional input space to approach the topology of the input. For each input the Euclidean distance between the input and each neuron of the SOM is computed. The closest neuron is called the best-matching unit (BMU). The SOM is now updated by moving the BMU closer to the input (weighted sum) as well as its neighbors. One can use a fixed neighborhood or a decay factor for this.

SOMs are conceptually simple and they are a very powerful tool to discover clusters

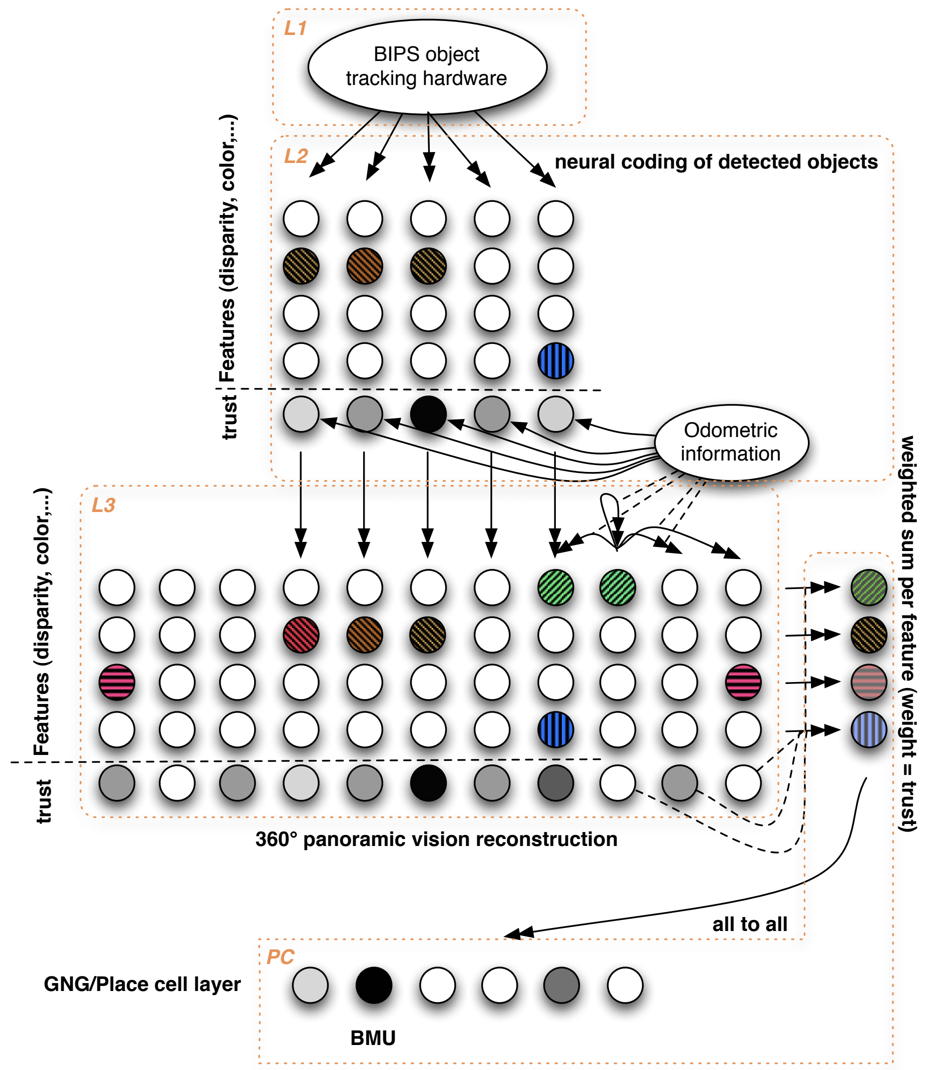

L1
BIPS object tracking hardware
L2
neural coding of detected objects
Features (disparity, color,...)
trust
Odometric information
L3
Features (disparity, color,...)
trust
weighted sum per feature (weight = trust)
360° panoramic vision reconstruction
all to all
PC
GNG/Place cell layer
BMU

Figure A1. Overview of the visual system. L1: First layer of the visual system, consisting of the Bio-Inspired Perception System hardware to detect and track objects. L2: In this layer the features of the objects from L1 are coded on a set of neurons. It only contains information on the objects within the field of view. L3: This layer contains a memory of all features around the robot (360 degrees). The L2 layer projects onto this layer to update the part of the memory currently within the field of view. The odometric system is used to modulate the projections from L2 to L3 to prevent updating L3 when L2 is expected to be unreliable. L3 is egocentric and hence there are lateral connections between neurons of different directions which are modulated by the odometric system (rotation of the head). PC: Finally, the features from L3 are averaged (weighted) over all directions and project onto the place cells.

in a dataset. A first disadvantage of SOMs is that the number of neurons is fixed. This problem can be overcome by trying out different sizes and visualizing the result. A more important problem is that the topology is fixed, i.e. the neighbors of a neuron stay the same. This problem is hard to overcome when the topology of the input is highly irregular.

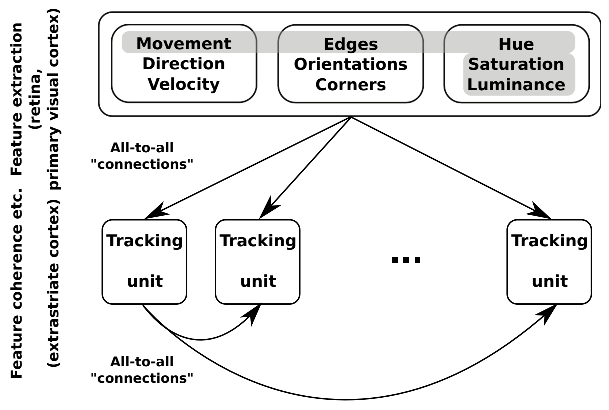

Feature coherence etc. (extrastriate cortex)
Feature extraction (retina, primary visual cortex)
Movement Direction Velocity
Edges Orientations Corners
Hue Saturation Luminance
All-to-all "connections"
Tracking unit
Tracking unit
...
Tracking unit
All-to-all "connections"

Figure A2. Details of the BIPS system. First elementary features are extracted and then objects are tracked by the tracking units.

A last problem is that one typically chooses a large number of neurons to create clearly distinct zones in the output. While this is not a disadvantage for visualization, one would prefer a small number of output neurons (i.e. place cells) in which each neuron codes for a distinct zone in the input space.

To overcome these problems we moved to a different and closely related type of artificial neural network, the so-called Growing Neural Gas (GNG) [44]. GNGs are created incrementally by inserting a new neuron after a number of input samples by splitting the neuron with the largest accumulated error (sum of distances) into two new neurons.

The topology itself is also learned by keeping the neighborhood of the neurons up to date. GNGs are generally better at approximating input topologies with high-error zones than SOMs, because the topology is not kept fixed and neurons are placed where they are most useful. A SOM can be seen as a predefined graph of which only the value of each node is updated (their position). A GNG also learns the graph by inserting nodes and edges. Both SOMs and GNGs are vaguely similar to the classical k-means algorithm [91], but the update rules are local.

We used the Modular toolkit for Data Processing implementation [92] of the GNG algorithm with the default parameters. The distance measure used was the Euclidean distance, as decorrelating the input variables with the Mahalanobis distance [93] did not improve the quality of the results.

We now compute the input to the GNG layer as follows, where d again refers to the orientation of the neuron, j to a feature and $j = N_{feat}$ to a trust neuron:

$$u_j^{PC}(t) = \frac{\sum_d n^{L3}_{d,N_{feat}}(t) n^{L3}_{d,j}(t)}{\sum{}_d n^{L3}_{d,N_{feat}}(t) n^{L3}_{d,0}(t) \dots \sum_d n^{L3}_{d,N_{feat}}(t) n^{L3}_{d,N_{feat}-1}(t)} \tag{A.1}$$

This equation is valid for all features except for the confidence neurons, which do not project onto the GNG layer (they only modulate the other inputs).

Because the GNG algorithm inserts a new neuron after a fixed number of samples, the number neurons would grow out of bounds after a while. To solve this problem, we fix the maximum number of neurons. This way, the animat creates new neurons at the beginning of the exploration and is forced to reuse the existing neurons when the maximum number of neurons is reached.

For a sample to be considered by the GNG-layer for learning, the mean trust level $(\sum_j n^j_{conf}(t)/N_{SC})$ must be greater than 0.75. This limit was found empirically as the quality of the place cells stopped improving above this value. A limit of 0.75 means that the robot needs to be able to see about 270 degrees of its environment from time to time. This is mostly due to the fact that we are using an open environment with resembling cues and a very noisy and unstable input. In a labyrinth with distinct cues in the corridors, we estimate that the robot can navigate with much less information.

Another solution to enable navigation with a lower mean trust level is to use a GNG-layer with much more neurons and to perform a longer exploration. This way the robot will learn orientation-dependent place cells. The problem is that the planning strategy needs to learn much more connections, because multiple representations will exist for the same place.

Appendix A.3. Place cells activation

The activation of the place cells is computed as follows:

$$u^{PC}_k = \frac{\sum_{o=1}^{O} n^{L3}_{conf,o} n^{L3}_{feat,o,k}}{||{n^{L3}_{conf}}^T n^{L3}_{feat}||} \tag{A.2}$$

$$\Delta_j = ||v_j - u^{PC}|| \tag{A.3}$$

$$r^{PC}_j = \frac{\Delta^\nu_j}{\max_i \Delta^\nu_i ||\Delta||} \tag{A.4}$$

Here v_j are the weights of the j-th node in the GNG (the j-th place cell). The index o stands for orientation (each orientation has a set of feature neurons and a trust neuron). $n^{L3}_{conf,o}$ is the o-th element of the vector (1 element per direction) of the vector of trust neurons n^{L3}_{conf}. Similarly, $n^{L3}_{feat,o,k}$ is the neuron of the k-th feature for direction o. Hence, n^{L3}_{feat} is an O by F matrix with O the number of orientations and F the number of features. We used $O = 960$ to prevent aliasing in L3 when the robot turns its head. As we sum over all orientations, the overhead of a large O only influences L3. ν defines the smoothness of the place cell activation and is a tuning parameter.

Throughout the main text, we use $n^{\bar{L3}}_{conf}$ to refer to the mean trust. This is defined naively as:

$$n^{\bar{L3}}_{conf} = \frac{\sum_{o=1}^{O} n^{L3}_{conf,o}}{O} \tag{A.5}$$

Appendix B. List of parameters

Name	Value	Explanation	Remarks
N_{PC}	100	No. of place cells	Arbitrary, but limits the precision of the gating network/topological map.
N_{PFC}	98	No. of nodes in topological map	Arbitrary, but here fixed to N_{PC}, minus the place cells that are never the BMU.
N_{ANG}	36	No. of direction neurons	Per connection in the topological map.
N_{DIST}	35	No. of distance neurons	Per connection in the topological map.
β	0.75	Min. trust for map/path update	
ι	0.7	Goal diffusion factor	
ζ	1 or ∞	Selection probability exponent	Lower values speed up learning, higher values make the robot follow an optimal policy.
λ	0.76	GN eligibility traces decay factor	
ξ	0.05	GN learning rate	
γ	0.8	Future reward decay factor	
τ_{forget}	0.02	Reward location decay rate	
τ_{learn}	0.2	Reward location learn rate	
ν	2	Place cells smoothness	
N_{GP}	36	No. of taxon input directions	Arbitrary, the precision can be higher than $2\pi/N_{GP}$ degrees because it's a population code.
N_{dir}	36	No. of taxon output directions	See N_{GP}
η	0.1	Taxon learning rate	
α	0.8	Taxon future reward discount rate	
κ	0.5	Taxon eligibility traces decay factor	
ω	$\pi/8$	Taxon action generalization	