Adaptive LEGO Robots.
A Robot=Human View on Robotics.

Henrik Hautop Lund, Claus Bjerre, Jes Hvoldal Nielsen, Michael Nielsen, Kasper Støy

LEGO Lab
University of Aarhus
Aabogade 34
8200 Aarhus N., Denmark

hhl@daimi.au.dk, nimrod@daimi.au.dk, hvoldal@daimi.au.dk, mic@daimi.au.dk, u930413@daimi.au.dk
http://www.daimi.au.dk/~hhl
http://legolab.daimi.au.dk

Abstract
In many applications, robots were viewed as a machine. This has resulted in interaction and actuation which is characteristic for machines. When constructing adaptive LEGO robots, we take another view, namely that the robot should resemble a human (or a biological creature) rather than a machine. This has implication on the interaction, the actuation and the control of the robot. Here, I will describe how the robot-as-human approach is investigated in a number of LEGO Mindstorms robot applications. These include making facial expressions, which allows a LEGO robot to express internal "moods", and thereby we might achieve a better human-robot interaction. Another application is the Adaptive LEGO Pet Robot. The Adaptive LEGO Pet Robot's control is based on a modular behaviour system, where a number of the modules are evolved neural networks. Further, the Adaptive LEGO Pet Robot has a number of internal drives such as restlessness and hunger, which allow the robot to react on the internal drives. The human-robot interaction is facilitated by allowing the human to train the LEGO pet robot (rather than to program the robot) to make associations between spoken words (via speech recognition) and evolved behaviours. The Adaptive LEGO Pet Robot is an example of scaling up evolutionary robotics to complex behaviours by combining evolutionary robotics with behaviour-based robotics.

1. Introduction
Often, when constructing robots, engineers and roboticists tend to take the view that robots should be like machines. This view seems to stem from the view that a robot should be a device on which we, as users, possess full control. We can call this a *robot=machine view* on robotics, since it leads roboticists to think of robots as machines, because, traditionally, we have full control over machines. The robot=machine view influences both the way in which we project robotic systems, the way we build robots, and the way in which we program robots. When projecting a new robotic system, we think of a new machine-application and a scenario where we, as users, have full control over each single action of the robot(s) in the system. For example, we would view an assembly line as a suitable place for a robotic system, because we have full control over the surrounding environment and can make a robot with machine-like actions to perform sequences of actions that are fully controlled. With knowledge about everything that happens in the surrounding environment (e.g. speed of the assembly line, the shape and size of the objects on the assembly line, etc.), we can use inverse kinematics to calculate how the robot shall move in order to fulfil the task that we have in mind. In scenarios like this, industrial robots (which are traditionally instances of the robot=machine view on robotics) are successful. However, as mentioned above, the success is fulfilled when we have full control over the environment. Unfortunately, this is often not the case. A simple, but nevertheless very illustrative, example is Chaplin's character sketch of a machine-like worker at an assembly line, who cannot keep up his performance when the speed of the assembly line suddenly increases. Chaplin's machine-like "robotic" behaviour is pre-programmed to have a specific, machine-like sequence of actions, and it cannot change adequately in response to the change in the environmental conditions.
Another view, which we would like to promote here, is the *robot=human view* on robotics[1]. According to this view, robots should be like humans (or biological systems) rather than like machines. If we take the robot=human view, we are forced to make robots that are totally different from the traditional industrial robots, since humans do not have machine-like actuators, machine-like sensors, machine-like interaction with the environment. Further, humans are not pre-programmed to go through a specific sequence of actions. Instead, humans have soft motions and their control is based on the interaction with the surrounding environment. Therefore, when taking the robot=human instead of the robot=machine view on robotics, we have to change both the way we project, build and program the robot systems. For instance, we should try to model the way humans interact with the surrounding environment through soft (and not machine-like) actuation, through vision and sound, etc. Importantly, we learn from humans that adaptations to changes in the surrounding the environment are essential for the "success" of humans. In Chaplin's example, a more human-like scenario would allow the worker to change his behaviour when the environment changes instead of having to go through the same sequence of actions over and over again.
In order to study the robot=human view on robotics, we have performed a number of robotic experiments. First, we look at the interaction between robot and human. A meaningful interaction between an autonomous robot and a human demands that the human can interpret the intentions and, for example, the internal state of the autonomous robot. In section 2, we look at

[1] We could also call this a robot=biological system view on robotics. The only reason not to do so is that the term robot=human seems to give better associations.

perception and understanding of emotions, and communication of emotions in a LEGO Mindstorms robot model with facial expressions and sound processing. Secondly, in section 3, we look at the interaction between human and a more complex robot model, namely an adaptive LEGO pet robot. The adaptive LEGO pet robot is allowed to learn from humans and the situations in the surrounding environment, which causes emergent behaviours to emerge in the pet robot. The work illustrates the possibility to combine evolutionary and behaviour-based approaches to achieve more complex control systems capable of self-adaptation and learning. Section 4 discusses the work on the initial robot=human experiments, and presents some future directions of research in this field. It should be noted that the work presented here is work in progress, which means that part of the work is presented as discussion (especially section 2).

2. An Empathy and Sympathy Robot Model

When designing robots from the viewpoint of robot=human, the interaction between the robot(s) and the user becomes essential. We learn from the study of humans that the ability to express and communicate internal states is an essential characteristic. There exists a vast number of definitions of emotions, and we will not go into a discussion of these definitions here, but rather provide one definition, simply for clarifying what we are talking about in the following, without trying to claim that this is the most appropriate definition. Here, we define emotions as a product of internal state and they are communicated to the external with (dramatically) changes of external behaviour. Theory tells us that humans are able to perceive and understand emotions, and to produce emotions. Perceiving and understanding emotions is known as *empathy*, and communicating emotions is known as *sympathy*. Communication of emotions seems to be independent of the semantic level related to human language. For example, the carrier frequency of sound can be used to communicate and understand emotions. Also, the repetition frequency of sound can be used to express emotions. Imagine an agent with a small beeper. When emitting beeps with low frequency, humans interpret the agent as being in a quiet state, while when emitting beeps with high frequency, humans interpret the agent as being in an anxious state.

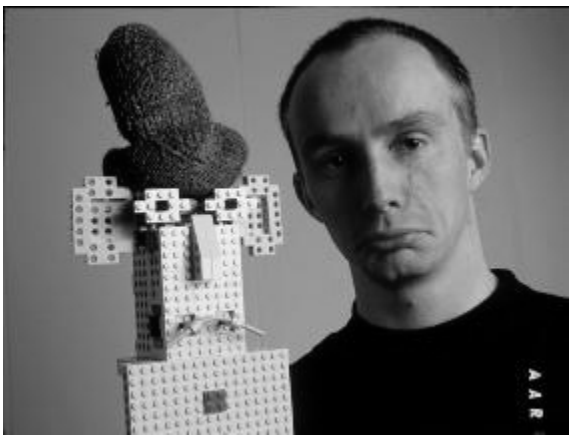

Figure 1. Expressing internal states with a LEGO Mindstorms robot. ©Volker Steger, 1999.

We designed a LEGO Mindstorms robot to study empathy and sympathy as a model for facilitating robot-human interaction. The robot is designed to have facial expressions by moving eyes and mouth, and can express simple sounds (the robot is shown on figure 1). For *sympathy*, the simple robot uses light inputs (two light sensors) and an (internal) drive unit telling the robot to be active or rest. The light sensors register the frequency of light changes in front of the robot (e.g. waving with a white piece of paper) and can go from 0 (no activation) to 1 (high activation). The activation of the drive unit can go from 0 (rest) to 1 (active). The motor behaviour of the robot (sounds and facial expression) is a function of the activation of the two units. If the light and drive units have opposite phase, the robot responds with a high frequency of beeps (e.g. the human user continuously waves to the robot when it wants to rest), and when in the same phase, the robot responds with a low frequency of beeps.

In the same way, *empathy* is modelled by not interpreting semantics, but rather by looking at amplitude and frequency of the communication from the user as a model of the emotional state of the communicator. In the simple robot model, the robot has a light unit and an empathic unit. The empathic unit activation is an interpreter of the emotional state of an external communicator. For instance, an agent (child, adult, other robot) emits light (e.g. by waving white paper). The empathic unit interprets this light. If it reaches a high level, it will be categorised as an aggressive state of the communicator.

3. An Adaptive LEGO Pet Robot

The example of empathy and sympathy in a robot showed one aspect of the robot=human view on robotics, but it did not provide many different kinds of distinctive behaviours in the same robot. Our study on an adaptive LEGO pet robot is moving in the direction of providing a number of distinct behaviours in the same robot that should interact with humans.

3.1 Related Pet Robot Work

Our work is not the first to use a pet robot as a case study. The project with the most notable similarity with our project is the artificial emotional creature project by Takanori Shibata and Kazuro Tanie [5] who are focusing on human machine interaction, where the machine is a pet-like robot in a fur costume, such as a seal or a cat. The robots have quite complex mechanics making it possible to interact with humans in various ways. Emotions are displayed using the actuators in a way that would be expected from a real animal of the same kind. Motivation of the behaviours may be generated through competition among the emotions of the pet. It seems to have the basic (or primary) emotions, such as love, happiness, anger, fear and sadness. These emotions are considered as innate emotions. Until now the learning capabilities of these creatures are quite limited, though some secondary emotions seems to be acquired through learning from interaction with the environment and other creatures.

Furby from Tiger Electronics, which is currently a huge commercial success, is a small furry robot that is able to talk and wriggle. It has a number of touch and tactile sensors that makes it sensitive to human interaction. It has no learning capabilities, but a simulated 'learning' algorithm ensures that the capabilities of the Furby gradually increase.

The Sony pet robot [6] has until now, almost entirely focused on the impressive hardware platform, only recently has the work been broadened to implement complex adaptive software. Currently it is only implemented with a rudimentary behaviour based system.

The Creatures software game from Cyberlife Technology consists of Albian agents called Norns, who are created using the opposite approach and are entirely software based [1]. Their 'brains' are quite complex adaptive behaviour based systems that generate own internal motivations and learn from mistakes. The software system of the Norns is very interesting and combined with a adequate hardware it could be an example of an

embodied, situated, emergent and seemingly intelligent robot that is motivated by its own feelings and interaction with the environment.

3.2 Adaptive LEGO Pet Robot Overview

We put emphasis on allowing the user to be able to develop (parts of) the pet robot when interacting with the pet robot. Again, this approach arises from the robot=human view on robotics. When a child (or human in general) is interacting with a pet animal, the child is not thinking of the animal as a machine with a fixed program, and hence is not considering it possible to *program* the brain of the pet (as with the robot=machine view on robotics). Instead, the child might be able to *train* the pet to do some specific tricks and obey some specific commands. In our adaptive LEGO pet robot project, we are therefore trying develop a robot system that allows such an interaction.
In general, our pet robot project aims at illustrating how the evolutionary and behaviour-based approaches can be combined to achieve more complex control systems capable of self-adaptation and learning, as indicated by Lee, Hallam, and Lund [3]. Our first steps towards creating a believable autonomous, dog type, pet robot and a number of experiments with this pet robot show the adaptation and learning ability of the agent.
Basic behaviours are created using evolution of neural networks in a simulated environment that runs on a parallel evolutionary system, the behaviours executed by the evolved neural networks are tested on a physical robot, and implemented in the behaviour based system. The more simple behaviours, such as tail waggling for example, are hand coded The behaviour based system makes the robot able to use and combine the evolved behaviours according to the current situation, which depends on sensory input, what it has learned in its life so far, and the internal states, such as hunger and restlessness. Thus learning from humans and the situations in the surrounding environment causes emergent behaviour to emerge in the robot, something that is not normally done by behaviour-based control systems. Figure 2 shows a schematic drawing of the robot.

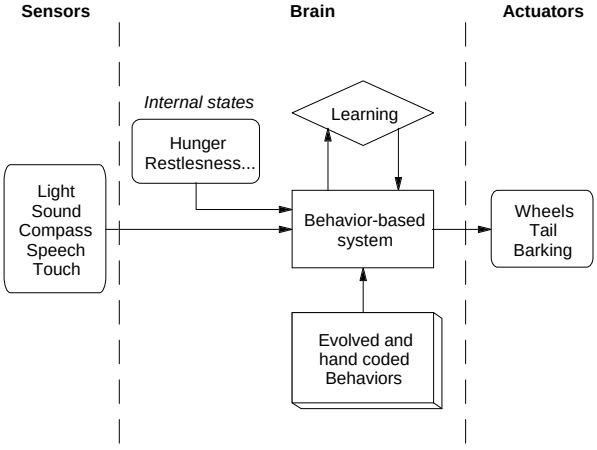

Figure 2. Schematics of the pet-robot. ©LEGO Lab, 1999.

3.3 Integrating New Hardware with LEGO Mindstorms

We choose to use the LEGO Mindstorms robotic kit as the hardware platform for the adaptive pet robot, because this robotic kit allows fast prototyping, flexibility in the robot morphology (which we view as essential to robotics [4]), and easy integration of new hardware. For instance, for the adaptive pet robot, we made a number of new sensors available for the LEGO Mindstorms robotic kit, including digital compass sensor, bend sensor, directional hearing sensor, and speech recognition (through communication with host computer).
The LEGO Mindstorms control unit (RCX) is constructed to work with the specially designed LEGO sensor types. The LEGO sensors can be read in two modes, active and passive. Active sensors are sensors that require current, such as light sensors and rotation sensors. Passive sensors require no supply current to generate the response, a touch sensor, which basically is a resistor that can be either connected or disconnected, is a passive sensor. When an input port on the RCX is configured for an active sensor the voltage is a little higher than when it is configured for a passive sensor, about 7 volts for active and 5 volts for passive, furthermore it is possible to draw a limited current, about 15 mA, from a sensor input that is configured active, and thus it is possible to drive some simple electronics with the input port.
Our wish was to use digital sensors with RCX. It should be possible to use these digital sensors like the other sensor types on the RCX, meaning that the digital equipment should be plugged directly onto the RCX. The converter must be a digital to analogue converter; the analogue signal received by the RCX is then again converted to a digital signal inside the RCX.
All the sensors act as resistors. The LEGO Dacta touch sensors have an infinite resistance when not pressed, and a fixed, low, resistance when pressed. The temperature sensor has a variable resistance, depending on the temperature. This means that a converter must transform a digital signal into some resistance.
The general idea is to have each bit from the digital equipment open a transistor connected with a resistor. If the i'th bit of the digital input is turned on, the resistance should be R_i, which means that if bits *b* are turned on, then resistance will be:

$$R_{total} = \left(\sum_i \frac{1}{b_i R_i} \right)^{-1}$$

It was decided to implement a 6 bit converter that should give a readout between app. 0 and 1023 if it was connected to an RCX input that is interpreted as a light sensor, which is the response from an input port configured to drive an active sensor. When an input on the RCX is interpreted as a light sensor it is possible to read the 10 bit raw value from the sensor hardware. The 6-bit digital converter is shown in figure 3.

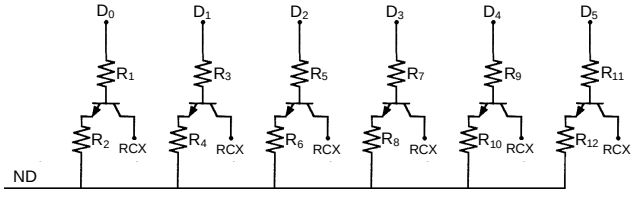

Figure 3. The 6-bit digital converter. ©LEGO Lab, 1999.

The poor range and resolution of the Lego light sensors made it necessary to construct a new sensor type that made it possible to determine the direction to some source in the environment. When working with voice recognition it was logical to make the robot determine the direction to the source of the sound.
The first set-up used two threshold microphones to detect the direction to a sound in front of the microphones. The left/right detection of this system based on a LEGO construction described in [2] was extended with an oscillator and a 4 bit binary counter to determine the delay between the triggering of the two microphones. Figure 4 is a diagram of the circuit. The readout of

bits 0 to 2 contains the delay between the microphones, bit 3 is set if the left microphone was triggered before the right, so only the three lowest bits in the counter are used. The distance between the microphones determines the longest period that can pass between the two microphones triggering, namely if the sound arrives from the extreme left or the extreme right. The counter is 4 bit, but only the three lowest are used, meaning that if the oscillator has a period of 80 μs, the longest period in the set-up shown in Figure is 7*80μs = 0.56ms, which gives a ideal distance between the microphones of 19 cm. to exploit all of the 3 bits.

It is of course important that the ears can be used in different set-ups, with varying distance between the microphones. In Figure 4 the oscillator can be adjusted from 40 μs to 100 μs, which means that the microphones must be placed at a distance from 9.5 cm to 24 cm.

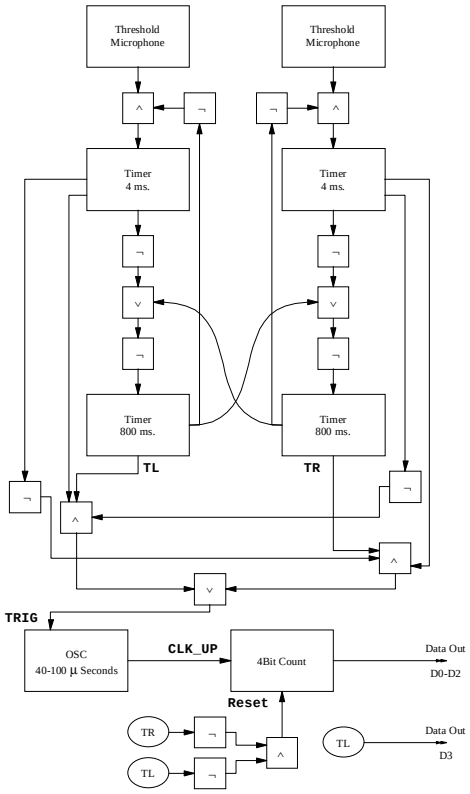

Figure 4. Determine the direction to a sound source in front of the microphones. ©LEGO Lab, 1999.

Further, we constructed bend sensors and digital compass sensors for the LEGO Mindstorms RCX, as shown in figure 5 and 6. The ears described above makes it possible for the robot to be attentive to sounds in the environment, however the communication between human and robot will be limited if the only thing that can be interpreted by the robot is the direction to the source of the sound. Making the robot able to understand a few words increases the possibilities of interaction immensely; it will then be possible to issue verbal commands to the robot, which then will be able to react upon them.

The LEGO robot has a very low bandwidth on the input ports, they are only sampled every 3 ms, which makes it impossible to sample and analyse the sound directly. A hardware solution with a microphone and a chip with speech recognition capabilities can easily be interfaced with the RCX to overcome these limitations. The speech recognition used in this project is the HM2007 based circuit from Images Co. However, because this system (as is the state of the arts in speech recognition) is not very noise tolerant, we cannot place it on the robot, but have to speak very close to a microphone. Therefore, we chose to run the speech recognition via a host computer.

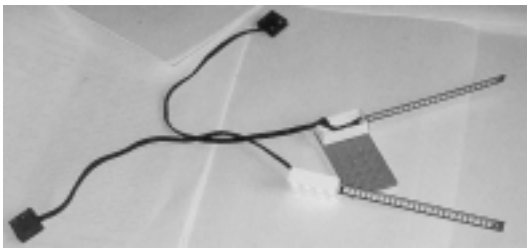

Figure 5. Two bend sensors mounted in a LEGO brick. ©LEGO Lab, 1999.

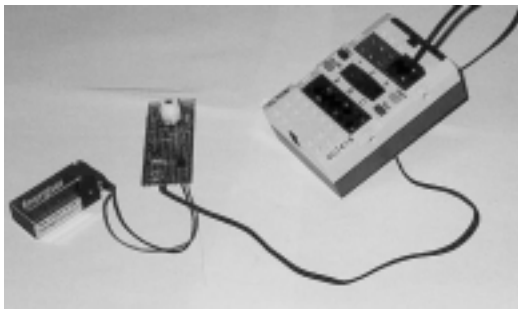

Figure 6. Digital compass sensor and battery. The digital compass is attached to a LEGO Mindstorms RCX input channel. ©LEGO Lab, 1999.

3.4 The Behaviour Engine

The behaviour engine is a framework for designing and implementing a system giving our robot the following properties:

- Action (behaviour) selection based on internal states and external input
- Emergent behaviours based on a set of behaviour primitives
- Learning by experience
- A high degree of self-sustainability
- Adaptive in a dynamic environment
- Easily scalable

Our behaviour engine is illustrated on figure 7. An underlying idea of the construction of the behaviour engine it is that is to be built from (partly) independent modules. Each module should be constructed in such a way that it can be thoroughly tested prior to the insertion into the system.

The Reinforcement Generator

The reinforcement generator is responsible for generating a reinforcement signal using the current and past sensor values as input. The older the input values are the less weight they carry. The reinforcement generator is constructed like a simple perceptron using a pseudo tapped delay line of sensor values as input.

An example of the use of training is the case of a pet robot that has been taught that the tone of word *shame* is positive, i.e. the reinforcement generator produces a positive reinforcement signal if this word is spoken (given as input to the reinforcement gene-

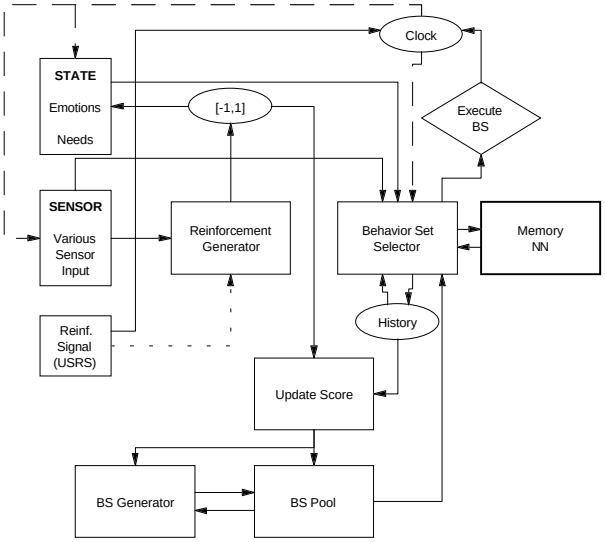

Figure 7. Behaviour engine with memory neural network. ©LEGO Lab, 1999.

rator). By punishing the pet robot every time this association is made it is possible to change the positive association of the tone of the word *shame* (which is obviously a misinterpretation by the pet robot) to something negative (which is more in line with the common interpretation of this word). This means it over time will be possible to change the meaning of the sensor input i.e. the robot can be brought up to behave in the way it is supposed to. The weights of the perceptron are trained with unsupervised learning.

Behaviour Set

The notion of a behaviour set is to link a set of sensor values and state values together with a score and the behaviours to be executed. Figure 8 shows the basic behaviour set being described in the following. The fields denoted by the *Si*'s are the sensor or state value fields. The *Bj* fields hold the information about which behaviours to execute if the behaviour set is selected. The number of *B* fields corresponds to the number of behaviour classes available. The reason for grouping the behaviours into classes is that orthogonality on the actuators is needed meaning that two behaviours, one in *Bp* and the other in *Bq*, cannot share the same actuators. In each of the behaviour classes there are two additional behaviours: the empty behaviour and a "wildcard" behaviour. The empty behaviour indicates that no behaviour from that particular class is to be executed and the "wildcard" behaviour indicates to continue executing the behaviour from this class selected in the prior behaviour set. Finally the *Score* field hold the current score of this behaviour set, a value used to keep track of when to remove a behaviour set if it has not been used for a while.

S0	S1	...	Si	B0	B1	...	Bj	Score

Figure 8. A basic behaviour set. *Sn* (*n* = [0,i]) denotes the sensor and state values and *Bn* (*n* = [0,j]) denotes the behaviours.

Each actuator has a series of possible behaviours it can implement independently of the other actuators, enabling the creation of emergent behaviours. It is important to notice that the emergent behaviours are dependent on the associations made between sensor values and the user supplied reinforcement signals in the reinforcement generator. The robot pet we are working on has the following groups of actuators that can work independently from one another:

- Wheel control
- Tail control
- Head control
- Voice control

The way the behaviour set is designed leaves the implementation details of each of the behaviours open. This mean that one behaviour may be implemented by a neural network while another may be implemented by a few lines of deterministic code.
Each of these behaviours can be developed and tested independently of the system. A neural network implementing an exploration behaviour can be evolved and tested in the simulator, put on a specially designed neural network chip or simulated in software where it can be tested once again in real life and finally integrated into the behaviour system.

Behaviour Set Selector

A system is needed for selecting the appropriate behaviour set. This system is implemented in the behaviour set selector module. This module ranks the behaviour sets available on the basis of the current sensor and state values of the behaviour set. The final rank of a behaviour set is determined by the distance between the current sensor and state values and the values stored in the behaviour set. What behaviour set to select depends on the rank unless the current active behaviour set is among the elite of the highest-ranking sets. To avoid a constant shift in the active behaviour set the one currently executing is favoured i.e. it has a higher probability of being selected even if it has a lower rank than the highest-ranking behaviour set. This higher probability decreases over time to allow a more graduate shift in behaviour.
We keep track of the last few selected behaviour sets to store the information about the past sequences of behaviour sets. Later on when a few extension are added it will enable the system to learn well performing sequences of behaviour sets i.e. sequences leading up to positive reinforcement. Poorly performing sequences i.e. sequences leading up to negative reinforcement will on the contrary be punished by the system. This will result in the ability of the system to acquire various sequences of behaviours over time but still the system has the ability to introduce uncertainties making the robot look like it has a will of its own.

Update Score

Whenever a reinforcement signal is generated the behaviour set currently selected gets its score updated. The function used to update the score is a function of the reinforcement signal, the current score of the behaviour set and the age of the behaviour set. Thus if a negative reinforcement signal is received the score is decreased and conversely increased if the reinforcement signal is positive.
Since the purpose of the score is to keep track of behaviour sets currently not executing, the score of behaviour sets currently not executing is reduced by some fixed value every n'th clock tick.

Memory Neural Network

Each time the behaviour selector is called upon to select a new behaviour set, the memory neural network is consulted. It is asked to estimate the reinforcement signal it might receive as a result of executing the behaviour set. In case the neural network

implementing the memory module is a simple feed forward network, it should have as input a brief history of the past behaviour sets selected and their matching sensor and state values.

Evolution of Behaviour Sets

We look upon the set of behaviour sets from an evolutionary algorithm point of view. This means that the set of behaviour sets is seen as a population of behaviour sets on which genetic operators like mutation and crossover can be applied. By using this approach we wish to obtain a high degree of emergent and adaptive behaviour that it would be hard to program explicitly.

State Variables

The state variables are split into two groups. The first group of state variables is classified as emotions and the second group as needs. Suggestions to which emotions and needs we can add to the system is given in the following:

- Hunger (needs)
- Sleep (needs)
- Anger (emotions)
- Sadness (emotions)
- Restlessness (emotions)

At each signal received by the external clock the values of the state module are updated. The values gradually change over time to reflect the ongoing change in the metabolic level. As an example, the need for sleep varies during the day and night on a 24-hour scale. Other values in the state module are changed dependent on the reinforcement signal generated on basis of the sensor values. These changes are not time dependent but depend on the interaction with the user and the surrounding environment.

To each need and emotion is attached one or more control variables used by the behaviour engine for various purposes during e.g. the selection of behaviour sets. These control variables give the personality of the pet robot. Thus changing the value or one or more of these variables can vary the personality of the pet robot adding an extra dimension to the already existing multiplicity of the behaviour engine. Since the personality of the pet robot is comprised by a set of control variables, it is possible to make a library of personalities giving the user the opportunity to change the personality of the robot at will.

4. Experimental Results

Some of the behaviours, such as sit down and waggle tail are hard coded into the system and will not be discussed. The more complex behaviours have been evolved using a simulator, which we first evaluated to be able to cross the gap between simulation and reality on a Khepera robot task. In the following section the results of the evolutionary processes will be presented, followed by the results of those behaviour inserted in the behavioural engine. Experiments show how the robot reacts when subjected to different kinds of reinforcement, how it learns to obey verbal commands, and how it acts autonomously when internal needs such as the need for food arise. All documentation shown is shots from the simulation environment; video sequences of the behaviour of the physical robot can be found at:
http://legolab.daimi.au.dk/Video

All behaviours are evolved in the same environment, see Figure 9. The arena is 5 times 3 meters, which corresponds to the size of an office in our institute. Within the room two L-shaped obstacles have been placed. All obstacles are surrounded by white paper taped to the floor. This means that the robot can use the vertically placed light sensors to obtain information about the environment. A green area in the northwest corner of the room marks the home spot, the place where the dog-basket is placed. Using this room might make the robots converge to a behaviour that is dependent on the layout of the room. This is of course the case when the homing behaviour is evolved, but in the case of exploration and food seeking, the diversity in the objects should make the evolved control systems able to work in a wider variety of environments.

When evolving a behaviour, the result can be dependent on the initial, randomly generated, population and the random spots in the environment on which the evaluations start. Therefore, to study robustness, each behaviour was evolved ten times.

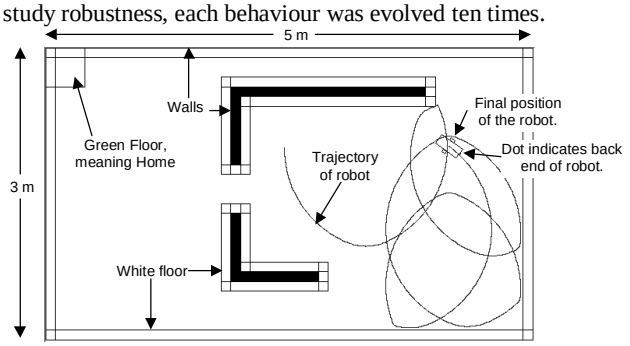

Figure 9. The environment. ©LEGO Lab, 1999.

Furthermore each experiment has been conducted twice, using two different types of neural networks, a simple feed-forward network, and a recurrent network. We have successfully evolved exploration, homing behaviour, and food-seeking behaviour, see for instance homing behaviour using a recurrent neural network and corresponding fitness curve on figure 10 and 11.

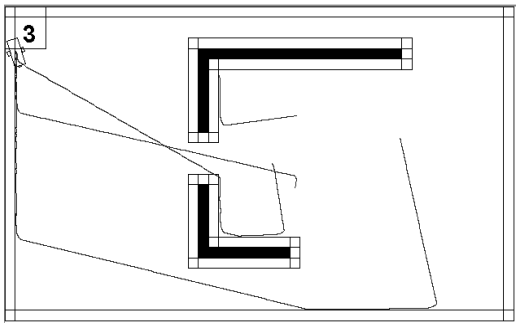

Figure 10. Homing behaviour using a recurrent network. The number 3 indicates that the pet robot reaches the “basket” three times. ©LEGO Lab, 1999.

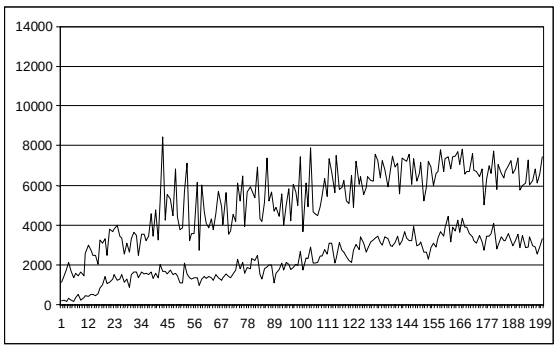

Figure 11. The fitness graph (best and average fitness) leading to the individual of Figure 10. ©LEGO Lab, 1999.

The behaviour engine was used in training an individual, to learn the commands for 3 different behaviours, and later to use those newly learned behaviour in the environment. The behaviours themselves were trained using the genetic system and the simulator. The evolved neural networks from the training sessions were coded into the execute-behaviour module. In addition to those, it got a sitting behaviour, that is hand coded, and 2 undefined behaviours, allowing the pet robot (dog) to mind it's own business. At first, the dog has no knowledge of any associations between the commands, and the behaviours, and some teaching is therefore required. This is done by punishing the dog, when it is doing something wrong, and hence one tries to allow the robot to make associations between commands and behaviours.

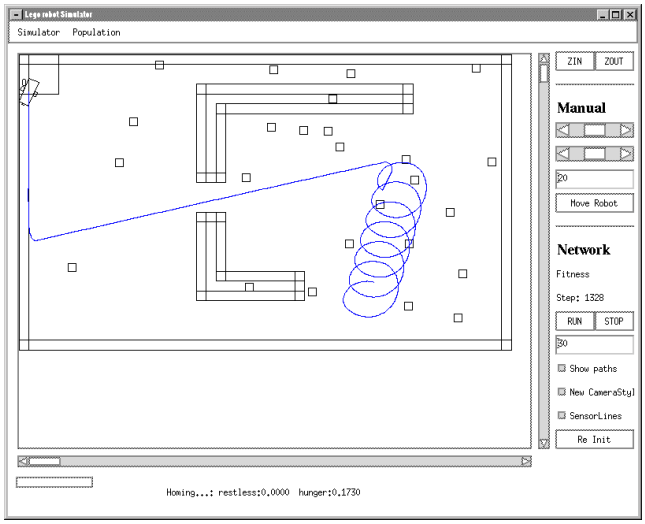

Figure 12. The robot dog gets the command *Explore* and explores the environment with a circling behaviour. When the command *Go Home* is said, the robot dog heads for its home. ©LEGO Lab, 1999.

Figure 12 shows an example of the robot that has been trained to some degree moving around in its environment. First it is given an *Explore* command, and it starts to explore the environment by circling behaviour. Then it is given the command *Go Home*, and it starts heading for the north west corner of the room where its basket is located. As can be observed at the bottom of the figure, the restlessness is at a minimum, the homing behaviour is also considered to be exciting and thereby reducing the restlessness (although not as quick as the exploration behaviour, which is considered to be more exciting). The hunger is increasing, but is still not near a critical level for the dog. Unfortunately, space limitations do not allow us to document all the experiments and replications in a more detailed manner. This will be done elsewhere.

5. Conclusions

We have described the *robot=human view on robotics*, and shown briefly how it influences the robot applications by putting emphasis on a human-like interaction with the surrounding environment (e.g. by expressing sympathy and empathy in an emotional robot). The emphasis on human-like interaction also directs the pet robot project to consider adaptiveness as the major feature to develop.

The pet robot project shows, as one of the first examples, that it is possible to scale up evolutionary robotics to more complex behaviours than were traditionally shown. This is achieved by using the divide-and-conquer principle of behaviour-based systems, instead of the traditional monolithic approach in evolutionary robotics.

Acknowledgement

The human-robot view is inspired by discussions with Christian Majgaard, LEGO. Part of this study was supported by LEGO, though the views expressed in this paper might not necessarily express the viewpoints of the LEGO Group. The study was facilitated through the help and interaction with Lars Nielsen, Michael Thomsen (now Interactive Institute), Bent Sørensen, and Michael Andersen from the LEGO Group. The facial expression studies were inspired by discussion with Orazio Miglino, II University of Naples, and Jakob Fredslund from the LEGO Lab made earlier prototypes of facial expressions with LEGO. In general, we are thankful for the interaction with all the students and researchers at the LEGO Lab.

References

[1] C. Davidson, "*Agents from Albia*", New scientist, 09 May 1998, New Scientist, RBI Limited 1998.

[2] D. W. Hogg, F. Martin, M. Resnick. "*Braitenberg Creatures*" E&L Memo No 13, MIT Media Lab. Cambridge, MA, 1991.

[3] W.-P. Lee, J. Hallam, and H. H. Lund. "*Learning Complex Robot Behaviours by Evolutionary Approach*", in Proceedings of 6th European Workshop on Learning Robots, 1997. In A. Birk and J. Demiris (eds.) Learning Robots. pp. 155-172. Lecture Notes in Artificial Intelligence 1545, Springer-Verlag, Heidelberg, 1999.

[4] Henrik Hautop Lund, John Hallam, and Wei-Po Lee. "*Evolving Robot Morphology*". Invited Paper. In Proceedings of IEEE 4th International Conference on Evolutionary Computation. IEEE Press, NJ, 1997.

[5] T. Shibata and K. Tanie, "*Creation of Subjective Value through Physical Interaction between Human and Machine*", Proceedings of the Fourth International Symposium on Artificial Life and Robotics (AROB 4th'99), pp. 20-23, ISAROB, Oita, 1999.

[6] Sony Press release, June 10, 1998. http://www.world.sony.com/CorporateInfo/News-E/199806/98-052/index.html