Teaching Robots how to become Autonomous

Nils Goerke Jan Illerhues Stefan Lohn
Division of Neural Computation, Department of Computer Science,
University of Bonn, Bonn, Germany

***Abstract*— In this paper we present a scheme to train a robot by a human teacher in a series of training phases from supervised training via correcting advises, reinforcement based training and internal critic to completely autonomous, unsupervised training. The controller of the robot consists of a network of artificial immune systems based RLAs that build up the hierarchical systemic architecture. In this paper we present new promising results from simulations and from a real robot system that has been trained with the proposed scheme.**

***Index Terms*— Systemic Architecture, RLA, Autonomous Robots**

I. Introduction

Teaching robots instead of programming them, is one of the most interesting ideas for robot control nowadays. A lot of approaches has dealt with training some of the functionalities of the robots by neural networks and other AI or machine learning methods [8]. Typically only a small fraction of the robot controller is adapted using these techniques [5]. Other approaches try to take the robot as an entire system and teach the robot some behaviours [2].

Most of these approaches use only one single learning and teaching paradigm to improve the robots capabilities. Therefore, they all have a problem to depict how the robot got the required basic abilities, and how the learning process can continue. Basic ideas for the presented work has been inspired by the idea to grow a robot up from an infantile system to a fully matured entity e.g. see [10]

The presented approach is trying to close exactly this gap. Starting with a complete “dumb“ robot, the first steps are made by extensive supervision of a teacher. During the training process the need for the teacher is reduced further and further, reaching an almost autonomous system, that is capable of improving its capabilities on its own (to some extent).

Some recent work has been published on training a robot control system with an artificial immune system based approach [4], [12] with reinforcement learning [9] is ending up with a very large system of active units, after a tremendous large number of training steps. The approach presented here, is trying to overcome this effect by using the knowledge of a teacher in the beginning. Preliminary work following the proposed approach has recently been published [6] and [7].

II. The Approach

The entire approach of making the robot autonomous by a specialised training procedure is based on three major paradigms:

- Training in selected phases
  Training the robot is performed in a set of phases which will consequently get rid of help from the teacher: from (1) supervised training, via (2) correcting advises, to (3) a reinforcement, followed by (4) building up an internal critic system to finally (5) unsupervised, critic based improvement.
- RLAs as core elements for the controller
  The controller is build up from a set of elements (RLAs) that implement a Rule-Like-Association System. Each RLA combines a condition (C) with an action (A) to be performed, and an expectation (E) that is likely to be the resulting situation. Several of these RLAs combine together to the RLA-System that allows a kind of action planning.
- Hierarchical systemic architecture
  The modular concept Systemic Architecture [3], is among other specifications, organised in layers. Each layer is representing a stage of capabilities for the robot that have been reached during the robot‘s individual development. The modules within the layers are implemented as trained RLA-systems.

III. RLA System

Rule-Like-Association Systems are a development inspired from artificial immune systems approaches. They have been successfully applied for robot control, and training robot behaviour with reinforcement based learning. They implement a special kind of knowledge base, that enables a limited kind of planning possibilities. The RLA systems gets a situation vector S defined in input or feature space, and delivers actions to be performed. The RLA systems consists of a lot of individual RLAs that can be combined together (see Fig. 1).

A. RLA Structure

Each RLA item of the RLA system is structured into three parts: a Condition part (C), an Action part (A) and an Expectation part (E). Whenever a situation fulfils the condition stored within one of the RLAs, this very RLA is turned on and the respective action (A) is activated (e.g. command for a robot). For controlling robots the association between condition and action would be sufficient. The Expectation part is dedicated to enable planning capabilities for the RLA system.

1) Condition C: The condition part of the RLA is typically implemented as a vector C defining a special position S in feature space $\mathcal{F}$. The components of this feature vector S reflect the situation the robot is in. Typically the feature

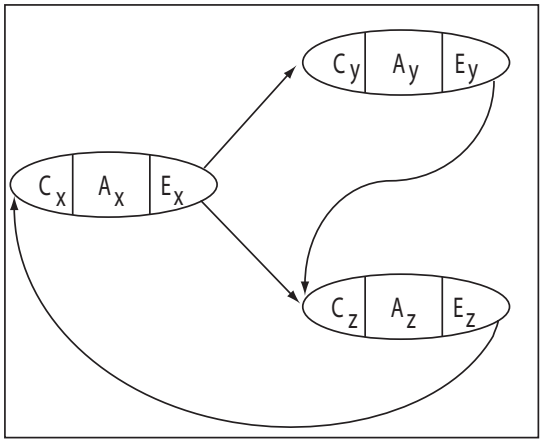

Fig. 1. Example of a RLA system with 3 RLAs. Choosing RLA X and performing its action $\mathbf{A}_x$ could lead to the situation described in $\mathbf{C}_y$, or to the situation described in $\mathbf{C}_z$. Choosing RLA Y leads to the situation $\mathbf{C}_z$. RLA Z is antecessor of RLA X.

vector S consists of low level values from the primary sensors (distance sensors ...), from virtual sensors derived from the primary sensor values (measure of parallelism,...), internal values of the robot (battery state, motor values, simulated emotions, ... and the results from information processing steps (position within a map, object detection ...). If the actual situation S of a robot matches the condition C_i of one RLA this RLA is turned on in a Winner-Takes-All manner. The RLA with the smallest distance between S and C_i is therefore the RLA with the best description of the current situation, and could be used to control the robot. If the distance between situation S and winning condition C_i smaller than a confidence radius R, the winning RLA controls the robot. If the distance is outside the confidence radius R, the C_i description of the current situation is not satisfactory. Then the current situation is declared as unknown, the motors are stopped and the learning procedure demands for a new command from the teacher.

2) Action A: The Action part of an RLA is storing the respective action that has to be activated when an RLA is turned on. The actions can be low-level (e.g. direct motor values, switch on/off, ...), mid level commands (e.g. control loops, ...), simple behaviours (e.g. wall following, ...) or complete high level tasks (e.g. build a map, find an object, ...). Typically the actions stored and activated belong to a predefined repertoire of valid actions. Note, that the action can be a complete RLA system implementing a set of complex behavioural action sequences.

3) Expectation E: The Expectation part (E) of the RLAs [4], [12] is designed to store the expected situation E_i after performing the action A_i can serve as basis for planning algorithms. In contrast to previous published work [9], [6], [7] the expectation part has been omitted in the presented work.

B. RLA Network

A set of several RLAs that work together to perform a special task is called an RLA network. Each individual RLA implements just a reaction to a condition in feature space, but the combination of several of these RLAs can establish a complete behaviour (e.g. wall following). While the RLA system is working, the situation S is compared to the conditions C_i and the RLA$_i$ triggers the respective stored action A_i. Thus, the robot performs actions and reaches different situations $S(t)$. The interconnection graph that represents the statistics of follow-up RLAs is called the RLA network: the nodes of this graph are the RLAs, the directed edges are weighted with the percentage of transitions between these two RLAs.

IV. Training in Phases

The major idea behind the phase-based training is to include different kinds of teaching and training to reduce the dependency on the teacher step by step. Therefore, we start with a completely untrained system and build up the knowledge base (RLA system) until the robot controller can act autonomously. As a remark: Bad teachers will create bad behaviour.

We have scheduled the training procedure into 5 phases that have different training characteristics. Please note, that the phases can, and will overlap during the training process, they are not strictly separated from each other.

1) Supervised Training: The teacher creates new RLAs
2) Correcting Advises: The teacher corrects existing RLAs
3) Reinforcement: The teacher gives a binary feedback
4) Internal Critic: The teacher creates the appraisal RLA
5) Unsupervised Improvement: The appraisal RLA system is used as internal critic

A. Supervised Training (1)

Whenever the controller encounters a situation where none of the available RLAs has a condition C_i similar enough to the current situation $S(t)$, the robot stops and asks the teacher for advise. The teacher gives a distinct command how to react in this very situation by specifying an action from the allowed repertoire.

Implemented as RLA approach: the teacher creates a new RLA j, the current situation $S(t)$ the robot is in, is taken as condition C_j, the command of the teacher (taken from the allowed repertoire) is stored as action A_j for this new RLA j. The expectation part E_j is left blank, and will be filled when the robot is using this very RLA later on. The next time, the robot is in this situation S, the RLA j will be turned on, and the robot will react accordingly with A_j.

Thus, a set of RLAs is created one by one and the robot is more and more capable to act without the help from the teacher.

It was found to be a good advice for the teacher to give clear, and distinct commands how to complete a given task.

B. Correcting Advises (2)

Once the RLAs are active and controlling the robot, it might turn out that some of the actions trained in phase (1) are not chosen carefully enough. In this case, there is the possibility to improve the behaviour by changing the respective action A_i, or even by completely deleting an entire RLA while the robot is running.

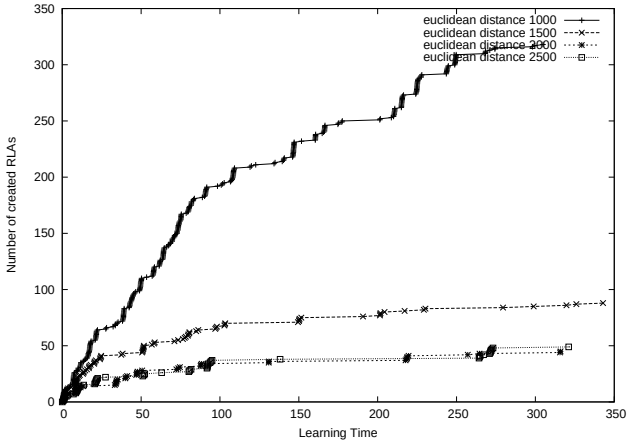

Fig. 2. Number of RLAs that were created during supervised training phase for the task of Wall-Following. The different curves are a result of a different confidence radius R during training.

The longer the training process of phase (1) and (2) has been accomplished the fewer corrections will become necessary.

Once again, it is a good advice for the teacher to have a clear vision how to complete the given task.

C. Reinforcement (3)

To further reduce the need for intervention of the teacher, phase (3) implements a reinforcement based learning [11]. The teacher has only to provide a reinforcement signal as direct feedback of the current behaviour, and the reinforcement learning will improve the RLA system through learning. To use the TD-learning scheme a state value function and a ε-greedy extension to the Winner-Take-All selection of the RLAs is implemented.

A crucial parameter for reinforcement of real world systems is the temporal aspect. The current RLA might not be the adequate RLA to blame or to reward for the current situation. Typically a set of conditions and actions (RLAs) encountered before the current situation should be rewarded. We have implemented an eligibility trace, with different learning effectivity for the most recent activated RLAs.

D. Internal Critic (4)

It is a well known fact, that reinforcement based improvement, might require a large number of reinforcement commands. To further reduce the presence of a human teacher, phase (4) implements an internal critic, to train, and improve the RLA system. The internal critic tries to mimic the reinforcement commands of the teacher, and will subsequently substitute his commands.

The approach presented here, is to implement the internal critic as a special appraisal RLA system. The condition vector of these RLAs will focus more on mid-level or high-level situations, and lesser on low-level sensory values. The actions of this internal critic will be reinforcement signals dedicated to be used as input to the reinforcement learning of phase (3).

In phase (4) the teacher puts the robot in selected, exemplary situations and creates an appraisal RLA, with a clear statement if this situation is positive or should be avoided. The action part of these appraisal RLAs is the positive or negative reinforcement signal, including the temporal eligibility parameter for the reinforcement learning. The expectation part is omitted at the present stage of development. It could be used to implement some kind of planning in reinforcement space.

E. Unsupervised Improvement (5)

Once the appraisal RLA system is up and working, its output can be used as an adapted critic to train and improve the RLA system which was set up in phase (1), (2) and (3). With this internally generated reinforcement signal, the robot evaluates its previous actions and transitions of RLAs, and alters the existing RLA network.

V. Layers of the systemic architecture

A systemic architecture is a design principle for building controllers. The architecture is structured in layers representing different hierarchies of capabilities. Several mid levels or higher levels can exist. The systemic architecture provides an information up-stream for the sensory data, and a commanding down stream for the activities.

A. Low Level

The lowest level of sensory data are the primary sensors (e.g. distance sensors, ...) they can be used as components for the features space; situation S and condition C. The lowest level actions are direct commands to the actuators (e.g. motor speed). They can be used as repertoire for the RLA action part (A). Low level expectations are typically motor or sensory situations.

B. Mid Levels

The mid level sensory items are the result of information processing modules. These results can be a part of the feature space as well: e.g. the result from a localisation process, or a local vicinity map, or the output of an object recognition process (e.g. doorfinding). The mid level commands trigger complete behaviours. They can be rather simple like wall following, or more complicated like explore the world and build a map. Mid level behaviours can be realised through complete RLA systems. The mid level expectations are typically situations like no object in front, wall on left, door next to the right, parallel to wall on left side, ...

C. Planning

The planning level within the systemic architecture is just one variant of the higher level capabilities. Within this level the upstream information is aggregated and used to implement a plan based behaviour for the system. This level is referring to the RLA system, but typically not implemented by RLAs alone. A planning algorithm which is working on the established RLA expectations (E) can be implemented to find an RLA to start from for reaching the desired expectation (e.g. door, exit from maze, ...).

VI. Implementation

A. Robot and Environment

The robot "Kurt2" (see figure 3) is an autonomous, mobile agent. A notebook is fixed onto the robot's chassis, running the robot's control software which has a direct CAN-Bus connection to the robot.

Fig. 3. The robot KURT-2, with the controlling notebook on top, and the infrared distance sensors.

The robot has a total of 18 infrared distance sensors with a range from 10cm to 60cm (with a nonlinear scaling from 550 to 70).

There are three wheels on each side of the robot which are hard connected, thus doing exactly the same; each side is controlled by one motor. The two motors are able to move the robot with a speed of up to $0.9\frac{m}{sec}$ forward and backward. Setting the motor speed of each side individually enables the robot to not only drive straight ahead, but also to make curves, u-turns and 360° rotations.

The robot control scheme is designed to work in a common office environment, with walls, hallways, rooms, doors, tables, chairs (non stationary, slow) and arbitrary moving obstacles (e.g. humans, non stationary, faster). For development and evaluation purposes we started with an artificial environment: smaller in size, no fast moving obstacles, and no legs of chairs that fit between the sensor beams and would thus be invisible for the robot.

The environment that was used for most of the experiments (see Fig. 4), is a static well controlled testing ground of $3m \times 4m$. Within the environment no obstacles are smaller than the gap between the sensors. All obstacles are stationary, or substantially slower than the robot; we can easily alter the shape of the environment by moving one of the wooden walls to a different location, even during a testrun, without harming the robot behaviour as the movements of the robot do not rely on a map, but are a consequence of the current situation.

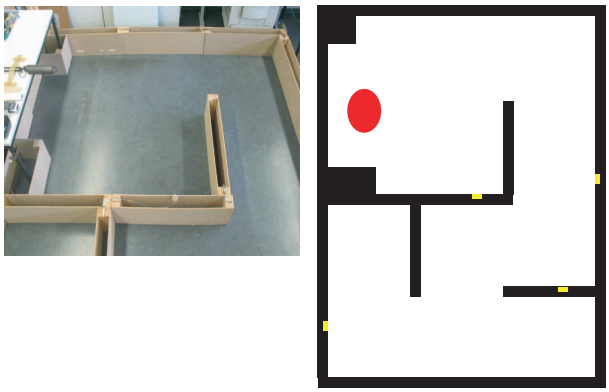

Fig. 4. The testing environment the robot is placed.

B. Feature Space

The feature space $\mathcal{F}$ is a multi-dimensional set of sensory values. Typically the feature space consists of recent, raw sensory values, of derived recent values (virtual sensors) and of values derived from past values to represent the history while the robot is moving.

1) Feature Space: Basic Level: We have chosen a 72 dimensional feature space consisting of 2 sets with 36 values of raw data and derived values; 36 recent values, 36 values from the past. Each set consists of 18 values from the infrared distance sensors (s_1, $\ldots s_18$), 4 minimum and 4 maximum sensors and 8 sensors indicating the relative orientation of the robot with respect to nearby walls. One maximum sensor gives the maximal sensor value from sensors with identical heading, the minimum sensor respectively; the 4 sensors represent the headings front, left side, right side and reverse. The orientation sensors (e.g. os_{l+}, os_{l-} are calculated as the difference between two sensors (e.g. s_1, s_2) with the same heading (e.g. left): $\text{os}_{l+} = s_1 - s_2$ if $s_1 > s_2$ and $\text{os}_{l+} = 0$ else, and. $\text{os}_{l-} = s_2 - s_1$ if $s_1 > s_2$ and $\text{os}_{l-} = 0$ else.

Thus a total of 36 values is derived from the recent situation, the second 36 values are the same items averaged over 10 seconds (equal to average over the last 1000 time steps). The basic level features space has 72 dimensions. In contrast to our earlier publications [6] and [7] no attention vector was necessary, and the Euclidean distance could be taken to compare the situation vector $S \epsilon \mathcal{F}$ with the condition C_i.

2) Feature Space: Mid Level: Since the mid level RLAs depend on already trained low level capabilities (fixed repertoire of actions) the feature space is designed accordingly. For the mid level feature space we have taken a 12 dimensional discretised space, consisting of the last 12 differing RLA actions. The distance measure between the situation S and the condition C_i is the number of actions that are different. The confidence Radius R is set rather high to 6 (out of 12 possible).

C. Teaching Interface

Since the training has to be performed in direct interaction with the robot, the interface between the robot and the human teacher was implemented as a mobile device. Thus the human teacher has a better chance to monitor thebehaviourr on site. A PDA is used as interface between the robot and the teacher. The PDA communicates with the notebook installed on the robot "KURT2" via WLAN. With this interface the user may send different information to the robot:

- Motor Commands
  The user may send the robot direct motor commands.
- Behaviours
  The user can choose the behaviour that the robot should perform.
- Reinforcements
  The user may give the robot reinforcements to help evaluate situations or actions.
- Organising Commands
  These are commands that organise learning, e.g. write the output of the RLA structure into a file.

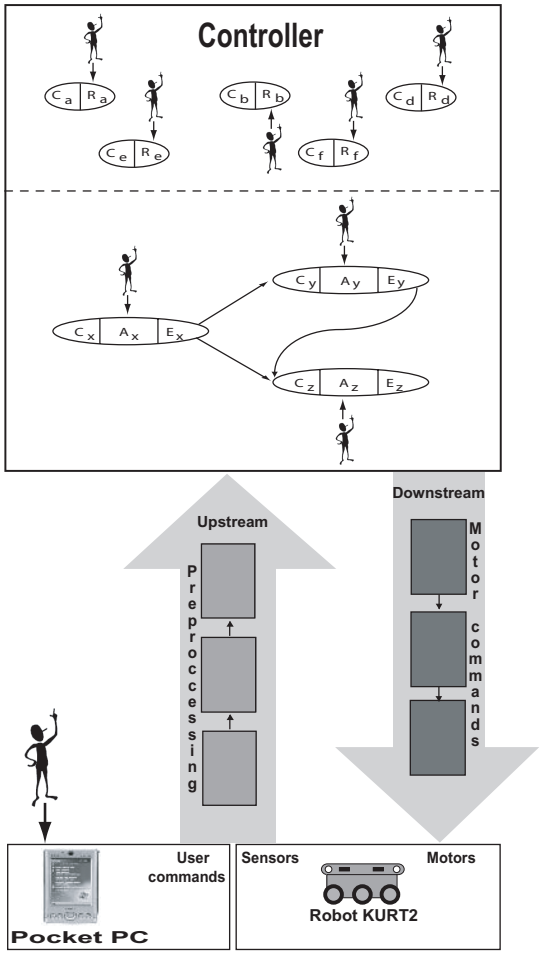

Fig. 5. The systemic architecture, with the upstream and the downstream and a RLA based controller and the appraisal RLA system on top. The PDA based user interface is depicted in the lower left corner.

VII. Results

To evaluate the capabilities of our approach we have chosen a set of with varying complexity, including the examples from our previous work [6], [7] and novel tasks that can serve as action primitives for the mid level control. Please note, that the present approach is not using any more an attention focus. All described results for the low level features were performed with a uniform 72 dimensional features space and the Euclidean distance measure for comparing S with C_i.

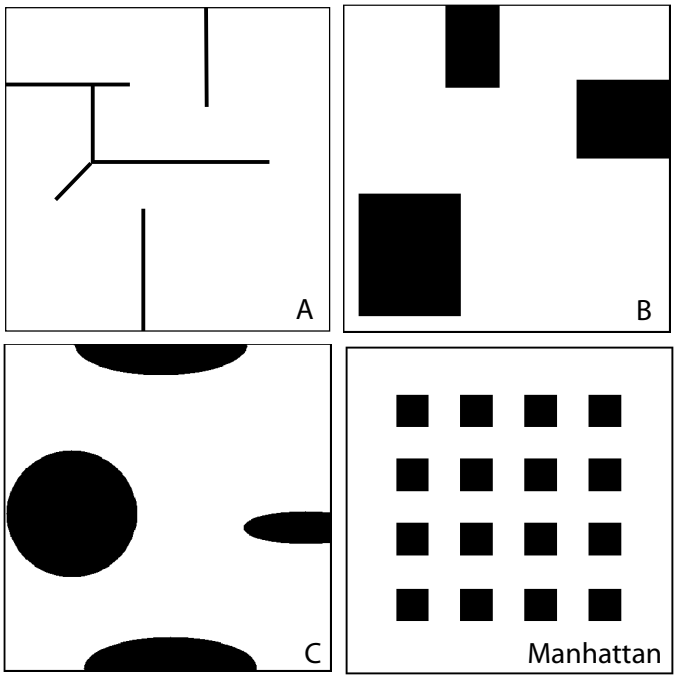

Fig. 6. The 4 test worlds: upper left: world A,usedd for training, upper right: test world B, lower left test world C, Manhattan World lower right.

1) Basic Level: The following tasks for the basic level have beensuccessfullyy trained (in a simulation and with the real robot) using phase (1), (2) and (3) of the described training procedure.

1) Fleeing behaviour including a collision avoidance:
   A behaviour that turns the robot away from an object, followed by a very fast movement (fleeing) including collision avoidance comparable to a type 3-b Braitenberg vehicle [1].
   Result: 67 RLAs, 261sec of teaching.

2) Wall Following Behaviour:
   A classical wall followingbehaviourr, coping with convex and concave corners and switchbacks. This wall following behaviour has been trained within world A 6 and has beensuccessfullyy performed a wall following in world B and C 6.
   Result: 91 RLAs, 345sec of teaching.

3) Circular shaped trajectory:
   A trajectory that implements a circle around an object that is smaller than the distance between two sensors (a wall follower would fail).

Result: 35 RLAs, 100sec of teaching.

4) 8 shaped trajectory:
   The task of performing an 8-shaped trajectory around two obstacles 7 combines a left-handed and a right-handed wall-following behaviour. See [7] forcomparisonn.
   Result: 122 RLAs, 370sec of teaching.

5) Straight On:
   Move straight on in the *Manhattan* shaped environment 6 without collisions, and stay in the middle of the *streets*. This behaviour was dedicated to serve as a movement primitive for the mid level behaviours.
   Result: 71 RLAs

6) Turn Left:
   A left turn in the *Manhattan world*, no collisions, stay in the middle of the street.
   Result: 43 RLAs.

7) Turn Right:
   A right turn in the *Manhattan world*, no collisions, stay in the middle of the street.
   Result: 21 RLAs.

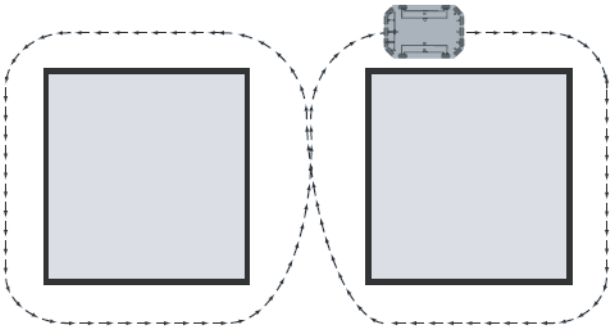

Fig. 7. The 8-shaped movement around two obstacles.

2) Mid Level: We have chosen the following three basic level capabilities to test the second level within the architecture *Straight On*, *Turn Left*, *Turn Right* These three behaviours have been trained in common in 1165sec and required a total of 135 RLAs. As an alternative these three behaviours could as well have been trained seperately, which results in a smaller learning time. To show that it is possible to train different RLA systems commonly we have performed the combined training.

The task of this mid level behaviour was to perform anarbitraryy (but fixed) trajectory through the *Manhattan World*. A total of 197 Mid Level RLA were created to learn the arbitrary trajectory in 797 seconds.

VIII. Conclusions

The presented approach provides a method to teach a robot using different training paradigms. From distinct teacher commands (creating new RLAs) via correcting advises of the teacher to a reinforcement based improvement, and further using a trained RLA based internal critic finally reaching a robot that is capable to improve its capabilities on its own. Using the artificial immune based approach we are able to go through these different training paradigms with one approach for the control structure. Even a high dimensional feature space ($>$70 dimensions) showed to be rather an advantage for learning than beeing a problem. As improvement of previous work, the attention focus could be omitted, treating all dimensions of the feature space equally. This is a simplification of the training procedure.

Within this work, it was demonstrated that the basic level capabilities could be implemented by training RLA systems, and that they can be used subsequently by the mid level RLA system as action repertoire. Thus the first step into a hierarchical learning procedure is shown to be working.

The results are promising and the authors are convinced that the presented method can be applied to a wider field than robot control alone. The final goal to get rid of the nasty *low-level* robot programming has come one step further into reach.

We do no longer programme our robot:
We teach the tasks by showing the actions
and judging the performed behaviour.

References

[1] Valentino Braitenberg. *Vehicles- Experiments in Synthetic Psychology*. The MIT Press, 1986.

[2] Ka Cheok. Gauging intelligence of mobile robots. In *Mobile Robots: Moving Intelligence*, pages 134–146. ARS: Advance Robotic Systems International, 2006.

[3] Dr. Nils Goerke. Perspectives for the next decade of neural computation. In *Proc. of NATO Advanced Research Workshop on: Limitations and Future Trends in Neural Computation (LFTNC'01)*, pages 1–9, 2001.

[4] Emma Hart, Peter Ross, Andrew Webb, and Alistair Lawson. A role for immunology in '"next generation"' robot controllers. In *ICARIS03, Artificial Immune Systems, Second International Conference*, pages 46–56, 2003.

[5] A Ijspeert, H Kimura, and H Witte. Amam 2005 research on adaptive motion in animals and machines. In *Proc, 3rd International Symposium on Adaptive Motion in Animals and Machines (AMAM 2005)*, pages III–VII. Technical University Ilmenau, Germany, 2006.

[6] Jan Illerhues. Entwicklung eines verfahrens zum beschleunigten lernen von handlungsplanungen für mobile roboter. Diploma Thesis, Dept. of Computer Science, University of Bonn, Bonn, Germany, 2006.

[7] Jan Illerhues and Nils Goerke. A human aided learning process for an artificial immune system based robot control. In *Proc. of the International Conference on Informatics in Control, Automation and Robotics, ICINCO'07, Angers, Paper ID 462*, 2007.

[8] Buchli Jonas. *Mobile Robots: Moving Intelligence.* ARS: Advance Robotic Systems International, 2006.

[9] Juergen Rattenberger, Patrick M. Poelz, Erich Prem, Emma Hart, Andrew Webb, and Peter Ross. Artifical immune networks for robot control. In *Proc Fourth Int Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, volume 117, pages 151–154, 2004.

[10] Peter Ross. Notes on the RLA network implementation. Technical report, Napier University, Scotland, 2003.

[11] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[12] Andrew Webb, Peter Ross, Emma Hart, and Alistair Lawson. Generating robot behaviours by evolving immune networks. Technical report, Napier University, Scotland, 2003.