Advances in Complex Systems
© World Scientific Publishing Company

Robustness of guided self-organization against sensorimotor disruptions

GEORG MARTIUS
Max Planck Institute for Mathematics in the Sciences
Inselstr. 22, 04103 Leipzig, Germany
martius@mis.mpg.de

Received (received date)
Revised (revised date)

Self-organizing processes are crucial for the development of living beings. Practical applications in robots may benefit from the self-organization of behavior, e.g. to increase fault tolerance and enhance flexibility, provided that external goals can also be achieved. We present results on the guidance of self-organizing control by visual target stimuli and show a remarkable robustness to sensorimotor disruptions. In a proof of concept study an autonomous wheeled robot is learning an object finding and ball-pushing task from scratch within a few minutes in continuous domains. The robustness is demonstrated by the rapid recovery of the performance after severe changes of the sensor configuration.

Keywords: Autonomous robots; learning; guided self-organization; robustness; dynamical systems; self-modeling.

Self-organization is a phenomenon observed in many complex systems. Examples are the pigmentation of animal coats, collective behavior of schools of fish or flocks of birds or ants, neuronal development in the brain (cortical maps), and formation of flows in pedestrian movements [20, 3]. Its central property is the emergence, meaning the spontaneous creation of structures and functions that are not directly explainable by the interactions of the individual constituents. Self-organizing systems have no central unit and are tolerant to faulty components [21]. Whereas traditionally engineered system are highly precise and reliable in their function, natural organisms are highly resilient and adaptive. Given these properties it appears attractive to utilize self-organization for controlling autonomous robots in order to make them less fragile against disturbances of all kinds.

There are different targets for self-organization in robotics. One is on the structural level, which is pursued by swarm robotics and modular robotics. In this paper we are concerned with self-organization on the behavioral and neural level. Self-organization is understood here as the formation of dynamical patterns that arise from the interaction of the robot with its environment, without being directly specified by the designer or a particular task. Many robotic implementations, especially in the Artificial Life community, use self-organization in one way or another, but only little work is done to obtain self-organized behavior from a general principle. Some studies have been devoted to the emergence of behavior from pure survival pressures utilizing artificial evolution [24, 25]. Another direction is undertaken by studies of intrinsically motivated behavior, typically in conjunction with reinforcement learning [17, 23]. Both face a complexity barrier due to computational costs or the curse of dimensionality.

A promising approach to the self-organization of behavior was provided by the homeokinesis principle [6] and the maximization of the predictive information [1]. Both approaches result in the establishment of low-level sensorimotor coordination with simple control structures, where simplicity is compensated by fast adaptation. The result is an on-going exploration of the sensorimotor patterns which depends strongly on the properties of the robot and its environment and is not hard wired [6, 10]. Similar properties are debated to characterize human perception of objects, which are supposed to be based on the set of suitable sensorimotor patterns or sensorimotor contingencies [16]. Self-organization, as we see it, can help to find appropriate sensorimotor contingencies.

Every autonomous learning system faces the classical dilemma of exploitation vs. exploration. In high-dimensional systems random exploration becomes quickly ineffective, whereas a behavioral self-organization may offer a scalable and embodiment driven exploration. This complies with the theory of embodied intelligence [18] which stresses the importance of the utilization of the particular embodiment. To pursue goals, however, the self-organized behavior needs to be guided in some way. In previous papers [12, 14] we investigated different mechanisms to guide the self-organization (GSO) with goal-related information and it was shown to be very effective, especially in high-dimensional systems [13]. In our formalism, GSO was used so far only with proprioceptive sensors (e. g. joint sensors). In this paper it is extended to the use of exterioceptive sensors, meaning sensors that measure quantities of the external world (e. g. a camera). These have a much more complex relation to the actions, for instance their values are determined by own movements and also by object movements. Moreover, data from these sensors are not always available and its quality differs a lot depending on the robots actions and location. The required changes of the algorithm are proposed and tested.

Let us specify the conditions and the outcome of the proposed guided self-organization algorithm. Given a robot in a changing environment with unknown sensors and motors, both in continuous domain, with the property that there are sensors measuring the causes of actions in a locally approximately linear way. Let the desired behavior be described by target sensor values (possibly time dependent). Targeted sensors must be dependent on actions (either directly or by their derivative) in a locally linear way. The algorithm establishes a continuously adapting sensorimotor coordination producing active behavior (sensor values are variate but well structured) and simultaneously learns how to achieve the target sensor values which are pursued with adjustable preference.

The task chosen to demonstrate the effectiveness of the algorithm is to find and push balls through an environment. This task is comparably simple and can be achieved with a suitably crafted PD-controller or similar. However, the point is not to provide a solution for a particularly hard or unsolved task, but instead to study the robustness of the approach to severe changes in the sensor configuration. For that we will change the orientation of the camera abruptly while the robot is interacting with the environment and is to maintain its task performance. The setup is somewhat similar to the human experiments with prism goggles that invert the retinal image [22]. It was found that humans can cope with such changes in their perceptions and restore their motor coordination substantially, suggesting an ongoing adaptation of sensorimotor contingencies. The same holds for our approach that relies on the continuous self-modeling of the sensorimotor loop which allows for learning from scratch and for coping with disruptions. Due to the lack of a long term memory, in our approach, previously learned coordinations cannot be recalled but need to be relearned.

An interesting study [2] achieved also a resilience to morphological changes by continuous self-modeling with a physics simulation as a highly specialized internal model. The physics simulation has very powerful prediction capabilities but can only model specific preprogrammed effects and lacks environmental factors (like a soft ground). In our view the important feature of self-organizing systems is exactly their ability to adapt to unforeseeable situations. A different route is to learn the interpretation of the sensor and motor apparatus and to build an hierarchical model [19]. It is a powerful method but it relies on a static environment and discrete actions. In this paper we use a generic internal model and continuous actions (instead of discrete predefined actions). Furthermore, the bootstrapping, self-modeling, and goal-direction is all continuously pursued online.

Robustness to sensor disruptions was also obtained with a combination of artificial evolution and homeostasis [8, 9], another paradigm of self-regulation, stabilizing the internal state against external perturbations. There a simpler task was used and it required thousands of generations. Importantly the perturbations had to be frequently present during the evolutionary process. We show that it can be achieved within a few minutes of life-interaction where unforeseen perturbation occur.

This paper is structured as follows: first the homeokinetic learning algorithm is briefly introduced in Section 1 and the guided self-organization mechanisms is presented in Section 2. The setup and the teaching inputs are given in Section 3 and the dynamics and behavior are analyzed in Section 4. Most notably the robustness to disruptions in the visual sensor configuration is investigated in Section 4.2. We finish with a discussion and outlook.

1. Self-Organized Closed Loop Control

Self-organizing control for autonomous robots can be achieved by establishing an intrinsic drive towards behavioral activity as described by the homeokinetic principle [4], for details cf. [6]. Let us start with an intuitive description before the mathematical formulation of the approach is given. The robot is controlled by a simple feed-forward neural network which receives the sensor values and produces the motor values. If the parameters (weights) of the controller network are fixed then we have a purely reactive setup and only relatively simple behaviors can be produced. But what if the parameters change in some intrinsic and embodiment-related way? Then, if done appropriately, a sequence of behaviors is obtained that are all locally smooth and simple but globally rather complex. The homeokinetic principle is exactly about such a parameter dynamics. In order to obtain it the robot is additionally equipped with an adaptive internal predictor for the next sensor values. In a nutshell, the approach consists of adapting the parameters to maximize prediction quality and simultaneously to maximize sensitivity to changes in the sensor values.

Formally, we denote the sensor values at time t as $x_t \in \mathbb{R}^n$. The actions $y_t \in \mathbb{R}^m$ are generated by a controller function

$$y_t = K\left(x_t, C, h\right) = g\left(Cx_t + h\right) \tag{1}$$

where $g(\cdot)$ is a componentwise sigmoidal function, we use $g_i(z) = \tanh(z_i)$, $C \in \mathbb{R}^{m\times n}$ is the weight matrix and $h \in \mathbb{R}^m$ is the bias vector (one-layer neural network). All vectors are considered to be column vectors. The internal predictive model M maps the sensor values and actions to the predicted sensory inputs and is given by a simple linear neural network as

$$x_{t+1} = M(x_t, y_t, \mathcal{A}) + \xi_{t+1}, \tag{2}$$

$$M(x_t, y_t, \mathcal{A}) = Ay_t + Sx_t + b, \tag{3}$$

where $\mathcal{A} = (A, S, b)$ is the set of parameters and ξ is the mismatch between the predicted and the actually observed sensor values. The matrix $A \in \mathbb{R}^{n\times m}$ captures the effect of the actions onto the new sensor values, the matrix $S \in \mathbb{R}^{n\times n}$ models the intrinsic dynamics of the sensor values, and the vector $b \in \mathbb{R}^n$ represents an offset. Plugging Eq. 1 into Eq. 2 yields

$$x_{t+1} = M(x_t, K(x_t, C, h), \mathcal{A}) + \xi_{t+1} = \psi(x_t) + \xi_{t+1} \tag{4}$$

which is the dynamics system describing the sensor value evolution. All parameters are adapted and thus carry a time index, which is however omitted for clarity. More complex parametrization, e. g. with multilayer networks are discussed in [5]. We use the simple neural network here for better analyzability.

In order to improve the prediction quality of the internal predictive model, the parameters $\mathcal{A}$ are adapted online to minimize the prediction error $\|\xi\|^2$ (Eq. 4) via gradient descent. The learning rules will be given below.

If the parameters of the controller (C, h) are also adapted by the minimization of the prediction error then stable but typically trivial behaviors are achieved. The reason is that the sensorimotor dynamics is stabilized against perturbation, such that the robot may get trapped in any state with $\xi = 0$ which happens prevalently when it is doing nothing. In order to bring activity into the sensorimotor loop the

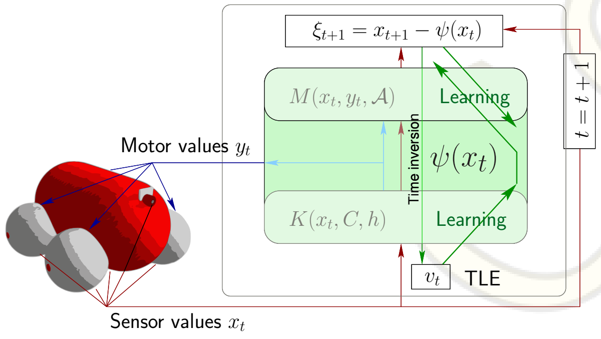

Fig. 1: The homeokinetic controller connected to a driving robot in the sensorimotor loop. The robot is equipped with wheel counters and a camera. The homeokinetic controller consists of the controller function K and the predictor M, both together form ψ (Eq. 4). The TLE is obtained by propagating ξ_{t+1} through ψ in inverted time.

homeokinetic paradigm [4, 6] instead suggests to use the so-called *reconstruction error*[a]. This error is the mismatch

$$v_t = x_t - \psi^{-1}(x_{t+1}) \tag{5}$$

between true sensor values x_t and reconstructed sensor values $\psi^{-1}(x_{t+1})$ assuming that ψ is invertible[b]. Intuitively it is the amount by which the sensor values should have been changed in order to compensate for the prediction error. The objective function minimizing the reconstruction error v_t is called *time-loop error* (TLE) and it can be approximated using the linearization $v_t = L^{-1}\xi_{t+1}$:

$$E_{TLE} = \|v_t\|^2 = \xi_{t+1}^\top \left(L_t L_t^\top\right)^{-1} \xi_{t+1}, \tag{6}$$

where $L_{t,ij} = \frac{\partial \psi(x_t)_i}{\partial x_{t,j}}$ is the Jacobian matrix of ψ at time t. Note that minimizing this error quantity increases the small eigenvalues of L, i. e. destabilizes the system, which is confined by the nonlinearities ($g(\cdot)$ in Eq. 1). This eliminates the trivial fixed points (in sensor space) and enables spontaneous symmetry breaking phenomena. Fig. 1 illustrates how the homeokinetic controller is connected to a robot.

The parameters of the controller (C, h) are adapted by a gradient descent on the TLE (6). This gives rise to the following parameter dynamics (all variables carry

[a]It was also called postdiction error in some papers, but we prefer the term reconstruction error.
[b]Using linearization and pseudoinverses, see below, the invertibility will not be required.

time index t):

$$\Delta C = -\epsilon_c \frac{\partial}{\partial C} E_{TLE} = \epsilon_c \mu v^\top - \boldsymbol{\epsilon} y x^\top \tag{7}$$

$$\Delta h = -\epsilon_c \frac{\partial}{\partial h} E_{TLE} = -\boldsymbol{\epsilon} y \tag{8}$$

where $\epsilon_c = 0.05$ is chosen for the learning rate. The diagonal matrix of channel dependent learning rates $\boldsymbol{\epsilon}$ is given by its matrix elements as

$$\boldsymbol{\epsilon}_{ij} = 2\epsilon_c \alpha \delta_{ij} \mu_i \zeta_i \tag{9}$$

with δ_{ij} being the Kronecker delta and α being a parameter for adjusting the sensitivity to perturbations ($\alpha = 1$ yields the original rules, we use $\alpha = 2$ for a higher sensitivity). The vectors $\zeta \in \mathbb{R}^m$ and $\mu \in \mathbb{R}^m$ are defined as $\mu = G' A^\top \left(L^\top\right)^{-1} v$, and $\zeta = Cv$ where G' is the diagonal matrix defined as $G'_{ij} = \delta_{ij} g'_i(Cx + h)$. For our parametrization the Jacobian matrix is given as $L = AG'C + S$.

We will deal with the case of more sensors than motors ($n > m$), which requires some adaption of the calculations for the following reason. For $S = 0$ the Jacobian matrix L is singular and cannot be inverted. Therefore all inversions are understood as pseudoinverses, however this can be done in different ways as detailed in [5] and it turns out to be relevant for the successful integration of additional sensors. In this paper the method is used for the first time. The implementation is given in the appendix Section A.1.

As mentioned above, the parameters $\mathcal{A}$ of the model are adapted online to minimize the prediction error $\|\xi\|^2$ (Eq. 4). However, the minimization is ambiguous with respect to A and S because y is a function of x, see (1). This is a general problem in learning forward models with reactive controllers. There are several options to deal with the ambiguities, but in any case we need to introduce a bias for capturing as much as possible of the relationship by the matrix A. Previously we used a simple discounting of the sensor branch (S). A slightly better approach is discussed in [5] which uses partly the TLE for the model learning. The learning rules are:

$$\Delta A = \epsilon_A \xi_{t+1} \left(y_t + \rho G' C v\right)^\top, \tag{10}$$

$$\Delta S = \epsilon_S \xi_{t+1} x_t^\top, \quad \Delta b = \epsilon_b \xi_{t+1}, \tag{11}$$

where ρ is a parameter to adjust the above mentioned bias and $\epsilon_A = 0.05$ and $\epsilon_S = \epsilon_b = 0.005$ are learning rates. In contrast to earlier publications we use here a smaller learning rate for S and b. Note that for $\rho = 0$ the original delta-rule is restored, we use $\rho = 0.1$. The choice of the parameters is not critical.

The learning rates are chosen to result in a fast synaptic dynamics, such that the system adapts quickly. Assuming sensory noise, the TLE is never zero nor has a vanishing gradient such that the rule (7) produces an itinerant trajectory in the parameter space, i. e. the robot traverses a sequence of behaviors that are determined by the interaction with the environment. These behaviors are, however, waxing and waning and their time span and transitions are hard to predict.

To get an idea of the resulting dynamics let us consider as a first example a robot with two wheels, each equipped with a wheel velocity sensor. In the beginning the robot rests, but after a short time it autonomously starts to drive forward and backward and to turn. If a wall is encountered such that the wheels stop the robot will immediately stop the motors and eventually drive in the free direction. Furthermore, high-dimensional systems such as snake- or chain-like robots, quadrupeds, and wheeled robots have been successfully controlled [6, 7]. It is of particular interest that the control algorithm induces a preference for movements with a high degree of coordination among the various degrees of freedom.

2. Guided Self-Organizing Control by Teaching

Guided self-organization focuses on the interplay between the emergent dynamics implied by self-organization and additional drives. The challenge in the combination of a self-organizing system with external goals becomes clear when recalling the characteristics of a self-organizing system. One important feature is the spontaneous breaking of symmetries of the system. This is a prerequisite for spontaneous pattern formation and is usually achieved by self-amplification. A nonlinear stabilization of the self-amplification forms another ingredient of self-organization. These two conditions are to be met for a successful guidance of a self-organizing system. A review of different approaches to guide the homeokinetic controller was given in [12]. Here we will restrict ourselves to one form of guidance where an external goal is given in terms of target sensor values at each moment in time. This is closely related to distal learning [11] which deals quite generally with the situation where some desired outputs are provided in a different domain than the actual controller outputs. Usually a forward model is learned that maps actions to sensations (or more generally to the space of the desired output signals). Then the mismatch between a desired and actual sensation can be transformed to the required change of action by back-propagation or inversion. We have a forward model already at hand (2), which is linear an can thus easily be inverted. For more complicated tasks a complex predictive model can be used.

Let us denote the target sensor values as x_t^G (G for guidance). We can then obtain the mismatch to the true sensor values as $\xi_t^G = x_t - x_t^G$ and calculate from that the teaching signal at the controller output

$$\eta_t^G = A^+ \xi_t^G, \tag{12}$$

where A^+ is the Moore Penrose pseudoinverse. Now we can implement a learning rule for the controller parameter to minimize the teaching error $E_G = \|\eta_t^G\|^2$ by a gradient descent. Unfortunately, the simple sum of both gradients (on E_G and E_{TLE}) is likely to steer the system out of balance and a fixed weighting between the two gradients cannot easily be identified that would satisfy an adequate pursuit of the goal and maintaining explorativity/self-regulation. A solution to this problem is the scaling of the gradient of the teaching error such that both gradients become

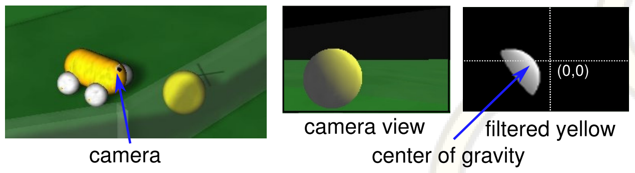

Fig. 2: Camera setup, image processing and sensor values.

compatible. It turns out that this transformation can be obtained using the natural gradient with a modified Jacobian matrix of the sensorimotor loop as a metric. The update for the controller parameters C is now given by

$$\frac{1}{\epsilon_c}\Delta C = -(1-\gamma)\frac{\partial E_{TLE}}{\partial C} - \gamma Q\frac{\partial E_G}{\partial C}, \tag{13}$$

where $\gamma > 0$ is the guidance factor deciding the strength of the guidance and Q defines the new metric given by $Q = A^\top \left(L_t L_t^\top\right)^{-1} A$ which works very stable. For the case considered here the explicit gradient in E_G is given by $Q\frac{\partial E_G}{\partial C} = -QG'\eta x^\top$. The weighted sum in Eq. 13 allows to select from the whole spectrum from pure self-organization to pure guidance and is an improvement to the earlier formulation [12]. For $\gamma = 0$ there is no guidance and we obtain the unmodified dynamics, cf. (7), and for $\gamma = 1$ there is no homeokinetic adaptation but only guidance. The parameters h_i are updated in the same way.

3. Integration of Vision

In the applications of the homeokinetic controller so far mostly proprioceptive sensors have been used. They are essential for developing a "feeling" of the body but for any environment related task exteroceptive sensors are required. Vision provides important information about the environment and about objects or other agents to interact with. Here we will for the first time investigate the integration of visual input into our framework of self-organizing control.

In the following we will conduct experiments with a four wheeled robot as introduced in Fig. 1. The robot is operated in two-wheeled mode, meaning that there are two motor channels and the motors on one side of the robot receive the same signal, which is the target wheel velocity. In addition to the two velocity sensors (x_l, x_r) (each is the average of the wheel velocities on one side) we attach a camera to the trunk of the robot, see Fig. 2. However, to be integrated in our dynamical system, which uses a linear model, the plain array of pixel values have a too complicated relation to the actions of the robot.

To simplify the scenario we only want the robot to interact with objects of a certain color, yellow in our case. Firstly, we calculate the center of gravity for horizontal (x_h) and vertical (x_v) direction of all yellow pixels obtaining a position each normalized to $[-1, 1]$. Note, for multiple objects this results in a position somewhere between them. Secondly, we approximate the size (x_s) of the object by the sum of all yellow pixels (normalized to $[0, \sqrt{2}]$). This is prone to light and shadow effects and is again a very crude approach but it will be shown to be sufficient for our applications. In addition we also provide the time derivatives of both quantities such that the vector of sensor values reads

$$x = (x_l, x_r, x_h, \dot{x}_h, x_v, \dot{x}_v, x_s, \dot{x}_s)^\top, \tag{14}$$

see Section A.2 for technical details. We typically add some sensor noise to the simulated sensors to make them more realistic. As a side effect the TLE does not become zero. The vision sensors are, given an object is visible, rather inaccurate and noisy due to light and shade, such that there is no need to add additional noise. So only the wheel velocities sensors x_l and x_r contain normal distributed noise ($\mathcal{N}(0, 0.02)$).

Exteroceptive sensors in general and our vision sensors in particular are not active for substantial periods during the behavior. For instance the position sensor (x_h, x_v) is essentially undefined if no object is in sight. Why is that a problem? The predictive model must correlate the actions with the sensations, and if there is no object to see then the correlations will be obviously zero. A simple solution is to prevent learning of the predictive model on invalid sensor values. We implement this by setting the undefined sensor value to 0 and modify the prediction error as follows:

$$(\xi_i)_t = \begin{cases} 0 & \text{if } (x_i)_t = 0 \text{ or } (x_i)_{t-1} = 0 \\ (\xi_i)_t & \text{otherwise.} \end{cases} \tag{15}$$

Note if an object is visible then the exact value of 0 is very unlikely, so that this double meaning is not critical.

3.1. *Guiding toward the Ball*

We are now going to implement a guidance that drives the robot toward an object in sight. Let us consider the desired sensor state: in order to have the object in the center of the field of vision the position sensors (x_h, x_v) should be zero (since they are normalized to $[-1, 1]$), but in general we have a target position (p_h, p_v). If the robot should push objects, e. g. a ball, then the size sensor value (x_s) is to be large. Alternatively if the robot should keep a certain distance, for instance when interacting with other robots, then a smaller value is required. We denote the desired size by s.

With the linear predictive model the relation between actions and position/size can only be captured in certain situations. In particular we deal here with resting and moving objects, which cause a different sensor response. As a new mechanism we propose to use a desired value for the derivatives as well. Fortunately we can use the most simple set-point control formula with damping: $\dot{x} = -\alpha(x - x^{\text{Desired}}) - \mu\dot{x}$, where α is a rate and μ is the damping constant. This differential equation has a fixed point at $x = x^{\text{Desired}}$.

The sensor teaching vector x^G is thus given in components as

$$x_l^G = x_l, \qquad x_r^G = x_r, \tag{16}$$
$$x_h^G = p_h, \qquad \dot{x}_h^G = -\alpha(x_h - p_h) - \mu\dot{x}_h, \tag{17}$$
$$x_v^G = p_v, \qquad \dot{x}_v^G = -\alpha(x_v - p_v) - \mu\dot{x}_v, \tag{18}$$
$$x_s^G = s, \qquad \dot{x}_s^G = -\alpha\,(x_s - s) - \mu\dot{x}_s, \tag{19}$$

where $\mu = 0.1$ and $\alpha = 1$. Note, the wheel velocity sensors x_l and x_r produce no teaching signal. For the following experiments we use $p_h = p_v = 0$ (center position) and $s = \sqrt{2}$ (maximal size).

To implement the entire algorithm we first need to initialize all parameters A, b, C, h, S with 0 except for $C_{ii} = A_{ii} = 0.9$[c]. Then the following steps are iterated forever: acquire sensor values x_{t+1} then calculate x^G (16)–(19), η_t (12), and ξ_t (4) to update C, h (13) using (7),(8) and A, S, b (10),(11); finally calculate y_{t+1} using $K(x_{t+1})$ (1) which is sent to the motors. Note that the updating can only be done from the second time step on because the previous sensor values are required, which form the only memory beside the parameters.

4. Results

All the experiments are performed in virtual reality in our robot simulator [15]. The state and parameter dynamics runs with a frequency of 25 Hz, the physical simulation with 100 Hz.

4.1. *Pushing Balls*

The first experiment should test whether the guidance mechanism is able to influence the self-organized behavior to find and push balls. This involves the establishment of the required sensorimotor mappings from scratch in a changing environment (balls can move). The formal definition of the goal is specified by the target sensor state x^G Eqs. 16–19. We place the robot together with five balls into a circular corridor, as displayed in Fig. 3(a), such that the robot can possibly push a ball for a long distance without getting stopped by corners. Those parameters of the model (A) connecting to the vision sensors are initialized with zero, such that the guidance has no effect independently of the guidance factor. Recall that the forward model transforms the teaching signal to nominal changes in motor values Eq. 12, which will be zero if the model did not learn anything. Once the robot learns to move, the model starts to correlate actions with the visual sensors. In this way the guidance starts to actually influence the behavior, such that the robot sees a ball more often and the model can improve further. Eventually the robot starts to steer at a ball

[c] A and C must be chosen such that the first m eigenvalues of L are positive and preferable that the initial feedback strength in the loop is subcritical (no activity). Alternatively an initial motor babbling can be used to initialize them, see [5].

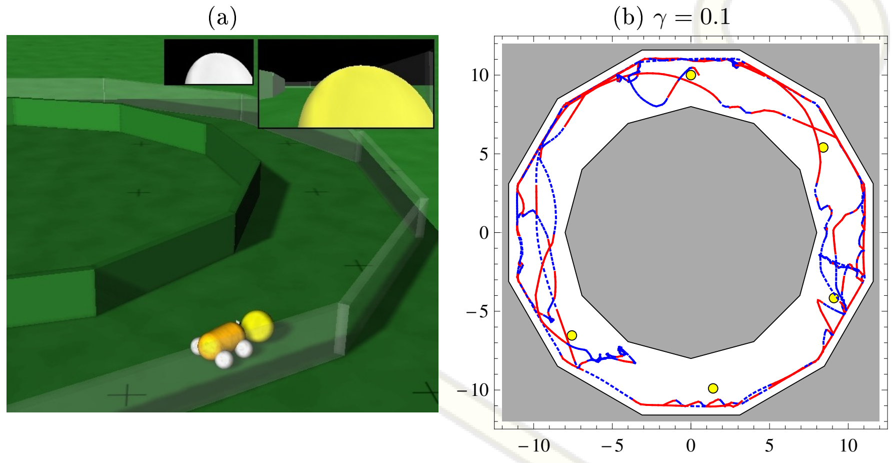

Fig. 3: Ball playing scenario. The robot is placed in a circular corridor. **(a)** Screenshot from the simulation. The right inlet shows the camera image and the left displays the color filtered image; **(b)** Part of a sample trajectory of the robot (minutes 5–10) for $\gamma = 0.1$ colored in red (solid) if the robot is close (within two body length) to the ball and it was in sight, and in blue (dashed) otherwise. The yellow disks show the initial positions of the balls.

and pushes it along the arena. Note that the robot has a round front shape such that the ball easily drifts away to either side while pushed. From time to time the robot still performs exploratory actions such that the ball gets lost and a ball needs to be found again. A part of a trajectory of the guided robot is shown in Fig. 3(b).

Note that there can be more than one ball in the field of view at the same time. However, the sensors cannot distinguish different objects, since the visual sensor (x_h, x_v) provides a position between the objects and the size (x_s) sensor returns a sum of the sizes. Nevertheless, the robot copes with this situation without problems. The robot steers at a group of balls and decides rather spontaneously which one it will touch. The final choice depends on how well the different balls are visible, when they leave the field of view, and other perturbations.

Let us quantitatively analyze the behavior. For that we consider the average distance to the closest ball and the cumulative time a ball was in the sight of the robot. This gives a good measure on whether the guidance was followed and the robot is indeed approaching the balls. If the robot is also pushing the balls along the arena, then the traveling distance of the balls raises, which we display together with the other quantities in Fig. 4(a). Indeed, for intermediate values of the guidance factor the time a ball was in sight increases from 100 sec to 600 sec. The same holds for the average distance of the robot to the closest ball which decreases from 5

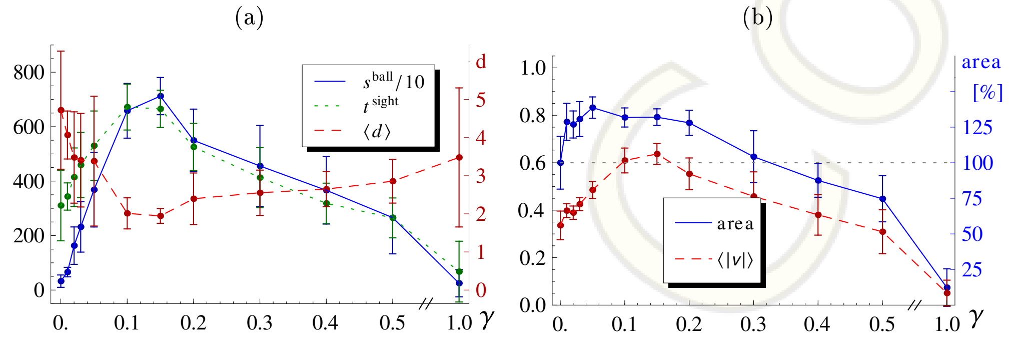

Fig. 4: Behavioral quantification of the ball playing scenario. Both panels show the mean and standard deviation of 10 simulations each 30 min long, in dependence of the guidance factor γ. **(a)** Traveling distance of the balls s^{ball} (scaled), cumulative time a ball was in sight t^{sight} (in sec), and average distance to the closest ball $\langle d \rangle$ (right axis, minimum 0.8). **(b)** Average absolute velocity of the robot (left axis) and area coverage (box counting method), given in percent of the case without guidance ($\gamma = 0$) (right axis).

to a value of 2[d]. Why does not the average distance go much below 2? Firstly, the plots include the entire simulation time including the phase where the robot has to acquire basic knowledge about its body. Secondly, it can take a long time and driving distance to find a ball again when it is lost, for instance through an exploratory action. Due to the inner circular walls of the arena the balls are not visible everywhere and finally the distribution of distances is skewed, see below.

The traveling distance of the balls raises from nearly zero to more than 7500 units (scaled by 1/5 in the plot), which corresponds to about 100 rounds in the arena (in 30 min).

In Fig. 4(b) we show that the aspects of the behavior that are not particularly subject to the guidance, namely the covered area of the arena by the robot and its average velocity are not negatively effected by the guidance, at least for moderate guidance strengths. The area coverage and the velocity go up when the task is performed, because the robot drives much more straight and forward than without the guidance.

When the guidance is too strong self-organized adaptation and external pressures become out of balance and the performance drops. Especially visible is this effect at $\gamma = 1$ where no homeokinetic learning takes place (Eq. 13) and the robot fails to move in a coordinated fashion, see Fig. 4(b).

Taking a closer look at the distance to the closest ball, we find that the mean is not such an appropriate measure in the guided situation since the distribution

[d](The size of the robot is 1 and the ball has radius of 0.3, resulting in a minimum of 0.8).

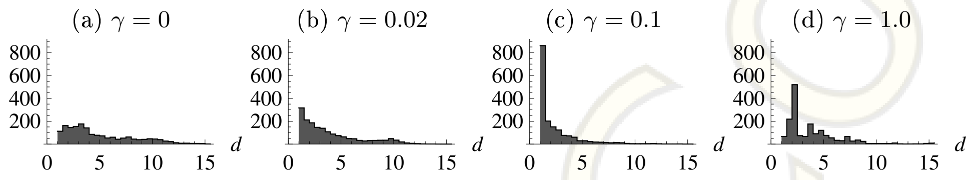

Fig. 5: Distribution of distance to the ball in the ball playing scenario. All panels show the histogram (in sec) of the distance d averaged over all simulations for one particular value of the guidance factor γ. **(a)** No guidance; **(b)** weak guidance; **(c)** intermediate guidance; **(d)** overly strong guidance (no self-organization).

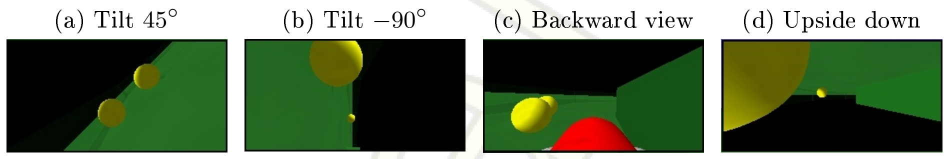

Fig. 6: Radical changes to the visual perception. In addition to the normal setup of the camera (Fig. 3) it is rotated by 45° **(a)**, −90° **(b)**, and 180° **(d)** along the optical axis, and lifted and rotated by 180° **(c)** along the vertical axis yielding a backward view. Note the different perspective and the appearance of the robot's body in the camera view in **(c)**.

of distances is not Gaussian but rather skewed as shown in Fig. 5. Without guidance the distribution of distances is almost flat, whereas for weak and intermediate guidance strengths the distribution is skewed with a strong preference for short distances. For overly strong guidance ($\gamma = 1$) the robot gets predominantly stuck at the walls because the sensorimotor coordination is pushed away from its sensitive regime, such that the histogram is rather arbitrary.

4.2. *Robustness against Structural Changes*

So far, the robot learned to control its body and to fulfill the task. It could be achieved similarly with evolutionary algorithms or even with hard-wired connections similar to the well known Braitenberg vehicles. There are, however, two big differences to our approach, firstly, the robot learns the behavior from scratch during the interaction with the environment and is thus robust against perturbations and, secondly, the robot does not get stuck at walls or in corners (shown below), all without infrared sensors. The latter would be quite difficult to achieve with a fixed wiring. In order to underpin the claim for robustness let us consider experiments that induce structural changes to the behaving robot. In fact we performed quite radical changes to the camera setup, namely to rotate and flip the camera abruptly,

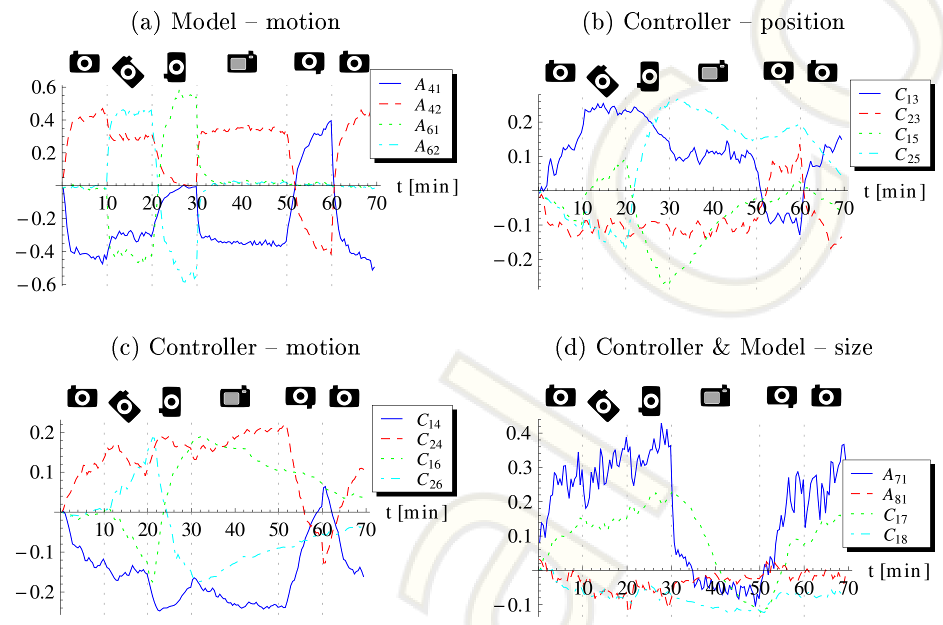

Fig. 7: Fast relearning: evolution of parameters for a changing camera. The camera is changed every 10 min, illustrated by the vertical lines. Its orientation on the body is shown by the icons. All values are mean values for 10 independent runs. Shown are elements of the model matrix (A) and controller matrix (C). The indexes refer to the sensor and motor value vectors, see Eq. 14. **(a)** Model parameters connecting left and right motor command with visual motion input ($\dot{x}_h$,$\dot{x}_v$). **(b)** Controller parameters connecting visual position (x_h,x_v) with left and right motor neuron. **(c)** Controller parameters connecting visual motion ($\dot{x}_h$,$\dot{x}_v$) with left and right motor neuron. **(d)** Controller and Model parameters connecting visual size (x_s,$\dot{x}_s$) and left wheel. The model parameters adapt very quickly to the new camera configurations. The controller utilizes both the position and the motion of the ball, however its adaptation is much slower compared to the model. Parameters: $\gamma = 0.1$.

see Fig. 6. These changes have severe consequences for the sensorimotor dynamics, because some sensor values swap signs or change from being useless to becoming important and vice versa.

We use the same circular arena as in the previous section. In our simulated experiments the camera setup is initially normal and is changed every 10 minutes to the setups shown in Fig. 6. Only the backwards view is kept for 20 min. Finally the normal setup is used again, such that an experiment lasted 70 min in total. We conducted 10 experiments with $\gamma = 0.1$ and present the evolution of the relevant model and controller parameters in Fig. 7.

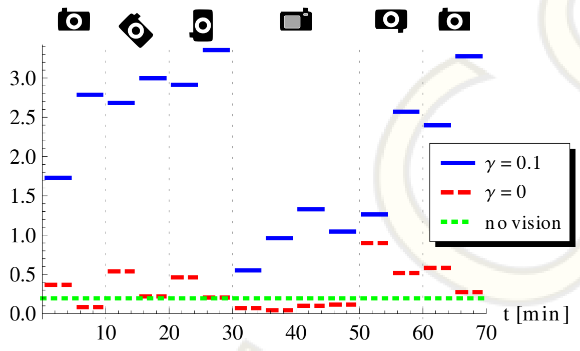

Fig. 8: Performance recovery for a changing camera configuration. Depicted is the summed average velocity of all balls within intervals of 5 min corresponding to the simulations in Fig. 7. For comparison the case without guidance ($\gamma = 0$) is displayed. The base line (green, dotted) represents the average ball movement of a blind robot.

Especially the model parameters relating motor values with the motion sensors, Fig. 7(a), evidently show that the correct correspondence is learned within a few minutes after each switching event. This, however, is only possible if the behavior of the robot is such that a ball remains frequently in the field of vision, which is very hard, if e.g., the positional sensation just swapped sign. In this situation the major strength of the homeokinetic controller shows its fruits, namely its continuous and embodiment related explorative and drive. The controller parameters show that the incorporation of the vision sensors is changed drastically for the different situations, but also that both motion and position information is used. The positional information is required to steer towards the ball and the motion sensor is used avoid overshooting. The parameters C change slower than the model parameters. Note that the behavior is also influenced by the parameters h (not shown). These change more rapidly and help to realize the teaching signals on a shorter timescale until the C matrix captures the correspondence with the sensor values.

How is the performance in the task after the structural changes? To answer this question we present in Fig. 8 the average ball velocities within 5 minute intervals summed over all balls. Note, that since the balls are subject to rolling friction a constant pushing is required. For comparison the values without guidance and without vision (chance level as a baseline) are displayed. The performance within the first 5 minutes is already far above the baseline and it is doubled from the first to the second 5 minute interval. After each structural disruption the performance drops a bit and is recovered in the second 5 min interval for each setting. Only the setting with camera pointing backward yields worse performance, which is due to the partial obstruction of the visual field by the body. Then the most drastic disruption occurs when the view is switched from backward to forward, but upside down. Here all visual sensor modalities change sign. Nevertheless the performance raises in the second 5 min interval to the performance of before. We can conclude that the performance is rapidly recovered even after severe changes in the sensor modalities.

Let us discuss what is adapted and what learned from scratch. At the beginning of an experiment the robot learns the behavior from scratch. When the camera is first turned by $45°$ then comparably small adaptations occur, see interval 10-20 min in Fig. 7. For instance the sensors for vertical position and motion get slowly integrated, but the remaining structure stays the same and in fact the performance drops only slightly (Fig. 8). When the camera is turned to $-90°$ then a drastic change occurs. The meaning of the size sensor does not change, but the position and motion sensors require a completely different coupling, which is slowly established (interval 20-30 min). This may be called learning from scratch, but in fact it is worse, it is learning from a wrong configuration. When the switch occurs the controller acts to avoid the ball. To manage this challenge an exploration is required that focuses on the wrong aspects of the model, which is what happens in our approach, where the adaptation speed is actually increased if the prediction errors raise (see Eq. 6). Since the controller does not explicitly know when a structural change occurs it is always adapting in a continuous manner. However, there is no long-term memory such that the controller cannot remember previously experienced configurations.

4.3. *Nontrivial Environment*

Finally, we will illustrate that the guidance is successful also in more complicated environments. The robot is placed together with four balls in an arena with two oval chambers connected by a wide corridor. The larger chamber has a wall in the center, see Fig. 9(a). As before the robot starts from scratch and has to learn how to move and to interpret the teaching signal.

In Fig. 9(b) a part of the trajectory of the guided robot is shown. The robot is visiting both chambers and pushes the balls not only at the walls but also in the free space e. g. along the corridor. Note the curvy shape of these parts that reflect the compensatory movements to keep the ball in front of the robot.

Let us have a look at the quantification of the behavior as above, see Fig. 10. Already for weak guidance ($\gamma = 0.05$) the task is substantially followed and up to $\gamma = 0.2$ the maximal performance is achieved. For stronger guidance the performance drops slowly until the robot stops moving in a coordinated way for $\gamma = 1$, because no self-organized adaptation occurs. The average distance to the closest ball drops drastically in the well guided case, cf. Fig. 10(a). The time of having a ball in sight raises up to about 500 sec out of 1200 total simulation time. Note that all plots include the entire simulation time including the phase where the robot has to acquire basic knowledge about its body.

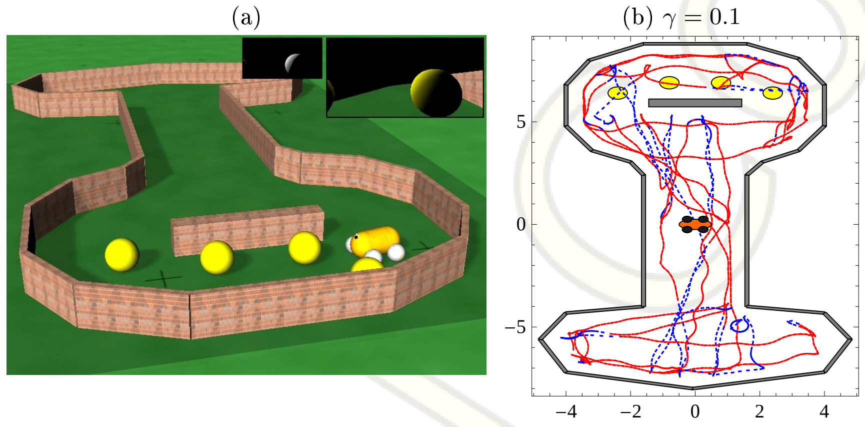

Fig. 9: Playing with 4 balls in a structured environment. **(a)** Screenshot from the simulation. The right inlet shows the camera image and the left displays the color filtered image. Note the effect of light and shade. **(b)** Part of a trajectory of the robot (min 5–10) for $\gamma = 0.1$. See Fig. 3 for a description.

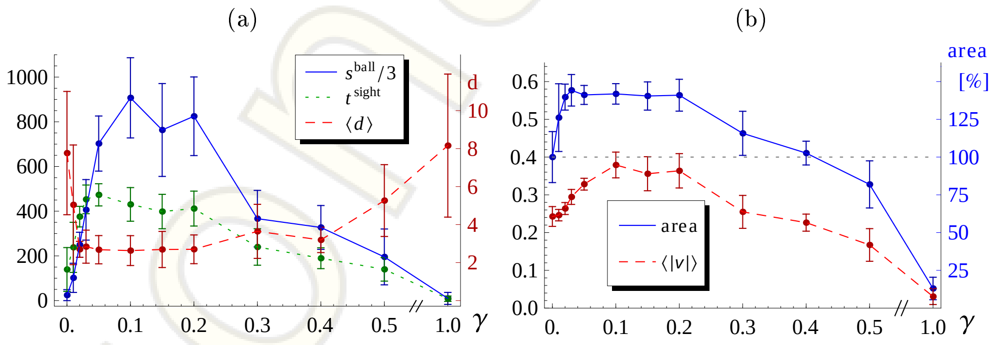

Fig. 10: Behavioral quantification for playing with many balls. Both panels show the mean and standard deviation of 10 simulations, 20 minutes long, in dependence of the guidance factor γ. See Fig. 4 for a description of the panels.

5. Discussion

The present paper elaborates on the robustness of guided self-organized control. We have presented here how to guide self-organizing behavior based on teaching signals in the visual domain and adapted the guidance mechanism to cope with exteroceptive sensors. The integration of visual sensors is very important since it allows an explicit perception of the environment and opens possibilities for a wider range of applications. We conducted experiments with a wheeled robot carrying a camera with simple visual preprocessing. The task was to learn from scratch how to find balls and push them around in an environment where the goal was only specified in terms of a desired visual sensor state. The entire sensorimotor coordination to fulfill this goal was learned by the robot within a few minutes. This involves the basic coordination to drive the robot and to integrate the vision sensors such that the balls are approached and balanced while pushed. The task to push the balls is not very complicated and can be achieved with a simple hand-crafted controller. However, to learn it from scratch in a short amount of time is hard. On top of that the orientation of camera was abruptly changed such that a completely different sensorimotor coordination becomes necessary. We found that guided self-organizing can cope with a wide range of configuration changes, even those where a complete change in the visual sensation occurs (signs of all visual sensors swapped). To our knowledge there is no other system that offers this kind of robustness and the rapid on-line learning. These properties are very attractive for developing autonomous robots and may be relevant for a wider range of applications.

The success of the approach to cope with such severe changes is due to exploration of the homeokinetic learning that is the faster the worse the predictive model. This in turn leads to a stronger exploration when unknown situations occur.

How important is the self-organizing part of the approach? When switching off self-organized adaption after the goal-oriented behavior was acquired it will be preserved, but the robot will get stuck at walls. Additionally the system will not be able to cope with strong changes, because of lacking exploration. On the other hand, when switching off guidance the task performance is kept for some time until the continuous parameter adaption will slowly destroy it. So both self-organization and guidance are required. We studied the impact of their balance and found good performance for a broad range of the guidance factor, which controls their relative strength. Also for the other parameters we found that their choice is rather uncritical.

So far the target sensor state has to be put in by the designer but one can also conceive that this can be learned from e. g. reward and punishment, similar to critic unit in the actor-critic approach to reinforcement learning. For more complicated tasks a more advanced predictive model has to be used, such that it is capable of capturing the consequences of the actions with respect to a task related sensor. The only restriction is that the model must be invertible or at least support an error back-propagation. If the former is true then it can be used for the homeokinetic learning as well, otherwise two predictive models can be used, one for the guidance and one for the self-organizing control.

The applicability of this approach to guided self-organization is limited to tasks, where the desired behaviors are at least to some extent produced by the homeokinetic controller. For instance for a legged robot some walking behaviors should be exhibited. Since only a very general objective is optimized it is an unlikely to obtain a walking behavior in the huge space of possible coherent behavioral patterns. So the guidance needs to be used on different levels. To overcome the limitation in applicability we are currently investigating a combination with reinforcement learning.

To conclude, guided self-organization is a promising route to control autonomous robots providing fast online learning and yielding robustness to changing conditions.

Acknowledgment: I thank Ralf Der and Michael Herrmann for fruitful discussions. The project was supported by the BMBF grant #01GQ0432 and the DFG (SPP 1527).

References

[1] Ay, N., Bernigau, H., Der, R., and Prokopenko, M., Information driven self-organization: The dynamical systems approach to autonomous robot behavior, *Theory Biosci., to appear* (2011).

[2] Bongard, J. C., Zykov, V., and Lipson, H., Resilient machines through continuous self-modeling, *Science* **314** (2006) 1118 –1121.

[3] Camazine, S., Franks, N. R., Sneyd, J., Bonabeau, E., Deneubourg, J.-L., and Theraula, G., *Self-Organization in Biological Systems* (Princeton University Press, Princeton, NJ, USA, 2001).

[4] Der, R., Self-organized acquisition of situated behaviors, *Theory Biosci.* **120** (2001) 179–187.

[5] Der, R. and Martius, G., Algorithmic implementation, in *[6]*, pp. 263–294.

[6] Der, R. and Martius, G., *The Playful Machine - Theoretical Foundation and Practical Realization of Self-Organizing Robots* (Springer, 2012).

[7] Der, R., Martius, G., Hesse, F., and Güttler, F., Videos of self-organized behavior in autonomous robots., `http://robot.informatik.uni-leipzig.de/videos` (2011).

[8] Di Paolo, E. A., Homeostatic adaptation to inversion of the visual field and other sensorimotor disruptions, *From animals to animats 6: Proceedings of the 6th International Conference on the Simulation of Adaptive Behavior* (2000) 440–449.

[9] Fine, P. F., Paolo, E. D., and Izquierdo, E., Adapting to your body, in *Proceedings of the 9th European Conference on Artificial Life*, eds. e Costa, F. A., Rocha, L. M., Costa, E., Harvey, I., and Coutinho, A. (Springer, Berlin, 2007), pp. 203–212.

[10] Hesse, F., Martius, G., Der, R., and Herrmann, J. M., A sensor-based learning algorithm for the self-organization of robot behavior, *Algorithms* **2** (2009) 398–409.

[11] Jordan, M. I. and Rumelhart, D. E., Forward models: Supervised learning with a distal teacher, *Cognitive Science* **16** (1992) 307–354.

[12] Martius, G. and Herrmann, J., Variants of guided self-organization for robot control, *Theory in Biosciences* (2012) 1–9.

[13] Martius, G. and Herrmann, J. M., Tipping the scales: Guidance and intrinsically motivated behavior, in *Advances in Artificial Life, ECAL 2011* (MIT Press, 2011), ISBN 978-0-262-29714-1, pp. 506–513.

[14] Martius, G., Herrmann, J. M., and Der, R., Guided self-organisation for autonomous robot development, in *Advances in Artificial Life 9th European Conference, ECAL 2007*, eds. Almeida e Costa, F., Rocha, L., Costa, E., Harvey, I., and Coutinho, A., *LNCS*, Vol. 4648 (Springer, 2007), ISBN 978-3-540-74912-7, pp. 766–775.

[15] Martius, G., Hesse, F., Güttler, F., and Der, R., LpzRobots: A free and powerful robot simulator, `http://robot.informatik.uni-leipzig.de/software` (2012).

[16] O'Regan, J. K. and Noë, A., A sensorimotor account of vision and visual consciousness., *The Behavioral and brain sciences* **24** (2001).

[17] Oudeyer, P.-Y., Kaplan, F., and Hafner, V., Intrinsic motivation systems for au-

tonomous mental development, *IEEE Transactions on Evolutionary Computation* **11** (2007).

[18] Pfeifer, R. and Bongard, J. C., *How the Body Shapes the Way We Think: A New View of Intelligence* (MIT Press, Cambridge, MA, 2006).

[19] Pierce, D. and Kuipers, B., Map learning with uninterpreted sensors and effectors, *Artif. Intell.* **92** (1997) 169–227.

[20] Polani, D., Foundations and formalizations of self-organization, in *Advances in Applied Self-organizing Systems*, ed. Prokopenko, M. (Springer, 2008), pp. 19–37.

[21] Prokopenko, M., Design vs self-organization, in *Advances in Applied Self-organizing Systems*, ed. Prokopenko, M. (Springer, 2008), pp. 3–17.

[22] Richter, H., Magnusson, S., Imamura, K., Fredrikson, M., Okura, M., Watanabe, Y., and Långström, B., Long-term adaptation to prism-induced inversion of the retinal images, *Experimental Brain Research* **144** (2002) 445–457.

[23] Schmidhuber, J., A possibility for implementing curiosity and boredom in model-building neural controllers, in *Proceedings of the First International Conference on Simulation of Adaptive Behavior* (MIT Press, Cambridge, MA, USA, 1990), pp. 222–227.

[24] Stefano, N., Behaviour as a complex adaptive system: On the role of self-organization in the development of individual and collective behaviour, *ComplexUs* **2** (2006) 195–203.

[25] Yaeger, L., Computational genetics, physiology, metabolism, neural systems, learning, vision and behaviour or polyworld: Life in a new context, in *Artificial Life III, Vol. XVII of SFI Studies in the Sciences of Complexity* (AddisonWesley, 1994), pp. 263–298.

Appendix A. Appendix

A.1. *Generalized Pseudoinverse*

The learning rules given in Eqs. 7 and 8 are still open to interpretation if the inverse of L does not exist, like in the case that the sensors outnumber the motors[e]. The usual way is to use the well known Moore-Penrose pseudoinverses. However, we want to introduce a generalization of that concept by way of a certain "sandwiching" technique, defining the inverse of a square matrix L as

$$L^{+}_{PQ} = Q\frac{1}{PLQ}P \tag{A.1}$$

choosing the matrices P and Q appropriately so that the inversion of PLQ becomes possible in the classical sense. This is trivial if all matrices P, Q, L are of full rank so that automatically $L^{+}_{PQ} = L^{-1}$. In all other cases the matrices P and Q have to be chosen appropriately so that the inverse of PLQ exists. For instance in our case rectangular matrices can be used that reduce the dimension of the matrix to invert to $(m \times m)$ (number of motors). This not only yields an invertible matrix (since L keeps at least rank m if initialized appropriately) but also reduces the computational cost. In our setup we have different natural choices for the matrices P and Q, namely

[e]Considering the case with the simple forward model ($S = 0$, which is the initial state) where $L = AG'C$. L is an $n \times n$ matrix and A and C are rectangular with one dimension $m < n$.

A, C and their transposes, since they compose L. We tried different variants, e. g. $L^+_{A^\top C^\top}$, $L^+_{A^\top A}$, and L^+_{CA} see [5], where the latter turned out to work best. Thus, all occurrences of L^{-1} are substituted by

$$L^+_{CA} = A\frac{1}{CLA}C. \tag{A.2}$$

Note, the regularization becomes time dependent, due to the varying "sandwiching" matrices, but this does not render a problem to our experience. Bear in mind that the inversion is not strictly defined and there is a whole set of valid solutions.

A.2. *Technical Camera Details*

The simulated camera we use here provides an array of $W \times H$ color pixels with $W = 256$ and $H = 128$ and it has field of view of 120° horizontal and 60° vertical. The pixel values are give in the HSV color model: hue, saturation, value/brightness. This makes the color distinction easier. Let us denote the pixels of the camera by the three arrays $h_{ij} \in [0, 180]$ (hue), $s_{ij} \in\in [0, 255]$ (saturation) and $v_{ij} \in [0, 255]$ (value/brightness) where $i = 0, \dots, H-1$ runs over all rows and $j = 0, \dots, W-1$ runs over all columns.

We select the pixels of interest using a very simple threshold method

$$p_{ij} = \begin{cases} v_{ij} & \text{if } h_{min} \le h_{ij} \le h_{max} \text{ and } v_{ij} > v_{min} \text{ and } s_{ij} > s_{min} \\ 0 & \text{otherwise} \end{cases} \tag{A.3}$$

where h_{min} and h_{max} define the color interval and v_{min} and s_{min} define the minimum brightness and saturation which we set to about 25% of the value range. Using the sum of all pixels values S $= \left(\sum_{ij} p_{ij}\right)$ we calculate the center of gravity as

$$(x_h, x_v) = \left(\frac{2}{S\,H} \sum_{ij} (i - H/2) p_{ij},\ \frac{2}{S\,W} \sum_{ij} (j - W/2) * p_{ij} \right) \tag{A.4}$$

normalized to interval $[-1, 1]$. The amount of interesting pixels in the view is interpreted as the size of the object in sight as is given by

$$x_s = \sqrt{S/(W\,H)/128}, \tag{A.5}$$

which is normalized to $[0, \sqrt{2}]$. Since objects are only partially visible when at the border of the field of view we set x_s to zero when the x_h or x_v is close to the border ($|x_h| > 0.75$). The time derivatives of both quantities are calculated using difference quotient of subsequent time steps with a scaling factor (5 for $\dot{x}_h$ and $\dot{x}_v$, and 10 for $\dot{x}_s$).