4

EMOBOT: A Robot Control Architecture Based on Emotion-Like Internal Values

Nils Goerke
Dept of Neural Computation, University of Bonn
Germany

1. Introduction

Controlling robots and other autonomous agents has been successfully performed over a long period using a wide variety of approaches within various applications. The dilemma between using predefined methods (reliable but rather inflexible) and learned capabilities (more flexible, but sometimes inexact) is still a challenge for the field.
Designing a structure that is in principle capable of performing the desired tasks and attempting to gain this capability throughout a learning process is one of the most interesting areas in contemporary design of robot controllers (Braitenberg, 1985, and Brooks, 1985 &1986). Neural network approaches and neural learning paradigms have been widely used to implement parts of these control architectures.
The idea of generating "emotion-like" behaviour for technical agents seems to be a valuable approach to modelling more sophisticated and more flexible capabilities within robotics. An extension of the "emotion-like" approach to behaviour using the capabilities of adaptive mechanisms seems to yield a more complex class of behavioural features.
The realisations of learning emotion-driven robot controllers are not meant to compete with the established and approved paradigms in robotics and autonomous systems, but try to provide a set of "extra" features to the field (Gadanho &Hallam, 1998a,b and Cos & Hayes 2002 and Gadanho & Custodio, 2002). The author is convinced that the proposed "exotic" approach is a valuable alternative to established approaches to learning behaviour.
Since no "teacher" in the classical sense is available for the envisaged task, a reinforcement based learning scheme has been developed to alter the entries of the action selection matrix.

2. Controlling autonomous robots by selection of behaviour primitives

2.1 Different levels of control

Autonomous robots can be controlled on different levels with different scientific disciplines involved (classical control theory, neuroinformatics, biocybernetics, artificial intelligence, psychology, cognitive science). From the low-level control like motor speed, driving duration and movement direction via basic capabilities like obstacle avoidance and simple control loops to mid-level behavioural capabilities like searching, foraging map making,...

Source: Mobile Robots, Moving Intelligence, ISBN: 3-86611-284-X, Edited by Jonas Buchli, pp. 576, ARS/plV, Germany, December 2006

up to high-level behaviours that we are used to now from our domestic animals (McFarland, 1999) and that we would like to obtain for our robots.

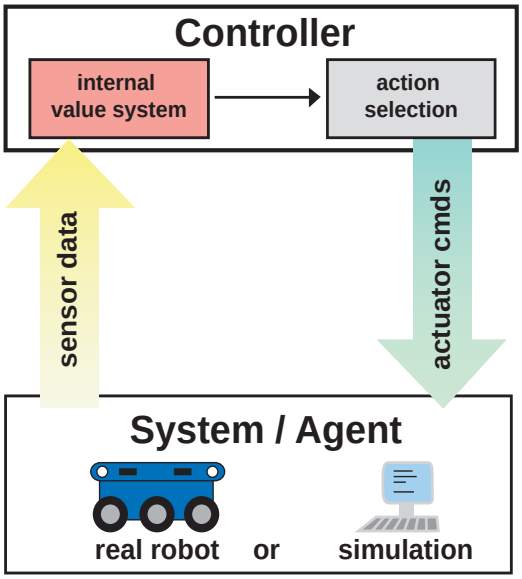


Fig. 1. Systemic architecture with the robot, the sensory upstream (left) the actuatory downstream (right), the action selection (upper right) and the internal value system.

2.2 Low-level control

The basic level of robot control is directly aligned to the physical properties of the robot system: actuators (which are mostly electrical motors) and sensors (which typically yield electrical values). The robot actuators are controlled following the solutions from control theory: typically the motor position, motor speed and the motor torques are governed using P-, I-, D- or PID-controllers. Autonomous agents are typically controlled via the speed of their wheels and the by the duration of activity. Different speed demands for the wheels will result in different turning angles, and thereby in different movement directions (drive forward, turn left, turn right, turn around, stop …).

Controlling autonomous robots just by means of low-level control is annoying and time consuming. Complex tasks can only be accomplished with a big effort.

2.3 Mid-level control 1

The next level of complexity in robot control realises basic robot capabilities that imply a minimal level of internal representation. A basic control mechanism is a feed-back loop that uses the sensor values to calculate the motor activities as set points for the low-level control. Typical examples are: reactive movements like Braitenberg type vehicles (e.g. 3b), wall-following behaviour, reactive obstacle avoidance, a "follow me" behaviour, random-move until a special condition is reached...

Each of these mid-level capabilities can be implemented into a controller as behaviour primitive of its own. A clever choice of these behavioural modules can lead to a more

complex and smart behaviour of the robot. Still, controlling an agent by these mid-level capabilities can be very laborious.

2.3 Mid-level control 2

The really interesting part in robot control is dealing with the combination of lower levelled capabilities. The mid-level control is combining and sequencing several of the capabilities that have been realised in the lower levels of the control hierarchy to gain more complex behaviour.
This stage of robot control requires an internal memory structure, some planning capabilities and an explicit task to be performed. Dependent on the task and the effective situation the robot is in, different behaviour primitives (mid-level 1) can be triggered, stopped, or modified to complete the desired task.
Typical examples for mid-level-2 capabilities are: search for an object, search for a location, foraging behaviour, map making, escape from a labyrinth, kick a ball into a predefined direction...
All of these mid-level 2 behaviours depend on other mid-level capabilities, and are goal directed behaviours. Most nowadays robotic tasks, and most autonomous robot jobs, can be realised by combining mid-level-2 behaviours. The clever combination of these capabilities is still considered "artwork" and has not yet become a closed theory.

2.4 High-level control

High-level behaviours for autonomous robots are typically aligned with behaviours that we are aware of from our day-to-day life. We might know this kind of behaviours from our domestic animals and from our own (human) behaviour. We certainly want our robots to gain these behaviours, but today we are still lacking these capabilities in modern autonomous systems. Most high-level control tasks have linguistic commands that deal with symbolic information and with complex tasks. The following sentences might serve as examples for the kind of high-level control the robotic community is looking for: "Search for the third room on your left.", "Go and count the number of chairs.", "Deliver the mail.", "Look out for unusual objects.", "Tell me what you have found in room 3."
High-level control covers a wide range of capabilities that we would like to have for our robots. Although today only a few convincing implementations can be found, the progress achieved during the last years is tremendous and increases further. The author is convinced, that in a few years truly high-level control for robots will be available.

2.5 Controlling by Action Selection

One way to control a robot making benefit of all described levels of control is to combine the realised capabilities from the different levels by explicitly selecting them. Other approaches with action selection for robots can be found in (Arkin, 1998) and (Burghouts et al., 2003) and (Velásquez, 1997). Each capability is treated as an individual behaviour primitive which can be an action from one of the control levels, or a clever combination of several of these actions. The action selection mechanism triggers one of these behaviour primitives with respect to the condition the robot is in. The action selection can depend directly on the sensory state of the robot (e.g. mid-level control), can depend on some internally calculated results (e.g. world model, a map), will probably depend on the given task for the robot (e.g. goal directed behaviour), or can depend on the specially designed set of internal values.

The task of mapping the state of the robot onto one of the available behaviour primitives can be realised in various ways: e.g. neural networks, action selection matrix, mixing of experts, rule based, artificial immune system approach, fuzzy controllers, expert systems and a wide variety of further approaches.

3. The need for an internal value system

eside pure reactive behaviours autonomous agents, like robots can be governed by a set of internal values, giving these robots a sort of internal state. This internal state can serve as a data basis (even growing data basis) for guaranteeing autonomy in planning and behaviour for the robot. The more complex this internal data basis is, the better the involved planning algorithms can determine and calculate the robot actions, or sequences of actions.

Most robot control systems rely on some sort of world model as basis for the planning of actions, action sequences or even basic movements (see Murphy, 2002 or Siegwart and Nourbakhsh, 2004). The internal world model can be extremely basic and the resulting control mechanisms thus even "hard-wired" (e.g. Braitenberg (1985) vehicles type 3b), the world model can be more complex including topologic assumptions (e.g. maps) and calculations that are based on the monitored actions of the robot actuators (e.g. dead reckoning, SLAM). The world model can be extremely complex including a physic based simulation of the robot, the environment and/or even other non stationary entities (e.g. moving obstacles, multi robotics, robot soccer…). Beside the simulation of the world, internal values from the robot system can be used to model and represent the robot state: e.g. battery condition, speed of the wheels, duration of movement, and percentage of task fulfilled.

In addition to values and conditions that are related to physically measurable events of the robot and the environment, a special set of values can be defined, that represent internal values of the robot control system, e.g.: the condition of the task execution, the overall performance of the system, a time and/or distance dependent measure… This set of internal values is denoted Internal Value System (IVS). The action selection mechanism depends not only on physically measurable effects, but additionally on the internal value system of the robot.

4. Internal values inspired by psychology (emotion-like values)

4.1 Inspired by psychological concepts

Using the idea of an internal values system that is not only dependent of physical conditions but in addition on some-how meta-states of the robot the IVS has been designed. These internal values are designed in accordance with psychological terms that we (human beings) associate with "*Drives*" and "*Emotions*", (Damasio, 1994) and (Velásquez 1997). These internal values do not realise real "*Drives*" and "*Emotions*", but the robot controller is designed in such a way, that the robot shows a behaviour that seems to be governed by "*Drives*" and "*Emotions*" (see Canamero, 1997 for an alternative approach). The psychological concepts used, help the human operator to design and judge the resulting behaviour of the robot system. The robot behaviour is imitating emotionally driven behaviour. Yet we emphasise that at no stage in the proposed approach we claim to realise "*Drives*" or "*Emotions*" as they are assigned to living entities

4.2 "Drives"

The internal values that are aligned with the psychological term "Drives" are implemented in a way that tries to match with these psychological terms. We call these internal values "Primary Internal Values" because they depend directly and only on the results of the sensory upstream of the systemic architecture and on time. Drive values are bounded within the range from ***-1*** to ***+1***, with a desired value of ***0*** which is indicating that the robot is in a balanced internal state.
We have implemented four primary internal values:

- **Fatigue:**
  Movement makes the fatigue value rise linearly, resting makes it decrease linearly, but steeper than rising. The virtual binary sensor "*Active*" indicates if the robot is doing something or not.

$$Fa_{t+\Delta t} = Fa_t + \frac{\Delta t}{120s} * Active + \frac{\Delta t}{40s} * (1 - Active) \tag{1}$$

- **Hunger:**
  Hunger is linearly increasing in time, and reduced by a fixed amount through "Food". A virtual binary sensor indicating "*Food*" has been implemented.

$$Hu_{t+\Delta t} = Hu_t + \frac{\Delta t}{60s} * (1 - Food) + \frac{2}{3} * Food \tag{2}$$

- **Homesickness:**
  The longer the robot is not at the home position, the more the homesickness value rises, linearly in time. A linear but faster decrease occurs when the home position is reached. "*Home*" was realised via a virtual binary sensor indicating if a special area within the environment has been reached.

$$Ho_{t+\Delta t} = Ho_t + \frac{\Delta t}{240s} * Home + \frac{\Delta t}{20s} * (Home - 1) \tag{3}$$

- **Curiosity:**
  The more similar sensory input the robot encounters, the more the curiosity drive rises. Novel input patterns make the curiosity decrease. We have implemented a virtual sensor "*Dist*raction" measuring the flow of information through the sensors to guide the curiosity value. The curiosity rises linear with time and is decreased, if something interesting occurs. As long as the distraction is above a fixed threshold, indicating something interesting, the curiosity decreases linearly.

$$Cu_{t+\Delta t} = Cu_t + \frac{\Delta t}{60s} * (1 - Distr) + \frac{\Delta t}{1s} * Distr \tag{4}$$

4.3 "Emotion*s*"

The secondary set of internal states is called "*Emotions*". These internal values depend mainly on the satisfaction of the primary internal values (i.e. their closeness to 0) and are roughly aligned with the respective psychological term "*Emotion*" that we are familiar with. Still, we are not claiming to model the psychology. Our robot does not have any real "*Emotions*" (and will probably never have). These internal values reflect internal states that we have designed to show a specific behaviour. The selected four "*Emotions*" are four of the five universal emotions (Damasio, 1994), only sadness was replaced by boredom, since this emotion better fits the desired and implemented functionality.

- **Fear:**
  The internal value Fear increases with rising Fatigue and rising Homesickness values and decreases with falling Curiosity values as well as with time.

$$Fe_t = Fe_{t-\Delta t} + \frac{Fa_t}{15} + \frac{Ho_t + |Ho_t|}{20} + \frac{|Cu_t|}{30} - dec_{Fe} \tag{5}$$

- **Anger:**
  All of the four drives influence the emotion of Anger. Hunger causes it to increase strongly, Homesickness raises it a little, and even Curiosity increases it a little bit, while Fatigue decreases the Anger moderately, as it does with time.

$$An_t = An_{t-\Delta t} + \frac{Hu_t}{10} + \frac{Ho_t}{35} + \frac{Cu_t + |Cu_t|}{100} - \frac{Fa_t}{30} - dec_{An} \tag{6}$$

- **Boredom:**
  Boredom is increased if the drives are negative, and is decreased if the drives have positive value.

$$Bo_t = Bo_{t-\Delta t} + \frac{-Hu_t - Ho_t - Cu_t - Fa_t}{30} - dec_{Bo} \tag{7}$$

- **Happiness**:
  The internal value of Happiness increases when the drives are close to the balanced state (i.e. close to 0). High Hunger or Homesickness values decrease the Happiness more than Curiosity or Fatigue do. The other "Emotions" influence the Happiness as well: The lower the values for Fear, Anger and Boredom are, the more the Happiness value raises with time.

$$\begin{aligned} Ha_t = Ha_{t-\Delta t} &- \frac{(Hu_t + |Hu_t|) + (Ho_t + |Ho_t|) + 2An_t + 2Fe_t}{40} \\ &- \frac{(Cu_t + |Cu_t|) + (Fa_t + |Fa_t|) + 2Bo_t}{100} + dec_{Ha} \end{aligned} \tag{8}$$

4.4 The EMOBOT internal value system

The internal value system is part of the controller. It is designed to reflect internal states of the robot to support the control and selection processes. The internal values are a set of real valued components derived from the results of the sensory upstream and the inherent rules that govern the dynamics of these internal values. Within the IVS we have implemented two classes of values: primary ones ("Drives") that depend directly on time and the sensory upstream and secondary ones ("Emotions") that depend mainly on the primary internal values and on each other.

5. The EMOBOT control architecture

5.1 Systemic Architecture

The proposed control architecture for the robot is based on the ideas of systemic intelligence to span the bridge between sub-symbolic representation of knowledge (neural networks,

fuzzy control, fuzzy logic, rules, differential equations...) and symbolic described capabilities (goals, reasoning, behaviour, intention...). The intelligent behaviour shown by living systems and some technical artefacts is neither the consequence of the symbolic description of their tasks, nor the consequence of the sub-symbolic representation of information. It is postulated that intelligent behaviour arise only if the design of the system has been performed adequate (Kintzler, 2002 and 2003).
The systemic architecture is a design concept for building and structuring the architecture of controlling systems (Goerke, 2001 and 2003). Based on experience and knowledge from neurosciences, neurocomputing, psychology and cognitive sciences, a systemic architecture features a set of capabilities and principles to be portable onto a technical artefact. The design principles governing the systemic architecture are not meant to be strict, but are understood as an overall guideline for the design of a systemic architecture controller.
Following these design-principles, a systemic architecture consists of a hierarchical structured network of simple and if possible independently adaptable building blocks. Each building block consists of modules which process information and learn capabilities. Thereby the layers refer to the layers below by using the provided information and the range of functions and capabilities immanent in these underlying layers.

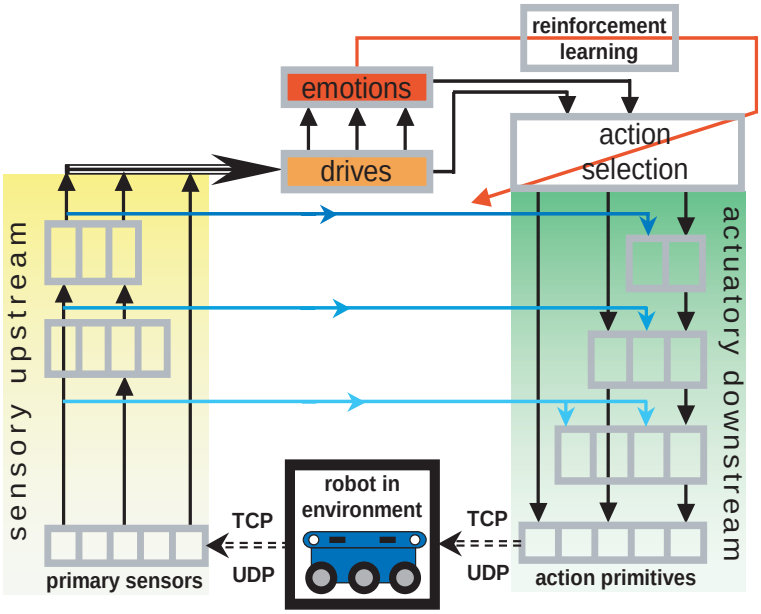


Fig. 2. Systemic architecture with the robot, the sensory upstream (left) the actuatory downstream (right), the action selection (upper right) with the reinforcement learning module and the internal value system with drives and emotions.

The presented approach implements the controller by a set of modules with distinct functionality. These modules are organised in layers, where higher layers are designed to implement higher functions. In addition, the architecture is organised into two main branches: One branch going from low level sensory data, upwards to higher sensory capabilities (sensory upstream, left branch in Fig. 1 and 2). The other branch is directed from

high level, highly sophisticated control schemes, down to low level motor functions (actuatory downstream, right branch in Fig. 1 and 2).

5.2 Mixing controllers and switching controllers

Within the low-level control and the mid-level-1 control layers of the architecture several modules have been realised. Each of these modules implements a behaviour primitive. This assembly of modules is called a "bank of behaviour primitives". The controller is now selecting and combining the different behaviour primitives to generate the steering commands for the actuatory system, with respect to sensory values and the to the internal value system "Drives" and "Emotions" (Goerke & Müller, 2003).
Three possible variations of implementing the controller are reasonable: switching controller σ,α, switching controller σ,Ω and mixing controller μ.
Switching controller:
The switching controller switches explicitly between the different behaviour primitives. If the switching controller is positioned before the bank of behaviour primitives, it will be denoted σ,α; if it is positioned behind the bank, it is denoted σ,Ω.
For the σ,α type, only one of the behaviour primitives have to be active at a time(faster). The disadvantage is that the results of the other behaviour primitives can not be taken into account for the switching process. For the σ,Ω variant all behaviour primitives have to be calculated (resource consuming) before the decision (switching) takes place.
Mixing controller*:*
The mixing controller combines (or mixes) the different results from the bank of implemented behaviour primitives to compute the control command for the robot. The combination can be realised as a liner-combination, or as a non-linear mapping from behaviour primitive outputs onto motor values (Skarmeta et al, 1999). The abbreviation for a mixing controller is μ**.** Mixing controllers are always positioned behind the bank of behaviour primitives, and therefore always of type μ, Ω.

5.3 Action selection

The task of selecting an adequate action from the available behaviour primitives with respect to the actual state of the robot is called action selection. This is reduced to the mapping from the continuous state space of the internal values to the discrete set of behaviour primitives. Several approaches for this mapping operation have been tested and implemented: e.g. action selection matrix, neural network (type MLP).

5.4 Action selection matrix

Within the presented approach, the EMOBOT controller is realised as a switching controller of type σ,α. With respect to the state of the internal values, the action selection chooses one of the available behaviour primitives. Therefore the space of possible states has been partitioned into disjoint areas. To each of these areas a behaviour primitive is assigned to (action selection matrix).
For this action selection, we have quantised each of the real valued "Drives" into the four regions: very low (-1.0, -0.5); low (-0.5; 0.0); high (0.0, +0.5); very high (+0.5, +1.0) see Fig. 2 left part. Thus, the four dimensional state space of the four "Drives" is segmented into 4x4x4x4 = 256 disjoint areas. Each of these areas is assigned to one of the behavioural primitives (see Fig. 3 right part and Fig. 10).

Depending on the state of the "Drives" from the internal value system, the switching controller triggers one of the behavioural primitives.

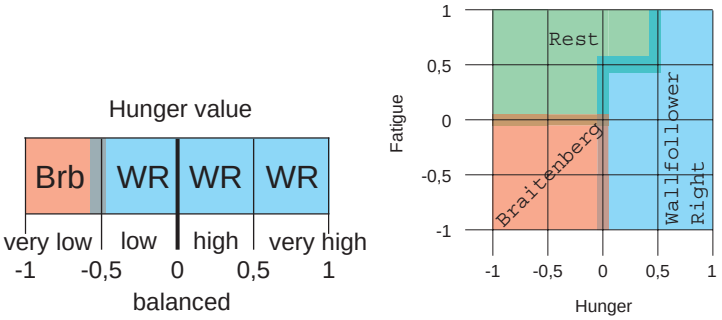


Fig. 3. The quantised internal values (e.g. Hunger and Fatigue) are quantised into four segments; each segment is assigned a behaviour primitive (action selection matrix).

5.5 Learning action selection

Learning the matrix means to assign adequate actions to the 256 state boxes. Since we have no real teacher available we can not use a supervised learning paradigm. Therefore, we have implemented a reinforcement based scheme to change the action (behavioural primitive) within each box.

Four paradigms to obtain an adequate reinforcement signal have been realised:

- **Reinforcement calculated by an objective function:**
  An external objective function is calculated to gain a reinforcement signal: e.g. distance to obstacles, time until a special situation is reached, number of turns, ratio of forward to turning movements, number of times a special position has been visited…
- **Reinforcement provided by a human observer**:
  A human operator observes the behaviour of the robot system and judges this behaviour with a reinforcement signal (reward and punishment). This reinforcement signal is extremely depending on the human observer, it is not reproducible nor is it stationary, as the mood the human operator is in, might change during the experiment.
- **Reinforcement calculated by a trigger signal or trigger event**:
  The reinforcement signal is triggered by an external event, e.g. a special position within the environment, a special obstacle configuration, or a special sensory input.
- **Reinforcement based on secondary internal values (state of "Emotions"):**
  The overall condition of the "Emotions" is used to generate the reinforcement signal. As long as the robot is in a "good emotional state" a positive reinforcement is generated to further foster the most recent behavioural primitives, and a negative reinforcement is generated as soon as the "condition of the robot" becomes unwanted (threshold) to punish the actions that has caused this.

5.6 Adaptive matrix entries

To make the action selection matrix learnable, we have extended each entry of the matrix into a priority vector, containing one entry for each of the possible behavioural primitives. The resulting matrix contains one priority value for each available action, for each of the possible IVS situations.

Now it is possible to use the priority value within the matrix for a winner-takes-all strategy, choosing the behavioural primitive with the highest priority. As a second variant the priority value can be used to determine the selection probability for choosing the respective action. The winner-takes-all strategy implements a pure exploitation, and the probability view implements the exploration mechanism of reinforcement learning (Sutton & Barto, 1998) and (Kaelbling et al., 2002).
The reinforcement learning is now capable of changing the priority values for each matrix entry with respect to the reinforcement signal. Punishment of a specific behavioural primitive is realised as decrease and rewarding this very action as moderate increase of the respective priority value (see Fig. 11).

5.7 Adaptive regions of interest

As an alternative to the fixed tiling of the state space into unit size boxes, the partitioning can be made adaptable. We propose to learn this partitioning with a neural network, e.g.: Self Organising feature Map (SOM), Learning Vector Quantisation (LVQ), Regional and Online Learnable Fields (ROLFs), Multi-SOMs.

5.8 Neural network based controller

As a first step towards a more general realisation, we propose a neural network, type Perceptron (P) or Multi Layer Perceptron (MLP). The input to the neural net, denoted with **X**, is the vector of the internal values. The output vector **Y** of the neural network is now post-processed with a winner takes all, *1 out of n*, decision process to form the binary decision vector ***DV***, with only one component being active $dv_w = +1$ and all other components being inactive $dv_i = -1$.. The active component w indicates the winner, and thus the behavioural primitive hat has been chosen for activation. Thus, the network is switching between the available actions with respect to the sensor and internal values, for further details see Goerke et al, 2004.

6. Realisations of the EMOBOT approach

To test and evaluate the approach we describe three realisations of the EMOBOT with different complexity. A simulation of an EMOBOT within a grid world, and with reduced capabilities. A simulation of an EMOBOT in a continuous valued environment. An implementation of the EMOBOT control approach for a real robot system within a well controlled testing environment. All realisations were focused on different aspects of the EMOBOT system and are therefore deliberately different.

6.1 Grid world EMOBOT

The simple EMOBOT realisation is based on a 26 x 22 grid world, with a boundary and several obstacles. Each cell of the grid can monitor and remember how often the robot has visited this very site. The grid based robot is one square of the grid in size; its orientation is limited to multiples of 45°. The robot sensors can monitor the three directly adjacent squares in front of the robot (yielding 8 possible obstacle situations).The sensory system can register from the grid world how often one of the eight surrounding squares have been visited by the robot up to now $M_{x,y}$. The actions the robot can perform within each time step are either to move one square forward, to turn 45° and to move, and to perform a 135° escape turn. Two basic behaviours have been realised for the grid based EMOBOT realisation. Both schemes are completely deterministic and have no stochastic components.

- **A pure Reactive behaviour R:**
  Depending of the actual obstacle information (one of eight possible situations), the reactive controller is switching between the different actions (see Fig. 4).
- **An eXplorative behaviour X**:
  Knowing how often the robot has visited the surrounding squares, the robots turns via the cell visited the least, and moves one step forward onto it, thus further exploring its world.

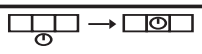
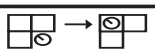
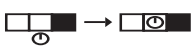
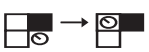
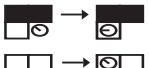
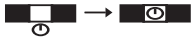
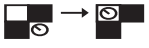
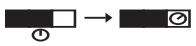
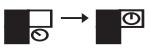
Sensory values left	front	right	Action	$t_i \rightarrow t_{i+1}$	$t_i \rightarrow t_{i+1}$
0	0	0	forward		
0	0	1	forward		
0	1	0	left,forward		
0	1	1	left,forward		
1	0	0	forward		
1	0	1	forward		
1	1	0	right,forward		
1	1	1	escape turn		

Fig. 4. The reactive behaviour for the grid based EMOBOT implementation.

The switching controller is choosing the explorative behaviour if the curiosity value is rising above a certain threshold. For the grid world based robot, the curiosity has been model different to equation (4) taking the discovered places and two control parameters p and q into account:

$$C_{i+1} = C_i + p + q \cdot \frac{C_i}{M_{x,y}} \tag{9}$$

To get a feeling what kind of behaviour the robot is establishing, the two behaviour primitives were tested separately. Within a second test phase the two actions were combined using the curiosity value to decide between the two behaviours: Reactive behaviour (R), and eXplorative behaviour (X). Using only the explorative movements the robot is exploring the whole world (or 90% of it). Depending on the actual starting position the robot will pass the different squares of the grid world different times (counted within $M_{x,y}$.). To reach 90%= 377 out of the possible 419 squares the explorative control needs just 550 steps (averaged over all 419 starting positions). Since no real searching strategy has been implemented, 550 required steps is regarded as reasonably good. The author is aware of exploration algorithms that perform a lot better; but the efficiency of the exploration is not the scope of the presented research.
Equipped with only the reactive behaviour the robot would never be able to explore the whole world, because the implemented control tends to establish stereotypic movements (see Fig. 5). In detail, only three stable trails emerge from the chosen combination environment and reactive behaviour (A,B,C). Depending on the starting position one of the three trails is reached, and the robot's movement is stuck within that trail.

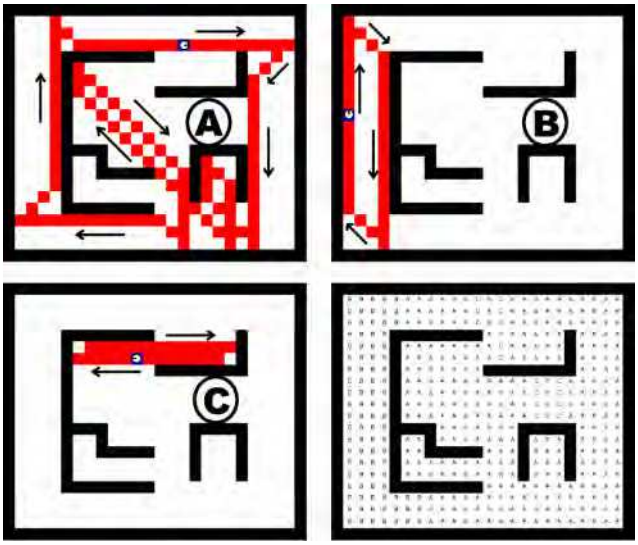


Fig. 5. The three trails (A,B,C) that result when using the reactive behaviour. The dependency of the trail from the starting position is depicted in lower right.

When both control mechanisms are combined with respect to curiosity and the threshold, the robot consequently shows a complex mix of both behaviours. With respect to the starting position and to the parameters (p and q) that model the curiosity the final behaviour has been investigated in extensive simulations (see Fig. 6). For each combination of p,q values all starting positions within the grid world have been tested and the resulting behaviour (Trail A,B,C or eXplorative behaviour) monitored and visualised (see Fig. 6). It is easy to see, that a variety of different behaviour can be reached by changing the Curiosity parameters p and q.

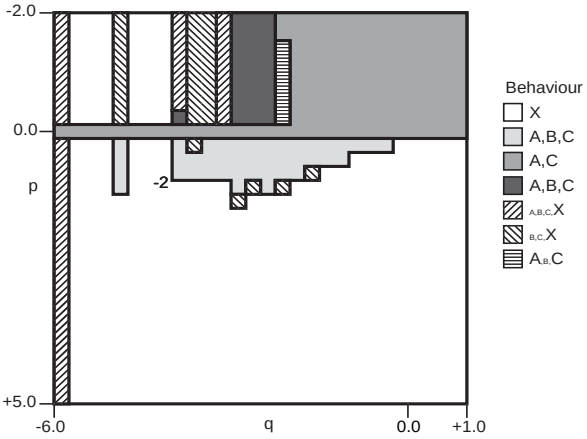


Fig. 6. The resulting behaviour with respect to Curiosity control parameters p,q:. The behaviour $_{B,C}X$ for example indicate mostly eXplorative movements with some starting positions leading to the trails B and C.

6.2 Real valued EMOBOT

The second implementation is the simulation of EMOBOT in a more realistic environment (real valued, several arbitrary shaped objects, Fig. 7), with more behavioural primitives (12) and with more internal values (5) as input for the action selection. The simulated robot has five distance measuring sensors (three front, two rear, see Fig. 7) and has the capability to move forward and to make turns.

The internal values modelled for this EMOBOT realisation are:

- **Boredom**: rising with time, decreasing with changing sensor values.
- **Curiosity:** rising with time, decreasing when new terrain is explored.
- **Home sweet Home**: small near the home position, increasing with Euclidian distance.
- **Joy:** rising in time, decreasing when a special sensory pattern is encountered.
- **Tiredness**: decreasing when the robot is inactive, moderate increase while turning strong increase when the robot is moving forward.

Fig. 7. The real values testing environment with some obstacles and the **H**ome position, including the robot with the five sensors.

The implemented behaviour primitives are taken from low-level and mid-level control schemes:

1. *Sit and rest*: just stop, rest and look for a while.
2. *20° left turn.*
3. *Forward*: move forward 4 times the robot size.
4. *Escape movement*: 180° turn and run.
5. *Step back*: move backwards, 3 times the robot size.
6. *Search exit*: turn right until free space in front.
7. *Random walk*: turns and moves randomly.
8. *Side step right:* 90°turn right, 2 forward, 90° turn left.
9. *20° right turn.*
10. *Search object*: move forward until front sensors detect something;
11. maximal 10 times the robot size.

12. *Wander*: move and turn right right-before bumping
13. *Braitenberg vehicle*: type 3b, obstacle avoiding

To investigate the capabilities of this approach, several series of simulations have been conducted; part of the results can be found in (Goerke et al., 2004). From the status of the internal value system a reinforcement signal is computed that serves as basis for training the action selecting neural network. The neural network is trained by using the reinforcement-generated artificial teacher vector, and a temporal discounting factor for recent situations as extension to the learning rule backpropagation of error. The internal value system, the neural network □□□-emotional controller, the bank of 12 behaviour primitives, and the simulated robot within an environment are implemented. Extensive simulations yielded a feasible way to obtain a variety of different and interesting behaviours for the EMOBOT.

6.2 EMOBOT implemented on a real robot system

The EMOBOT approach has been implemented to control a real robot using the set of 8 internal values (the 4 primary "Drives" and the four secondary "Emotions"). The action selection is based on the matrix with 256 adaptable entries. Each entry is triggering one of 10 implemented behaviour primitives (4 mid-level-1, 5 mid-level-2 and sit-and-rest).
The robot is equipped with 2 motors, each driving 3 wheels on one side of the robot. The size of the robot is 35cm wide, 45cm long and 35cm high. The robot is equipped with 6 ultrasonic distance measuring sensors, 18 infrared distance measuring sensors, and two ambient light sensors, (see Fig 8).

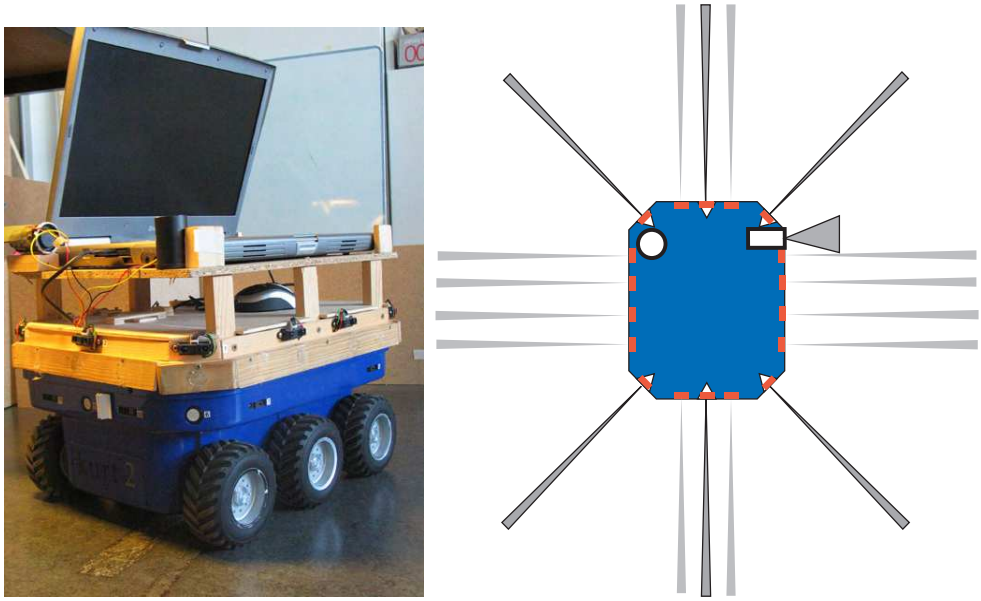


Fig. 8. The real robot with the controlling notebook on top (left picture) and the layout (right picture) for the distance and ambient light sensors (circle, rectangle).

In this work, only the infrared distance sensors and the ambient light sensors were used. The infrared sensors have a reliable range between 10cm and 80cm, the width of their beams

is 6-8cm. Obstacles closer than 10cm are not reliably detected by the infrared sensors, so that the controller needs to provide for maintaining a distance to obstacles greater than 10 cm. Objects that are smaller than the gap between the sensor beams (e.g. legs of tables) are invisible to the robot as well. One ambient light sensor heads to the right side of the robot (white rectangle in Fig. 8), the other to the ceiling (white circle in Fig. 8).
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 (humans). 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 depicted in Fig. 9 was used for most of the experiments is static, well controlled testing ground of 3mx4m. 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 test run, 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. The environment contains several passages with straight walls, 13 +90° corners, three sharp -180° corners and one -90° corner. In addition, four light sources have been placed in the environment for being visible by the ambient light sensor (lighter rectangles in Fig. 9). One place in the environment is lit (oval structure in Fig. 9) for being detected by the upward directed light sensor. This special location is regarded as home position for modelling homesickness.

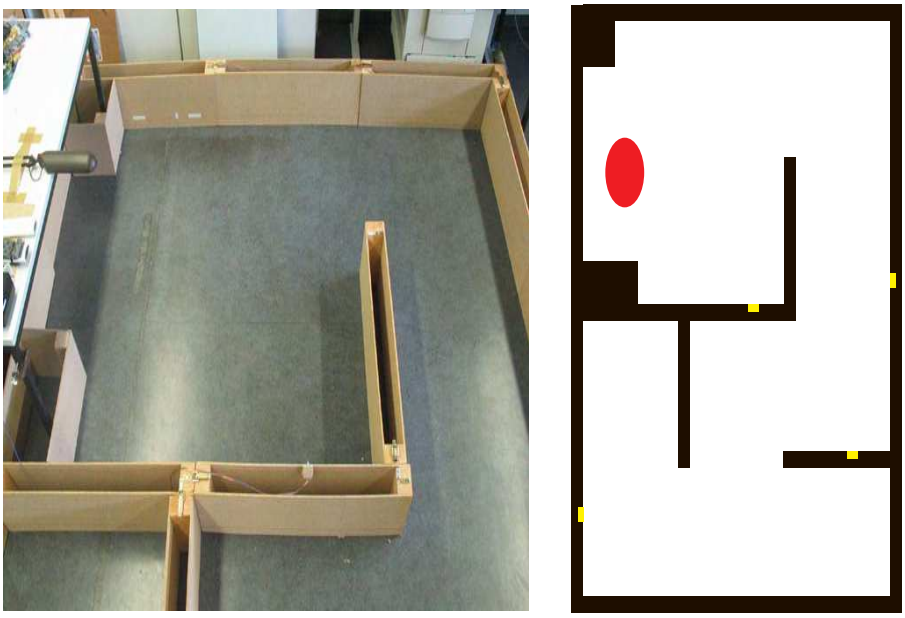


Fig. 9. The environment (picture) and a schema (right) with the home position (oval area) and the 4 locations of Food (lighter rectangles within walls).

The behaviour primitives implemented are only exemplary realisations that have been chosen with respect to the real robot system from the different controlling levels:

- **Low-level control**:
  Forward moves are performed with 0.2m/sec and turning with 96°/sec. The seven implemented behaviour primitives are: Move forward, Move backwards, Turn left, Turn right, Drive a curve to the left, Drive a curve to the right, **S** "Sit and Rest".
- **Mid-level-1 behaviours**:
  WL: Wall Following left wall, **WR** Wall Following right wall, **BrB** Braitenberg movement, type 3b, **Pos** Position the robot in front of an object.
- **Mid-level-2 behaviours**:
  R Random Move, **Esc** Escape, **Sea** Search a Passage, **Rou** Round-Trip, **Or** Orbit around an object.

	Ho	vL: very Low				L: Low				H: High				vH: very High			
Fa	Hu Cu	vL	L	H	vH	vL	L	H	vH	vL	L	H	vH	vL	L	H	vH
vL	vL	Brb	Brb	WR	WR	Brb	Brb	WR	WR	WL	WL	WR	WR	WL	WL	WL	WR
	L	Brb	Brb	WR	WR	Brb	Brb	WR	WR	Rnd	Rnd	WR	WR	WL	WL	WL	WR
	H	Esc	Esc	WR	WR	Rnd	Rnd	WR	WR	Rnd	Rnd	WR	WR	WL	WL	WL	WR
	vH	Esc	Esc	Esc	WR	Esc	Esc	Esc	WR	Esc	Esc	Esc	WR	WL	WL	WL	WR
L	vL	Brb	Brb	WR	WR	Brb	Brb	WR	WR	WL	WL	WR	WR	WL	WL	WL	WR
	L	Brb	Brb	WR	WR	Brb	Brb	WR	WR	Rnd	Rnd	WR	WR	WL	WL	WL	WR
	H	Esc	Esc	WR	WR	Rnd	Rnd	WR	WR	Rnd	Rnd	WR	WR	WL	WL	WL	WR
	vH	Esc	Esc	Esc	WR	Esc	Esc	Esc	WR	Esc	Esc	Esc	WR	WL	WL	WL	WR
H	vL	R	R	WR	WR	R	R	Pos	WR	R	R	Pos	WR	WL	WL	WL	WR
	L	R	R	WR	WR	R	R	Pos	WR	R	R	Pos	WR	WL	WL	WL	WR
	H	R	R	Pos	WR	R	R	Pos	WR	R	R	Pos	WR	WL	WL	WL	WR
	vH	Esc	Esc	Esc	WR	Esc	Esc	Esc	WR	Esc	Esc	Esc	WR	WL	WL	WL	WR
vH	vL	R	R	R	WR	R	R	R	WR	R	R	R	WR	R	R	R	WR
	L	R	R	R	WR	R	R	R	WR	R	R	R	WR	R	R	R	WR
	H	R	R	R	WR	R	R	R	WR	R	R	R	WR	R	R	R	WR
	vH	R	R	R	WR	R	R	R	WR	R	R	R	WR	R	R	R	WR

Fig. 10. Action selection matrix with 4x4x4x4=256 segments for the 4-dimensional state space of the 4 "Drives", **Fa** Fatigue, **Hu** Hunger, **Ho** Homesickness and **Cu** Curiosity. Each segment contains one of the 10 implemented behaviour primitives.

The learning action selection was first tested with the real robot system using just the internal value Hunger and the two behaviour primitives Braitenberg (Bbr) and Wallfollowing Right (WR) using automatic reinforcement, and a moderate exploitation. The

priority values were initialized using random values between 95 and 100 for each action in each box. Now, every 30 seconds the current action in the current box was punished, while after every discovery of Food the action in the box encountered 1.5 seconds before, was rewarded. After a run of 20 minutes, the values of the actions had changed to favour Braitenberg only in the case of a very low Hunger value and Wallfollowing Right elsewhere. In Fig. 11 the initial priority values and the final depicted during the reinforcement based learning process. The resulting 1-dimensional action selection matrix is shown below for a winner takes all strategy.

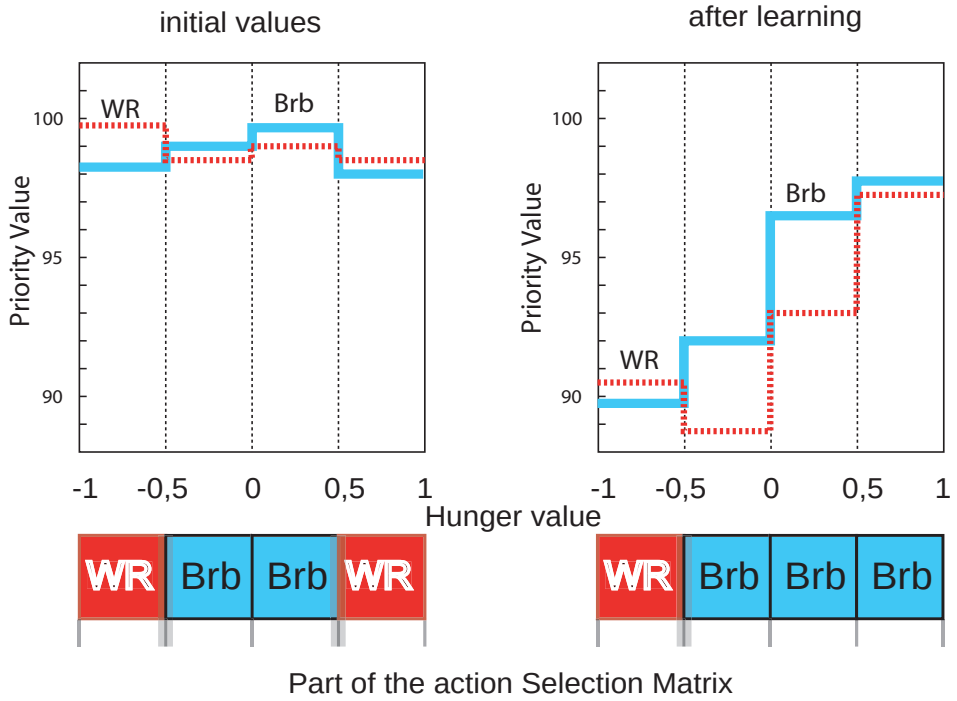


Fig. 11. The priority values are changed during the learning process using the reinforcement signal. Depicted is a part of the action selection matrix, where the active behaviour changes from Wall Following right (WR) to Braitenberg movement (Brb).

By this, the learning mechanism for matrices has been established. Now the secondary internal values can be used to generate a reinforcement signal and thus fill a randomly initialised matrix with action selection preferences. Further work is necessary to define adequate dependencies between the internal value system ("Emotion" values) and the reinforcement signal. Experiments conducted with the robot and with a human operator yielded changing action selection priority values and thus changing action selection matrices. One could easily "train" the robot to avoid certain actions, just by pure negative reinforcement (approx 5-10 minutes of training). More complex behaviour could be obtained by longer and more sophisticated reinforcement signals coming from the human operator. Due to the large action selection matrix (256 boxes) and the far larger priority value matrix (256 x 10) a completely human trained action matrix robot was omitted.

7. Conclusion

Getting autonomous robots to do the things we want them to do is still a challenge. Even the definition of parameters for a given controller type is very hard or realise for interesting robotic tasks. Designing controllers that are easily configured in an adequate way is far more complicated. The idea to make a controller learn an adequate set of parameters and functions for a given task is not completely new, but still not solved sufficiently. Biological entities demonstrate that it is in principle possible to realise such a system.

Based on these ideas, a hierarchical robot control structure, with three main information processing branches: a sensory upstream, an actuatory downstream, and a controller was developed. The controller consists of two main units: an internal value system (IVS) and a learning action controller. The internal values (Drives, Emotions) are fed with the results from the sensory upstream. Several (17) primary and virtual sensor modules have been implemented within the different layers of the sensory upstream. The internal values are the decision basis for the learning action controller. The action controller activates the different action modules within the hierarchy of the actuatory downstream. We have implemented a total of 14 action modules that realise basic, simple and complex motor actions, including several senso-motoric closed loop controlled behaviours.

The developed internal value system is aligned with psychological terms like "Drives" and "Emotions", without attempting to model these psychological concepts. The implemented internal values, like "Hunger", "Fatigue", "Fear", "Curiosity" or "Homesickness" are just meant as labels to indicate their specific functionality.

The action selection unit has been implemented as a matrix implementation of a switching controller. The content of the matrix is a result of pre-design and subsequent learning. Training the action selection is performed using part of the internal values as a reinforcement signal, as well as human teacher generated reinforcement.

The different realisation of the EMOBOT approach, with different kind of robots (grid world based, real value based and a real robot system) and with different set of internal values ("Drives" and "Emotions") show that the idea of controlling a robot with a set of meta-values, that are not directly connected to the outer world is a feasible approach to generate a complex behaviour.

For a given configuration of the behaviour primitives and the action selection and the internal value system it is difficult to predict the overall behaviour of the robot system. The other way round, to construct a special set-up with the goal to create a specific behaviour is far more complicated, but on at the same time far more interesting.

The results obtained, and partially presented within this work show clearly that the approach of learning hierarchical action selection based on internal values is a valuable way of finding robust robot controllers. Although a lot of scientific questions still remain unanswered we believe that the results presented are encouraging for the future.

8. References

Arkin, R. C. (1998). *Behaviour based robotics*. Bradford Books, MIT Press, Cambridge, Massachusetts.

Braitenberg, V.(1985). *Vehicles: Experiments in Synthetic Psychology*, The MIT Press, Cambridge, Massachusetts; London, England.

Brooks, R. (1985) A *robust layered control system for a mobile robot*, Technical Report: AIM-864, Massachusetts Institute of Technology, Cambridge, MA, USA.

Brooks, R. (1986), A *Robust Layered Control System for a Mobile Robot*, IEEE Journal of Robotics and Automation, Vol. 2, No. 1, March 1986, pp. 14-23;

Burghouts, G.J.; Heylen, D.; Poel, M.; op den Akker, R.; Nijholt, A. (2003). An Action Selection Architecture for an Emotional Agent. In *Proc. of 16th International Florida Artificial Intelligence Research Society Conference*, St.Augustine, Florida, USA.

Canamero D (1997) Modeling motivations and emotions as a basis for intelligent behavior , In *Proc. First Int. Symp. on Aut. Agents, AA'97*, The ACM Press.

Cos I, Hayes G (2002) Behaviour Control Using a Functional and Emotional Model. In *From animals to animats 7: Proc. 7th International Conference on Simulation of Adaptive Behaviour. pp. 226-227, MIT Press, Massachusetts, USA.*

Damasio, A.R. (1994). *Descartes' Error: emotion, reason and the human brain Robot*, New York, USA: Picador, 1994

Gadanho, S.C.; Hallam, J. (1998a). Emotion-driven Learning for Animat Control. In, *From animals to animats 5: Proc. 5th International Conference on Simulation of Adaptive Behaviour*, pp. 354-359, MIT Press, Massachusetts, USA.

Gadanho, S.C.; Hallam, J. (1998b). Exploring the Role of Emotions in Autonomous Robot Learning. In *AAAI Fall Symposium - Emotional and Intelligent: The tangled knot of cognition*. Also DAI Research Paper 909, University of Edinburgh.

Gadanho, S.C.; Custodio, L. (2002) Asynchronous Learning by Emotions and Cognition. In *From animals to animats 7: Proc 7th International Conference on Simulation of Adaptive Behaviour*. pp. 224-225, MIT Press, Massachusetts, USA.

Goerke, N. (2001). 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)*, pp. 1-9, Siena, Italy.

Goerke, N., (2003). Systemic Intelligence: Methods for Growing up Artefacts that Live', In: *Proceedings of the Int. Joint Conf. on Neural Networks, IJCNN03*, Portland, Oregon, USA, 2003.

Goerke, N.; Müller, J. (2003), Emotions are driving a robot, In: *Proceedings of the 8th Engineering Application of Neural Networks Conference (EANN03),* pp. 56-63, University of Malaga, Malaga, Spain..

Goerke, N.; Henne, T.; Müller, J. (2004) *Neural networks for the EMOBOT robot control architecture*, Neural Computing and Applications, Vol. 13, No. 4, pp. 299-308, Springer, Heidelberg.

Kaelbling, L.P.; Littman, M.L.; Moore, A.W. (1996). *Reinforcement Learning: A Survey*, J. Artificial Intelligence Research, **4:**237-285, Morgan Kaufmann.

Kintzler, F. (2002) *Entwicklung, Implementation und Test einer Kontroll-Architektur für aufwachsende Animaten,* Diploma-Thesis, Dept. of Computer Science, University of Bonn: Bonn, Germany.

Kintzler, F. (2003). Layout and first tests of a systemic architecture for growing up artefacts that live, In: *Proceedings of the 8th Engineering Application of Neural Networks Conference (EANN03),* pp. 72-79, University of Malaga, Malaga, Spain.

McFarland, D. (1999). *Animal Behaviour*. Addison Wesley, Longman, Harlow.

Murphy, R. (2000). *An Introduction to AI Robotics: Intelligent Robotics and Autonomous Agents*, The MIT Press, Cambridge, Massachusetts; London, England.

Siegwart, R.; Nourbakhsh, I.R.. (2004) *Introduction to Autonomous Mobile Robots*, A Bradford Book - The MIT Press, Cambridge, Massachusetts; London, England.

Skarmeta, A. G.; Barberà, H.M.; Alonso, M.S. (1999) *Learning Behaviour Fusion in Autonomous Mobile Robots*. ESTYLF'99, London, England.

Sutton, R.S. Barto, A.G. (1998). *Reinforcement Learning*, The MIT Press, Cambridge, Massachusetts; London, England.

Velásquez, J.D. (1997). When Robots Weep: Emotional Memories and Decision-Making, In *Proc. 15th National Conference on Artificial intelligence*, AAAI Press, Madison, Wisconsin, USA.

Mobile Robotics, Moving Intelligence

Edited by Jonas Buchli

ISBN 3-86611-284-X

Hard cover, 586 pages

Publisher Pro Literatur Verlag, Germany / ARS, Austria

Published online 01, December, 2006

Published in print edition December, 2006

This book covers many aspects of the exciting research in mobile robotics. It deals with different aspects of the control problem, especially also under uncertainty and faults. Mechanical design issues are discussed along with new sensor and actuator concepts. Games like soccer are a good example which comprise many of the aforementioned challenges in a single comprehensive and in the same time entertaining framework. Thus, the book comprises contributions dealing with aspects of the Robotcup competition. The reader will get a feel how the problems cover virtually all engineering disciplines ranging from theoretical research to very application specific work. In addition interesting problems for physics and mathematics arises out of such research. We hope this book will be an inspiring source of knowledge and ideas, stimulating further research in this exciting field. The promises and possible benefits of such efforts are manifold, they range from new transportation systems, intelligent cars to flexible assistants in factories and construction sites, over service robot which assist and support us in daily live, all the way to the possibility for efficient help for impaired and advances in prosthetics.

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821