True autonomy from self-organized adaptivity

Ralf Der **René Liebscher**
Universität Leipzig
Institut für Informatik
POB 920
D-04009 Leipzig
{der|liebschr}@informatik.uni-leipzig.de

Abstract

The paper is a step towards a systematic approach to the self-organization of behavior. We consider autonomous robots controlled by artificial neural networks trained by supervised learning. The learning signals however are generated by the agent itself on the basis of objectives which are entirely internal to the agent. This paradigm is realized by equipping the agent with an adaptive model of its own behavior and using the misfit between model and true behavior as the learning signal. In this way the robot learns to behave in a predictable way. At this level, the approach requires the presence of a drive for action without which the the trivial "do nothing" behavior prevails. However we show that a simple change – the inversion of time in the modelling process – eliminates the trivial solutions and in a general sense introduces the driving force of any self-organization process namely the noise driven emergence of instable modes. The approach is closely related to the principle of homeokinesis which is the dynamical pendant of homeostasis as introduced by Cannon (1939) and later Ashby (1954).

1 Introduction

Adaptivity is one of the main incentives for an artificial being striving for autonomy. Commonly adaptivity is seen as a tool for improving the performance of an agent in the completion of its tasks. A prominent example for adaptivity in the broad sense is reinforcement learning which is a biologically inspired learning paradigm. It is a valuable tool in order to adapt an agent for a specific task without having to give it a concrete learning signal. Nevertheless in a specific domain one must input a lot of domain specific knowledge and moreover the kind of reinforcement one gives introduces a semantics from outside. Adaptation is guided by the designer in a more or less indirect way. In the last consequence reinforcement learning may be seen as a different way of programming the robot to do specific tasks. Although biologically inspired it is not clear whether these adaptation paradigms can create machines which are as inspired as biological beings.

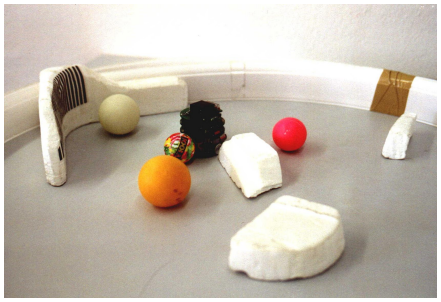

Figure 1: A typical environment where the Khepera robot learned to "live" in a purely self-organized way.

We will advocate a more rigorous understanding of autonomy which is based on the belief that true autonomy must involve the phenomenon of emergence. Before giving some ideas how this could be realized in the robotic domain let us first illustrate the goal in a concrete case. Consider a robot with a parameterized controller, a neural network, say, with synaptic weights initially in the *tabula rasa* condition. So there is no reaction of the robot to its sensor values and activities if present at all are only stochastic ones. The robot is to be in an environment with static and possibly also dynamic objects, cf. Fig 5 below. The task now is to find an objective which is entirely internal to the robot which drives the parameters so that the robot will start to move and while moving to develop its perception of the world and object related behavior. In particular in the case of moving objects like a ball the robot should learn to discriminate the ball from other objects and treat it different (play with it) since the ball reacts different to its activities. And of course (since this is a dream) we expect this to happen independently of the sensors the robot has and whether it moves with wheels or legs or ...

One possibility for the self-organized adaptation is that one uses a supervised learning algorithm for the

controller where however the agent is its own supervisor in the sense that he generates its own learning signals. This is the way we will follow in the present paper. This requires that the agent has an internal objective which generates the "internal drive" for the adaptation. The purpose of the present paper is to look for some possible candidates for such objectives and to investigate the behaviors which they produce.

We begin with the notion of (i) dynamical complexity. One may require that the trajectories (in sensor space) one observes are of a certain moderate complexity. This means that the robot neither stalls nor moves chaotically. The problem is how to define dynamical complexity in a domain invariant way. Another paradigm is (ii) that of the predictability of behavior which means to behave in such a way that the future values of the sensor readings stay predictable. We will demonstrate in the present paper how this engenders a smooth controlled behavior of the agent. Related to this is (iii) the sustainability of control. The aim would be to behave in such a way that after the current period of the "sense-think-act" cycle the agent is again in a state where control is possible in the same safe way.

Common to all these principles is the fact that they always have the "do nothing" behavior as a trivial solution. So, these principles work well only if the agent is given a drive for activity or curiosity or the like. We have presented this route to emerging behavior in earlier papers and will sketch the approach in the Sec. 2 below for the sake of completeness. The focus of the present paper mainly is on the emergence of drives. We will show in Sect. 3 that a simple change – the inversion of time in the modelling process – eliminates the trivial solutions and in a general sense introduces the driving force of any self-organization process namely the noise driven emergence of instable modes. We consider this the vital step towards emerging behavior. The appropriate measure is (iv) the stability of the time loop over one or more time steps. Preliminary results on the combination of the approaches are found in Sect. 4 which reports results with a Pioneer robot including the camera.

These ideas are rooted in the general framework of homeokinesis, cf. (Der et al., 1999) which is the dynamical pendant of homeostasis as introduced by Cannon (Cannon, 1939) and later Ashby (R.Ashby, 1954) and in the embodied intelligence approach (Pfeiffer and Scheier, 1999) as discussed in (Der, 2001).

2 Staying predictable

Let us now present some results under the predictability paradigm as formulated above. Our robot has an adaptive controller with output

$$y_t = K\left(x_t; c\right) \tag{1}$$

where $x_t \in \mathbf{R}^n$ are the sensor values observed at time t and c is the parameter vector. The adaptive model is to predict the true sensor values x_{t+1} at $t+1$ as

$$x^P_{t+1} = x_t + M\left(x_t, y_t; m\right) \tag{2}$$

with prediction error

$$E = \left\|x_{t+1} - x^P_{t+1}\right\|^2 = \left\|\Delta x_t - \mathcal{M}\left(x_t\right)\right\|^2 \tag{3}$$

where $\mathcal{M}\left(x_t\right)$ is shorthand for $M\left(x_t, y_t; m\right)$. We obtain the learning rules for the self-organized learning

$$\Delta m = -\eta \frac{\partial}{\partial m} E \qquad \Delta c = -\varepsilon \frac{\partial}{\partial c} E \tag{4}$$

Predictor and controller are to be learned concomitantly where model learning is much slower than the learning of the controller. In practical applications we may even use a model learned off line so that there is no interference between model and controller learning. In the experiments described below the model was the trivial one, i.e. the $M = 0$ so that there was no model learning at all.

We may interpret the above approach also in the terms of the dynamical complexity gap between internal (model) and external world. The idea above is to use the width of the complexity gap as learning signal for the adaptation of both the model and the controller. So the aim is to adapt behavior such that the complexity gap between true and model behavior is minimized.

By way of example, let us consider a robot moving with a fixed forward velocity, the controller output y being just the target turn speed of the robot in physical space. In the most simple setting we implement the controller by a single artificial neuron as [1]

$$y = \tanh\left(\sum_{i=1}^{n} c_i x_i\right) \tag{5}$$

where x_i is the current output of sensor i. In the case of the Khepera robot $x = (x_1, ..., x_8)$ might represent the values of the eight infrared sensors. The system is updated in regular times $t = 0, 1, 2, ...$ so that over the time the robot sees the trajectory x_t, $t = 0, 1, \ldots$ in sensor space the trajectory depending both on the environment and the starting position of the robot.

In the present paper we consider the most simple case $M = 0$, i.e. the model assumption is that the sensor values are constant over time. In order to update c according to eq. (4) we need the gradient of E with respect to the parameters of the controller. This gradient information is gained by a simple trick. We add a small

[1] We assume that the controller does not refer to earlier sensor values as a PD controller would do. However the latter case can easily be integrated into the present concept.

perturbation $h(t)$ to the postsynaptic potential z of the controller neuron, i.e. put

$$y = \tanh\left(\sum c_i x_i + h(t)\right) \tag{6}$$

and consider that the linear response of E to the perturbation is essentially $h(t)\,\partial E/\partial z$. From this we get the following learning rule

$$\Delta c_i = -\varepsilon\, x_i\, h(t)\, E \tag{7}$$

where $E = \sum_j (\Delta x_j)^2$ and $\Delta x = x_{t+1} - x_t$. Equations (6) and (7) constitute the complete algorithm for the present robot controller and can easily be translated to the case of a neural network as controller.

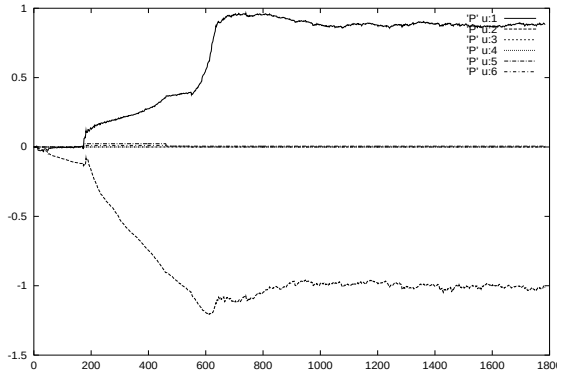

Figure 2: Wall following behavior. The robot sees the wall to its left. Plotted are the values of the controller parameters c_1 through c_6. One step is about 2 cm. After the very short initial learning period the robot followed the inner wall of a circle of about 80 cm. After about 200 steps the robot was transmitted to the outer wall of a circle of about 60 cm. The robot rapidly adapts to the new situation and stays stable in the new behavior for about 400 steps. Initiated by an external perturbation starting by step 600 there is a transition into a more stable (closer) wall following modus. The latter behavior is completely stable.

We have implemented the above algorithm on chip of a Khepera miniature robot. Starting from the *tabula rasa* initial condition the robot in a few minutes learned a number of different behaviors. Behaviors are contingent since there is no specified target behavior. Which behavior is going to emerge depends on the environmental conditions. For instance if the trainer keeps a ball in front of the robot for a while then the robot rapidly learns to stabilize the ball between its front sensors. This is obviously an appropriate behavior since in the ideal case this leaves the sensor values invariant.

In the same way one can train (or retrain) the robot to follow a wall by starting it repeatedly under a convenient angle to the wall so that it feels the benefit (which means minimum change in sensor values) from keeping

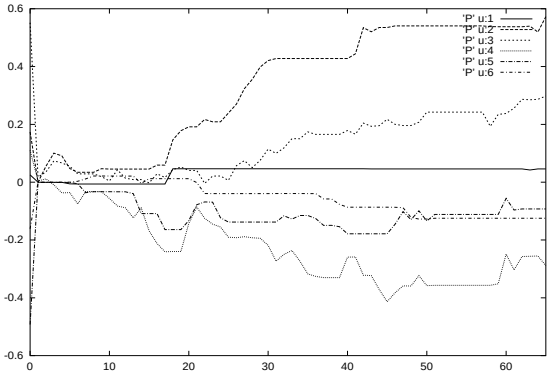

Figure 3: Ball balancing behavior. The robot moving with a constant velocity is presented a table tennis ball which is stabilized for some time by hand in front of it. After about 40 cm the robot is able of balancing the ball as long as its speed is sufficient to keep up with the it. The plot displays the parameters c_2 through c_5 of the controller. Convergence to a stable ball balancig behavior is reached after about 40 steps.

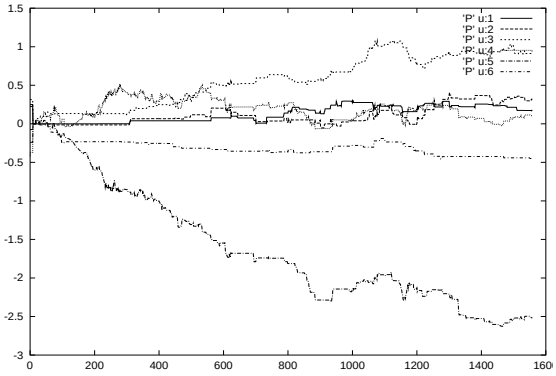

Figure 4: The right hand side front sensor was damaged nevertheless the robot learns a stable ball balancing behavior.

itself parallel to the wall . In the same way one may condition the robot to obstacle avoidance, optimal navigation through narrow corridors, and different modi of wall following. In many situations we observed that the controller converged towards the best working conditions, i.e. the behavior where control is most safely realized. The latter effect results from a nontrivial combination of nonlinearity and noise effects.

In the experiments the learning rate is chosen rather large, $\varepsilon = 0.7$ in the case of the Khepera. The choice of the oscillation frequency and amplitude is a little critical since one must guarantee that the distance covered in one period is sufficiently small so that the oscillation frequeny is to be high. This however is in conflict with the delay times caused by the inertia of the robot and other hardware effects.

3 Emerging drives from minimizing the time-loop error

Under the above paradigm nontrivial behaviors are emerging only if the robot is given a drive for activity (the constant forward velocity above). Of course these drives should emerge as well. Interestingly this happens already from a minor change of the above paradigm which formally consists in changing the direction of time in the prediction step.

3.1 The time loop

Let us call $\hat{x}_t$ the vector of sensor values as reconstructed from the true sensor values x_{t+1} observed one step later in time. We write the model backward in time as

$$\hat{x}_t = x_{t+1} + \mathcal{M}^{(-)}(x_{t+1})$$

which can be trained in the usual way. The difference

$$u = x_t - \hat{x}_t$$

is obtained from going from x_t to x_{t+1}, i.e. one step forward in time through the full world whereas the step from x_{t+1} to the reconstructed state $\hat{x}_t$ is backward in time. We call the sequence $x_t \rightarrow x_{t+1} \rightarrow \hat{x}_t$ the time loop and correspondingly

$$F = \|u\|^2 \tag{8}$$

the time loop error which depends on both c and m so that after having obtained u gradient descent yields the update for both c and m.

The time loop error is driven by two opposite forces. On the one hand the error is small if the current behavior is well repesented by the model. This favors behaviors corresponding to more or less predictable sensor values (smooth trajectories in sensor space) like keeping a ball under control as discussed above. In particular it favors the trivial behaviors. On the other hand we can write the full dynamics always as

$$x_{t+1} = x_t + \mathcal{M}(x_t) + \xi_t$$

where ξ_t is the part of the dynamics not covered by the model. Now

$$\hat{x}_t = x_{t+1} + \mathcal{M}^{(-)}(x_t + \mathcal{M}(x_t) + \xi_t)$$

so that $\mathcal{M}^{(-)}(x_t + \mathcal{M}(x_t)) = x_t$ which shows that the influence of the perturbation ξ is obtained by propagating it backward in time through the model. Time reversal inverts stable into instable behavior and vice versa. Therefore the modeling error is **minimized** if the behavior of the robot is **instable** in the forward in time direction. Consequently the trivial behaviors corresponding to stabilizing the robot are going to be destabilized by this mechanism. This creates new instable modes in the system which may loosely be associated with emerging drives for activity. However these modes can not grow unrestrictedly. Instead by the first mechanism they are so to say canalized in order to stay predictable.

3.2 Illustrating the time-loop error

In a nutshell the whole story is clear from the following oversimplified model. Consider the linear system with dynamics

$$x_{t+1} = K(x_t) + \xi \tag{9}$$

using $K(x) = cx$. The model is

$$x^P_{t+1} = cx \tag{10}$$

forward in time and

$$\hat{x}_t = c^{-1} x_{t+1} \tag{11}$$

backward in time so that the time loop error is

$$F = (\hat{x}_t - x_t)^2 = \xi^2 c^{-2} \tag{12}$$

with learning rule

$$\Delta c = \varepsilon \xi^2 c^{-3} \tag{13}$$

so that c is monotonously increasing by the learning. Obviously for $0 < c < 1$ the state x is stabilized at $x = 0$. If x is the velocity of the robot the latter will stay at rest. However the concomitant learning dynamics increases c and once $c > 1$ x starts diverging so that asymptotically either $x \rightarrow \infty$ or $x \rightarrow -\infty$. Starting with $x = 0$ it depends entirely on the noise which branch is chosen. This is what a spontaneous symmetry breaking is like.

3.3 The basic sensor-motor loop

In order to exemplify the nonlinear case we consider a sensor-motor loop consisting of a neuron which is to control the forward velocity x of a robot. The neuron output $y = K(x_t)$ prescribes the new target velocity x. Its input in the next time step is the true forward velocity x_{t+1} as read back from the wheel counters. Choosing $K(x_t) = \tanh(cx + H)$ the dynamics of this sensor-motor loop is given by

$$x_{t+1} = \tanh(cx_t + H) + \xi_t \tag{14}$$

where H may be interpreted as a bias and ξ_t is the noise due to the difference of the target and the true velocity of the wheels caused by slip and friction effects and so on. Without noise the dynamics formulated by eq. (14) for any set of parameters always converges to a fixed point where x does not change any more (the robot moves with constant velocity), the fixed point equation reading

$$x = \tanh(cx + H) \tag{15a}$$

With $H = 0$ there is a stable solution $x = 0$ for $0 < c < 1$ while for $c > 1$ there are two stable fixed points at $x = \pm q$ where q runs from 0 to 1 for increasing values of $c > 1$. The position of the FPs and hence of the speed of the robot is further modified by the value of H. With $c \gg 1$ we obtain a strong hysteresis effect since the influence of H is felt only after it is moved far enough to the other side. This is a pecularity of the present controller paradigm and is found to have a lot of interesting consequences in practice.

Assuming the average $\overline{\xi} = 0$ we find in a low noise approximation the time loop error as

$$F = u^2 = \left(\frac{x_{t+1} - y}{c\left(1 - y^2\right)} \right)^2$$

so that the learning rules are

$$\begin{aligned} \Delta c &= \varepsilon\left(x\right)\left(c - 2c^2 xy\right) \\ \Delta H &= -\varepsilon\left(x\right) c^2 y \\ \varepsilon\left(x\right) &= \varepsilon_0 \frac{\left(x_{t+1} - y\right)^2}{c^4\left(1 - y^2\right)^2} \end{aligned} \tag{16}$$

where $y = \tanh\left(cx_t + H\right)$ is the output of the neuron after seeing input x_t and $\varepsilon\left(x\right)$ may be seen as an effective learning rate. For practical applications it is convenient to use a regularization which means to use $\left(A - y^2\right)$ with $A > 1$ in the denominator. In the experiments we mostly used $\varepsilon\left(x\right) = \varepsilon_0\left(x_{t+1} - y\right)^2$ with no qualitative change in the learning behavior.

The behavior of the complete dynamics given by eqs. 14 and 16 is roughly sketched as follows. We consider first the case where c and H are small so that the tanh may be linearized so that in leading order and in the average over the noise

$$\Delta c = \varepsilon_1 c^{-3}$$

and $\Delta H = 0$. Using $c\left(0\right) > 0$ we obtain a monotonous increase in c which is $\sim \sqrt{t}$ for large t. Once $c > 1$ is reached this means that the state dynamics changes from a stabilizing behavior at $x = 0$ to x exploding exponentially. In this way an unstable mode is created - the robot starts moving. The direction of motion is determined by the noise so that we have a noise driven spontaneous symmetry breaking. For larger values of x this mode is caught by the nonlinearities but the additional degree of freedom introduced by the H dynamics drives the system into an irregular limit cycle.

3.4 Experiments

We have used the learning procedure of eq. 16 to control the forward velocity of both our Khepera and the Pioneer robot instead of giving it the fixed velocity as in Sec. 2. A robot controlled by the learning procedure of eq. 16 will move forward for some time and then reverse its velocity and so on, the distances covered by this erratic motion depending on the strength of the noise and the learning rate in a systematic way. The most interesting property of this controller paradigm however is observed if the robot collides with some obstacle. In this case the noise (difference between true and target velocity) is largely increased which leads to a very rapid relearning of the parameters c and in particular H such that the velocity of the robot is reversed almost immediately. In this way our learning dynamics may be said to generate an explorative behavior of the robot with a sensitive reaction to perturbations from the environment. Due to the sensitive reaction of the neuron to "slip and friction noise" our robot survives in nearly arbitrary environments without getting stuck in corners or at other obstacles. Moreover, learning is found to be extremely fast and permanently alert while under stationary environmental conditions it is convergent and reproducible.

The observed properties may be considered in a more general sense as finding a dynamical relation to empty space. In empty space the only information is from internal sensors (not present in the Khepera robot) and the wheel sensors feeling the motion of the robot. Now assume the unlearned controller generates a stochastic motion. The information the robot will gather is that the world is invariant w.r.t. translations, rotations, and time reversal. In this sense we might say the space is not empty but full of symmetries. An environment related behavior in empty space is defined by (dynamical) relations to these symmetries. The two possible relations are either (i) to obey the symmetries which in the present case means to stay in the state $v = 0$ or (ii) to break the symmetries which is what the robot learns to do under the above learning dynamics. The robot arrives at executing dynamical patterns (search patterns) in space.

4 Including vision

The ultimate goal of our approach is the realization of the scenario of emerging robot behavior described in Sec. 1. We are currently doing experiments with a Pioneer robot including the camera into the SM loop. Our vector of sensors values x now is written as

$$x = \left(v_l, v_r, s_1, \ldots s_K\right)$$

where v_l, v_r are the velocities of the wheels measured by the wheel sensors and $s = \left(s_1, \ldots s_K\right)$ is the vector of the pixel values of the camera where $s_i \in \mathbf{R}^3$ in the case of a color camera. There is a lot of additional problems of a more technical nature which are mainly based on the longer delay times in the SM loop due the time regimes of the underlying Saphira system and the video grabber. Another point is the complex input space of the video image.

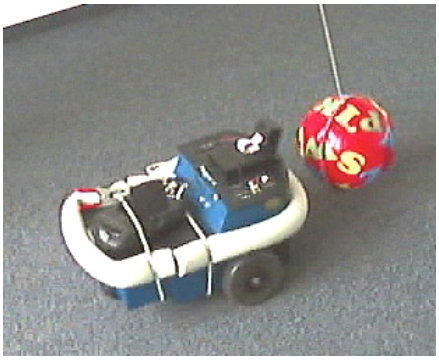

Figure 5: Typical situation of the Pioneer robot in visual contact with a ball. The controller initially is in the tabula rasa condition. Over a time of several minutes the robot gradually begins to move, to develop a sensorial contact and later on to react sensitive to the movements of the ball. In this initial phase the ball is suspended. After some time the robot is also able to follow the freely moving ball.

4.1 Preprocessing steps

So in order to simplify matters a little we do some preprocessing of the images. One is the the classification of each pixel if it has the color of the ball or not. This binary image is then scaled down to a 32x32 gray-valued image ($s_i \in \mathbf{R}$) which is fed directly into neuron 2. The other preprocessing step is the calculation of a velocity vector of the ball in this down-scaled image. We use here the relative changes of pixels in two consecutive images. We consider the change in pixel value as a kind of charge and calculate the dipole moment of this charge distribution. This dipole moment can be considered as the velocity vector $g \in \mathbf{R}^2$ of the ball in the image plane of the camera where $g_1(g_2)$ is the vertical (horizontal) component of this velocity. One advantage of this method, against simply using the movement of the center of gravity, is that if the ball leaves the field of view the moment becomes immediately zero even if there are still parts of the ball visible.

4.2 The controller

The controller consists of two neurons, neuron 1 controlling the forward velocity in the SM loop closed over the wheel sensors of the robot as described above. Additionally to the input v into this neuron we add the component g_1 of g which contains the information about the difference between the velocity of the ball and that of the robot in the direction of the forward velocity of the robot. Neuron 2 controls the turn velocity of the robot and sees the vector s of raw pixel values.

4.3 Learning

Our model assumption is that the velocity g_2 does not change over a time τ. Hence the prediction error is given by

$$E_\tau(t) = \left(g_2(t+\tau) - g_2(t)\right)^2 \tag{17}$$

Depending on $g_2(t)$ and τ the error is getting large if the ball will leave in the time τ the image plane of the camera. In order to get the gradient of this error as a function of the turn velocity we might again modulate the output of the neuron by some periodic oscillation. This has proven not very feasible in the present case since the motions of the ball are too fast as compared to the period of the oscillations so that the conditions for the applicability the learning rule eq. 7 are not well fulfilled.

We therefore use a different trick for the evaluation of the gradient. We observe that g_2 is (essentially) proportional to the turn velocity y of the robot, i.e. we put $g_2 = \mu w$ and learn μ as

$$\Delta\mu = -\eta\left(\mu w - g_2\right) w$$

with the ball at rest. In this way any trial movement of the ball can be translated into a virtual motion of the robot. In a coordinate system which is rotating with the robot the ball will appear as moving more or less randomly. The idea then is to use these fluctuations of the ball velocity and translate them into virtual perturbations of the turn velocity. This virtual modulation of w at time t is obtained as

$$s(t) = g_2 - \mu w \tag{18}$$

and the gradient of E_τ is

$$\frac{\partial}{\partial w} E_\tau \simeq \int_{t-T}^{t} s(t)\, E_\tau(t)$$

with T chosen empirically.

Putting the pieces together we have the following setting. The controller consists of the two neurons

$$\begin{aligned} y_1 &= \tanh\left(c_{11} v_{in} + H\right) \\ y_2 &= \tanh\left(\sum_{i=1}^{K} c_{2i} s_i\right) \end{aligned} \tag{19}$$

where $v_{in} = (\alpha g_1 + v)$ and $v = (v_l + v_r)/2$ as seen by the wheel sensors. The output y_1 (y_2) is the new target value for the forward (turn) velocity. The learning rule is

$$\begin{aligned} \Delta c_{11} &= \varepsilon_1 \xi^2 \left(c_{11} - 2 y_1 c_{11}^2 v_{in}\right) \\ \Delta H &= -\varepsilon_1 \xi^2 c_{11}^2 y_1 \\ \Delta c_2 i &= \varepsilon_2 \left(g_2 - \mu w\right) E_\tau(t) \left(1 - y_2^2\right) s_i \end{aligned} \tag{20}$$

where $\xi = (v - y_1)\left(c_{11}^2\left(2.0 - y_1^2\right)\right)^{-1}$ the 2.0 was introduced for numerical reasons and the update is sent through a squashing function since v_{in} can jump over orders of magnitudes.

4.4 Results

We have used the above learning rules for experiments with the Pioneer robot. The first result is that the control of the forward velocity, cf. eq. 20 works in the same way as with the Khepera robot. In fact the Pioneer executes an explorative behavior changing its velocity in a more or less regular pattern. In particular when encountering an obstacle the ensuing slip in the wheel velocities leads to a reversal of the velocity in a way that an uninformed observer would say that the robot feels the obstacles by the reaction of its wheels. We are not aware of comparable work in the literature.

We did a few experiments with the aim of camera guided ball control. However the results are not so good as with the Khepera. The problem is to obtain the gradient of the error introdued in eq. 17. With the Khepera this was done based on the response method by modulating the turn velocity. This is not feasible in the Pioneer case because of the larger delay times in realizing such oscillations as compared to the typical velocities of the ball. We presented the trick above of finding a virtual oscillation but this is a rather crude approach which works only if the statisitcs of the ball movements is appropriate which is to be guaranteed by the trainer but difficult to realize. However it is to be noted that this is a technicality of getting the gradient of the error and not a flaw of the general approach. One route of improvement is to learn the error as a function of the controller parameters and then to find the gradient by numerical derivation.

5 Conclusions

The main purpose of the paper was to show that general domain-invariant principles as formulated in the introduction can be used to derive detailed learning rules and that an autonomous robot using these rules develops domain related behaviors in a self-organized way. We understand self-organization in the strict sense as it is used in physics and was formulated in the theory of synergetic in a rather systematic way (Haken, 1987). The main point in these theories is that of modes becoming unstable leading to noise induced spontaneous symmetry breaking with the emergence of new patterns in space and time. We have seen in the present paper that the time loop error used as a learning signal provides a self-amplification mechanism which leads to the creation of new behavior modes.

The present paper has demonstrated the effects of some of the driving mechanisms to the self-organized adaptation and hence to the emergence of behavior. The aim of future work is to put the pieces together in one common picture which hopefully will show the realization of the "dream" formulated in the introduction of a robot which begins to develop activities and to take interest in the world since it has to concert its emerging activities with keeping the sensor values predictable.

We hope that the present results also do have some practical applications. On the one hand the self-organized learning algorithms may help as a domain-invariant generator for basic behaviors in the frame of behavior based robotics (Arkin, 1998) and of evolutionary robotics (Nolfi and Floreano, 2000). On the other hand they may help as auxiliary learning signals in reinforcement learning scenarios since our learning algorithms are generating domain specific learning signals in physical domains.

A last remark concerns the contingency of the behaviors generated by our bootstrap definitions. It is a further interesting outcome of our investigations that this contingency can be guided into desired directions by direct interaction with the robot providing it with the feedback so that one of the potential behaviors is amplified.

6 Acknowledgment

The work is supported by a grant by the Deutsche Forschungsgemeinschaft in the SPP -1125 (RoboCup).

References

Arkin, R. C. (1998). *Behavior Based Robotics*. MIT Press.

Cannon, W. B. (1939). *The wisdom of the body*. Norton, New York.

Der, R. (2001). Self-organized acquisition of situated behavior. *Theory Biosci.*, 120:179 – 187.

Der, R., Steinmetz, U., and Pasemann, F. (1999). Homeokinesis - a new principle to back up evolution with learning. In *Computational Intelligence for Modelling, Control, and Automation*, volume 55 of *Concurrent Systems Engineering Series*, pages 43 – 47. IOS Press.

Haken, H. (1987). *Advanced Synergetics*. Springer, Berlin.

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics*. MIT.

Pfeiffer, R. and Scheier, C. (1999). *Understanding Intelligence*. MIT Press.

R.Ashby, W. (1954). *Design for a Brain*. Chapman and Hill Ltd., London.