ELSEVIER

Applied Soft Computing 1 (2001) 3–20

Applied Soft Computing

www.elsevier.com/locate/asoc

Invited paper

Adaptive robotics in entertainment

Henrik Hautop Lund*

Maersk Mc-Kinney Moller Institute for Production Technology, University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark

Accepted 26 March 2001

Abstract

Techniques developed in the fields of evolutionary computation, adaptive systems, agents, and artificial neural networks can be used in entertainment robotics in order to provide easy access to the robot technology. We have developed a number of user-guided approaches based on the techniques from these research fields. These techniques include user-guided behaviour-based systems, user-guided evolutionary robotics, user-guided co-evolutionary robotics, and morphological development. All these techniques are applied to allow children to develop their own robot behaviours in a very easy and fast manner. Here, I show examples with development of Khepera robots and LEGO MINDSTORMS robots, including the World Cup'98 stadium, the Co-evolutionary Robot Soccer Show, the Toybot Soccer Player, the LEGO Interactive Football, and RoboCup Junior Rescue. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Robotics; Entertainment; Edutainment; Behaviour-based robotics; Evolutionary robotics; LEGO; Co-evolution; User-guided adaptive behaviour; Constructionism

1. Introduction

For the last 15 years, there has been a strong development of adaptive robotics, starting from the introduction of behaviour-based robotics by Brooks [3] and the more popular description of such systems by Braitenberg [2] (see also [4] for a review of the use of behaviour-based systems for robot control, and [1] for a thorough text book). There has been numerous examples of how to use artificial neural networks as robot controllers [8,19], evolutionary computation for development of the robot controllers [5,6,21], and different kinds of behaviour-based systems for control [1,3].

Most of these systems are supposed to adapt to the environment, and often one can consider an adaptation to the ecological niche (see [24]). The advances in making adaptation have been considerable. However, most often the robotic systems and the adaptation of these are tested only on simple problems such as obstacle avoidance, homing, line following, etc. which is in line with Braitenberg's [2] suggestions, but not complex enough to attract the attention over longer periods of time in an entertainment application or to fulfil the goals of most industrial applications.

A second problem appears in the autonomous systems approach. Often, the goal is to achieve fully autonomous robots, both in the development and the behaviour. This is highly desirable from a theoretical point of view and in some fully autonomous system applications, but sometimes, in other applications,

* Tel.: +45-6550-3574;
fax: +45-6550-7697; URL: http://www.mip.sdu.dk/~hhl.
E-mail address: hhl@mip.sdu.dk (H.H. Lund).

it may turn out to be less desirable. For instance, in entertainment that involves construction, the user would like to be able to direct the development of the system, and in production systems, the worker in a production hall should be able to re-configure the robot for flexible production.

We have addressed these two issues regarding adaptive robotic systems by developing methods for achieving complex robot behaviours and for interaction with the autonomous system. In the following, this development is described through example implementations of entertainment robots and user-guided approaches. Essentially, all examples implement intelligent agents in entertainment and edutainment.

2. Intelligent agents in entertainment

An intelligent agent can be defined as a system with "intelligence" that performs actions or interactions. Previously, such agents have been used in the entertainment industry mainly in computer games for achieving life-like behaviours [27,28]. For instance, in a computer game an agent may be used as a character with whom the user is supposed to interact. The agent may learn about the reactions of the user in order to increase its skills and in this way better challenge the user (as opposed to games where the agent is static and not adapting through the user's interactions).

If we define the user as an agent, we can say that the traditional use of intelligent agents has been a set-up involving two intelligent agents, namely the user and the computer game agent. I envision the modern use of intelligent agents to broaden this use to physical interactions with the artificial agent and the interplay between more intelligent agents [9]. In order to achieve physical interactions, it is necessary to move the intelligent agent from the pure, virtual reality in the computer to the physical reality. It is possible to move the actions and interactions between a user and a virtual, intelligent agent to a physical interaction through haptic interfaces. Another possibility is to make the complete transfer to the physical reality by constructing robots for this purpose. Examples of such robots are the SONY AIBO, LEGO MINDSTORMS robots, I-Cybie, etc. These physical robots facilitate interaction between two intelligent agents (user and robot) in the physical world. Further, it is possible to make interactions between three intelligent agents in the form of user, robot, and computer game. LEGO CyberMaster is an example that facilitates such a set-up, where the physical robot is controlled from a computer game, and the child can interact with the physical robot in the physical world. Finally, we can envision the interaction between four or more intelligent agent by making an intelligent room, so that we have the following intelligent agents to interact: user, robot, computer agent, and intelligent room (Table 1).

There are significant differences in between the different entertainment robots mentioned above. In some cases, the robots are fully autonomous both in development and behaviour (e.g. Furby) and so give *no* possibility for development by the user, in some cases there are *limited* possibilities for development by the user (e.g. I-Cybie and AIBO), and in other cases there are *extensive* possibilities for development by the user (e.g. LEGO MINDSTORMS and FischerTechnic robot). Here, I will concentrate on the latter kind of entertainment robots, since I view these systems to best facilitate an educational approach in applications for children.

Table 1
Classification of interaction between intelligent agents[a]

Number of intelligent agents	Intelligent agent(s)	Classification
1	Child	Traditional
1	Animation	Traditional
2	User, computer game	Traditional
2	User, robot	New
3	User, robot, computer game	New
4	User, robot, computer game, intelligent room	New

[a] It is possible to imagine a number of intelligent agents interacting with each other. Note that the human user is defined as an intelligent agent in this classification.

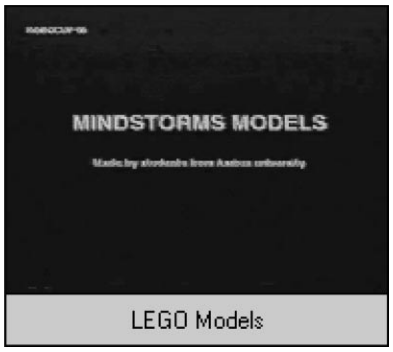


Fig. 1. LEGO MINDSTORMS models including the Pacman game.

In order to investigate the possibilities of transferring the intelligent agent from the traditional use in a computer game to the physical reality, we made a number of initial experiments with LEGO MINDSTORMS robots. They were displayed and used by children during RoboCup'98 in Paris, and can be viewed in Fig. 1. Especially the well-known computer game from the early 1980s called Pacman was developed as a robot game. Pacman is the game where a yellow cheese is controlled to move around in a maze while being chased by a number of ghosts. In the robot game, there would be two red ghost robots and a yellow cheese robot moving around in a maze made by attaching black adhesive tape on a white floor (a white plastic table cloth). All three robots were programmed using the behaviour-based approach. In the behaviour-based approach, the robot programmer first designs a low level of competence, implements and debugs this level. When this level is fully functional, the programmer can start adding new levels of competence one on top of each other. In the case of the Pacman game, the robots had to be programmed to have the following competencies: avoid lines, avoid when colliding, move forward, and turn in junctions. These behaviours would allow the robots to move around in the maze. By adding an extra layer, one could design a goal-directed behaviour. In this way, it was possible to design a fully autonomous display of three robots moving around in the maze. However, a fully autonomous Pacman game would be no fun for the children. So an extra layer of competence was introduced in the yellow cheese robot, namely the goal-directed behaviour through interaction by the child user. A joystick was made out of LEGO MINDSTORMS, and the commands received by the yellow cheese robot through infra-red communication from the joystick would enter as high-level commands in the behaviour-based system. In this way, the user would be able to direct the behaviour of the (no longer fully autonomous) robot, and try to direct it to the centre of the maze (the goal position) while avoiding being hunted down by the two red ghost robots, i.e. the two fully autonomous robots. Apart from fulfilling the goal of providing a fun and interesting physical robot game, the approach showed one of the advantages of the behaviour-based approach, namely the division into behaviours and the construction of layer after layer. It was comparably simple to add new layers of competencies in order to provide the necessary behaviours and to allow the interaction between the two intelligent agents: the child and the robot.

The success of the Pacman game prompted us to use similar approaches in other applications such as the fashion show (see Fig. 2), the artist robots (see Fig. 3), and the musician robots (see Fig. 4). The fashion robots make a catwalk and numerous robot models show different designed dresses. The artist robot

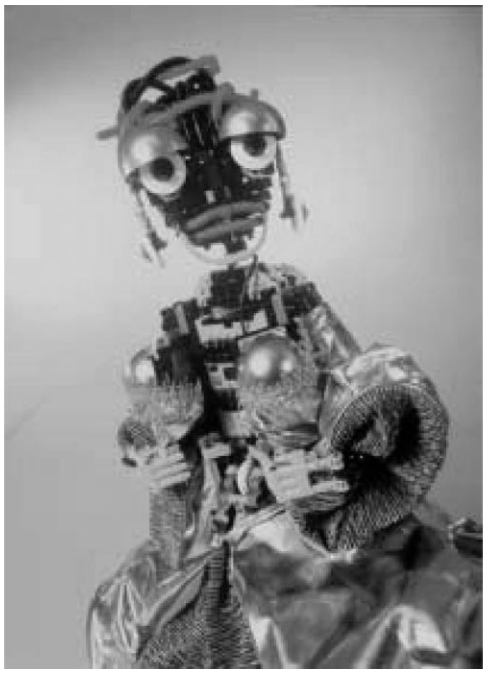


Fig. 2. A fashion robot for catwalk.

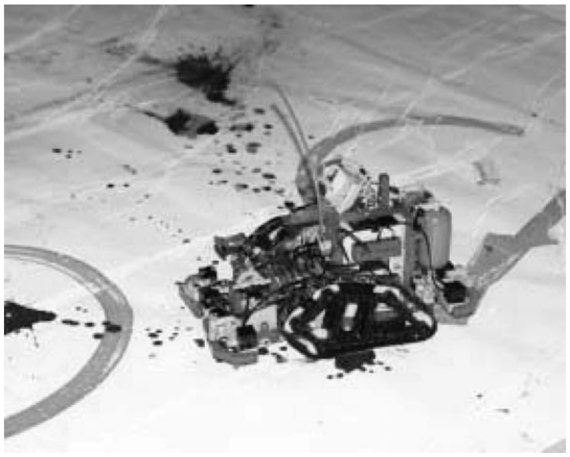


Fig. 3. An artist robot.

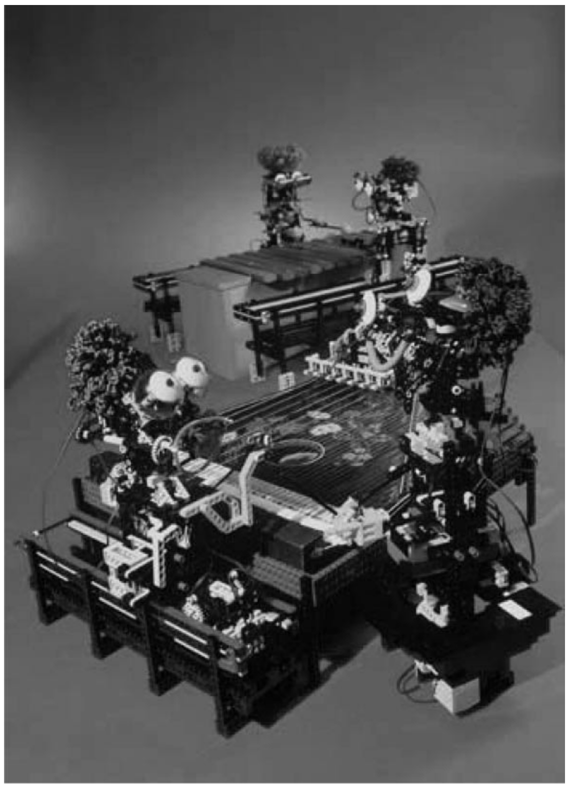


Fig. 4. The robot orchestra with musicians and conductor.

set-up is based on four autonomous robots painting on a canvas on the floor. They can either paint with a paintbrush, a pen, spray paint, or splash paint. The four robots are programmed with a similar behaviour-based approach as in the Pacman game, so that they all have different levels of competencies. A human artist is provided with the joystick and can select which of the four robots to interact with at a given time — the three other robot artists will be moving around fully autonomous. In this way, the human artist can interact with the autonomous multi-robot system, and direct the behaviours so as to influence the outcome: a human-robot designed work of art.

The orchestra composed of musician robots is another example of a multi-robot system where an intelligent agent tries to direct the behaviour of the other agents. Here, the single robot plays an instrument such as drums, xylophone, and string instrument. The robot orchestra played the Johann Strauss "An der schönen blauen Donau". We can describe a real orchestra as being nice if it is composed of good individual musicians, but it is also crucial that these musicians can co-ordinate their performance. The same is true for a robot orchestra. Since the robots are autonomous agents, they all play with their own timing, and so there is the risk that they may get out of beat. As known from real orchestras, a conductor can ensure that the individual intelligent agents (musicians) keep the beat by providing an external signal for the beat. Hence, a robot conductor was constructed, and this robot conductor provides the beat through infra-red communication, so that the individual autonomous robot musicians will adjust to the beat when they receive it. However, infra-red communication is often unreliable, so it is important that the robots are autonomous, so that they will perform even when the signal is not received. All that happens when missing a signal is that the robot will adjust to the beat only slightly later (e.g. 1 s later) when it receives the next signal from the conductor. This is one of the many advantages of behaviour-based systems: behaviours run in parallel and low level behaviours still work when higher level behaviours do not.

3. Soccer World Cup'98 demonstration

Another use of behaviour-based robotics in entertainment is for robot soccer. Before the LEGO

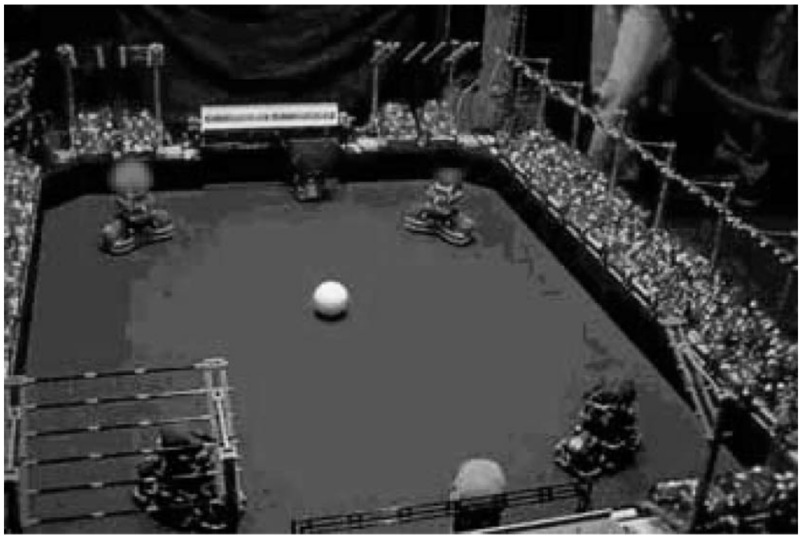


Fig. 5. The LEGO robot soccer set-up. There is one goalkeeper and two field players on each team (one red and one blue team). The stadium has light towers, scanning cameras that project images to large monitors, scoreboard, rolling commercials, and almost 1500 small LEGO spectators that make the "Mexican wave". Copyright H.H. Lund, 1998.

MINDSTORMS Robotic Invention System was to be released on market, we wanted to make a large-scale test of the robot kit. We selected to make a LEGO MINDSTORMS robot soccer game with a distributed behaviour-based system to be demonstrated in Paris during the soccer World Cup'98 (France). Robot soccer has been defined as a new landmark project for artificial intelligence, and its characteristics fitted our purpose. In contrast to previous artificial intelligence challenges such as computer chess, robot soccer is a dynamic and physical game, where real time control is essential. Further, where a game like chess might allow extensive use of symbolic representation, robot control puts emphasis on embodiment and many aspects of this prohibits the use of symbolic representation. In general, participating in robot soccer is believed to provide both students and researchers with knowledge about the importance of embodiment and the problems that ungrounded abstractions might lead to [14] (Fig. 5).

However, we also found a number of problems that had to be solved before robot soccer could be made appealing for a public audience. Robot soccer is a very young research field, so the performance of the robot soccer players might not look especially impressive from a public audience's point of view. Even though, we expected a huge public interest in our LEGO MINDSTORMS robot soccer demonstration (indeed, our LEGO MINDSTORMS robot soccer demonstration was broadcasted to an estimated 200–250 million television viewers world-wide), so it was important to alleviate this problem. The robot game had to be put into the right context. From an aesthetic point of view, other robot soccer players might be looked at as essentially cubic, metallic devices that move around in a pen and push a ball — it might not appear to be much like soccer to the public audience if the audience is not told in advance to look at this as soccer. Therefore, in our robot soccer game, we put much more emphasis on making a context that immediately would allow the public audience to recognise the game to be a soccer game.

This was done by making a whole stadium (which we named Stade de Victor LEGO) out of LEGO with light towers, rolling commercials, and almost 1500 LEGO spectators who made the "wave", by providing sounds related to the game (tackling, kicking, spectator noise, etc.), and by giving the robot soccer players a face (for further details, see [11,15]). Indeed, in the developing phase, we had a graphical designer to make huge colour drawings of possible scenarios, we had a technical designer to make appealing facial

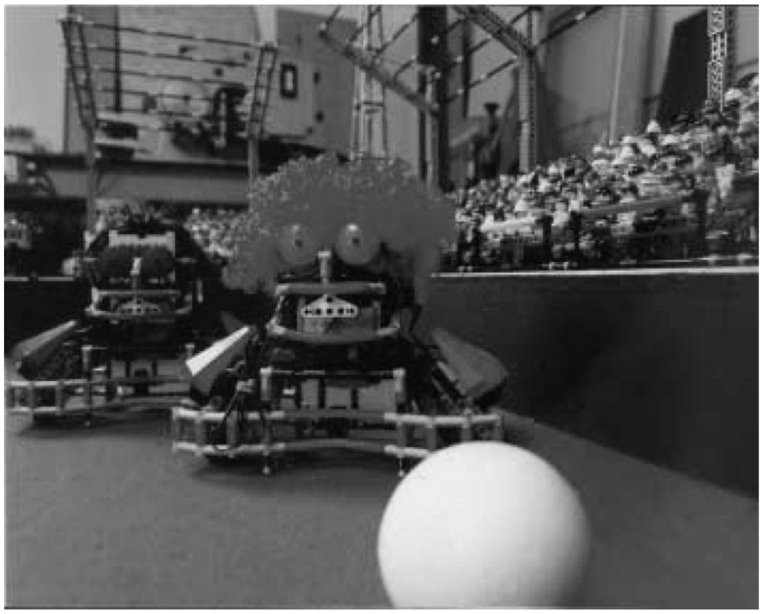


Fig. 6. A LEGO robot soccer player. Copyright H.H. Lund, 1998.

expression of the robots, and we made different scripts for games (how to enter the field, how to sing the national anthems, how to get into the kick off positions, what play strategies to use, etc.).

In the design of LEGO MINDSTORMS robot soccer players, we used three rotation sensors: two sensors to give information about the movement of the wheels, and one sensor to give information about the movement of the mouth (see Fig. 6). Two independent motors were used to drive the two wheels, and one motor was used to allow the mouth to open and close (to make the robot "shout" or "sing"). When opening the mouth, a small bar on the back of the robot would move down on the ground, which made the robot tip slightly forward. This was made in order to make a shout look even more lively. However, the functionality turned out to provide a nice, unintentional use. The border of the field had a chamfered edge to allow the ball not getting stuck up against the wall. However, because of the slight slope, the robot soccer players could get stuck at the edge, since the distance between the wheels touch point on the ground and the base of the robot was quite small. The problem was solved by allowing the robot soccer player to scream after being stuck for a short time. Apart from opening the mouth, this would make the bar on the back go down on the ground and tip the robot, so that it would actually get free.

In order to detect the position of the individual players and the ball, we used an overhead camera and processed the input in a hardware vision system connected to a host computer. The LEGO MINDSTORMS robot soccer players were programmed using the MINDSTORMS OCX together with Microsoft Visual Basic. The control system was a distributed system between the robots and the host computer. The host computer ran a program that collected co-ordinates from the hardware vision system, processed this information and sent angles to turn out via the infra-red transmitters. The robots had a control system that allowed them to collect the appropriate data, and react accordingly. We implemented a kind of behaviour-based control where some competencies are placed on the robot, while other higher levels of competence are placed on the host computer.

It is essential that low level primitives, such as the control of motors and sensors are placed directly on the robot. In order to ensure precise control, the input channels should be used to obtain feedback as often as possible. Control from a host computer of low level primitives such as turning a specific angle is very risky because infra-red communication might fail and hence feedback from sensors is lost until communication is re-established. Imagine that we allow one of the motors to turn until the angle sensor exceeds a specific threshold for turning the angle that we want the robot to turn. When having this control on the robot, we will be able to read the angle sensor at very short intervals, and hence stop the turning immediately after the threshold is reached. On the other hand, if we placed this low level control on the host computer, then we would have to send the reading of the angle sensor from the robot to the computer for each reading. The time for doing this depends on the transmission time for a message, but sometimes it might take even longer because we have to retransmit when the package is not received. Afterwards, the command to stop the motor when a value larger than the threshold was received has to be sent back to the RCX. Because of the long time delay, the robot will no longer turn the desired angle (or very close to the desired angle), but will turn a larger angle (and it will be larger the worse the communication is). In total, the time between input is received on the RCX until the motor command is activated will be much longer than when control is directly on the robot itself. Therefore, the control of low level primitives should take place on the robot itself in a distributed behaviour-based system.

Essentially, we are defining the time step (or sampling time interval) in the control system (note that even though we might talk of a reactive behavioural module, there will still be a time step defined by the sampling rate). Some competencies will need a very short time step in order to make appropriate control, while other competencies can do with a longer time step. Basically, we find that the necessary time step decreases in length as we go from higher levels of competence to lower levels of competence. Based on an analysis of the necessary time step together with an analysis of communication time between the different distributed units in the system, we can identify the distribution of competencies in the different computational units (in this case, in the host computer and the robots).

The analysis showed that the competencies for moving forward and turning specific angles should be implemented directly on the robot. Other higher level competencies were implemented on the host computer. These higher level competencies include an action decision mechanism that selects what angle to send to the robot, and, on top of this competence, a planner that decides where to have the robot to move in the field. The total of five (another classification might result in only four competencies, since turn and angle selection might be classified as just one competence, but since they are placed in each their component of the distributed system, we classify them as two competencies) competencies in the distributed behaviour-based system is shown in Table 2.

Table 2
The distributed behaviour-based control system of the LEGO robot soccer players

Behaviour	Level	Computational unit
Position planner	High level	Host computer
Angle selection	High level	Host computer
Turn (and scream)	Low level	Robot
Forward	Low level	Robot
Stop and wait	Low level	Robot

The simplest competence in the distributed behaviour-based control system is simply stop and wait, which is implemented in the robot. This control will simply make the robot stand still. One level up, a move forward command can overwrite (or subsume) the ‘stop and wait’ for a specific amount of time. The time of this subsumption was found with empirical tests. It allows the robot soccer player to move forward approximately 15 cm. With these two basic layers of competencies, the robot will move forward, stop and wait, and then again move forward, and so on. The third layer, which is also implemented in the robot, is the turn competence. It reads a robot specific (it has its own ID) register and turns the amount of degrees that is read in the register, if the value in the register has changed. Afterwards, it resets the value in the register to zero. A specific value in the register will trigger the scream behaviour (open and close mouth before turning). Hence, this third layer subsumes the two lower levels for the time that it takes the robot to turn, i.e. if a non-zero value is read in the register, the robot will turn, and when the turning movement has finished, the forward movement might be issued again, so in a total the robot will turn and then move forward. As mentioned above, it is essential that these low level competencies of the distributed behaviour-based system are implemented directly on the robot. (Further, in

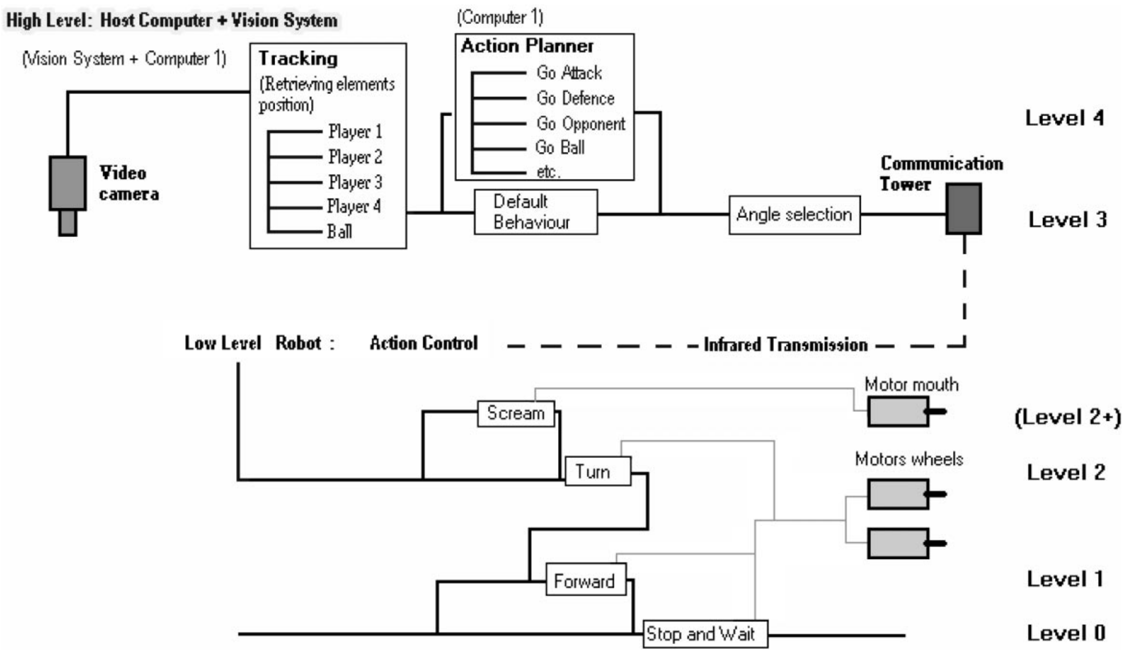


Fig. 7. Schematic view of the distributed behaviour-based control system of the LEGO MINDSTORMS robot soccer. The rectangles represent behavioural modules. A module on a higher level can subsume a module on a lower level. From below, each level is functional without the upper levels. Hence, the robot is functional with only level 0, with only level 0 + level 1, with only level 0 + level 1 + level 2, etc. The grey lines represent control commands to the motors. Control commands are sent by the behavioural modules, and can be overwritten by commands from higher levels (e.g. a motor command from the 'forward' module can overwrite the motor command from the 'stop and wait' module). A black line arriving from an upper level to a horizontal line might overwrite the information that is being sent on the horizontal line. For example, the implementation of level 4 ('action planner') will give an output that will overwrite the output from the 'default behaviour' module in level 3, and therefore the robot will take the action decided in the 'action planner' rather than the 'default behaviour'. Copyright H.H. Lund and L. Pagliarini, 1999.

order for the robot to turn the exact degree found in its register, extensive empirical measurements were used to estimate the relationship between reading of angle sensor and degree of turn. This information was then used in robot-individual look-up tables, very similar to the approach for estimating motor characteristics suggested by Miglino et al. [18].)

The higher level competencies of the distributed behaviour-based system were implemented on the host computer (see Fig. 7). First there is an angle selection competence that calculates the angle between two vectors and communicate it via the infrared communication towers. The two vectors represent the robot's previous movement, and the line between the robot's position and the desired position (where the robot should move to). We implemented a default position that each robot soccer player had to go to. As default, the blue players had to go to each their side of their own team's defence zone (they were supposed to be wing-backs), while the red players, as default, were supposed to go to the ball (they were more aggressive like forwards).

The highest level of competence in the distributed behaviour-based system is a position planner that decides the strategy of the single player, i.e. where the robots should move. This level subsumes the part of the angle selection competence that makes the single robot move with the default behaviour. In the LEGO MINDSTORMS robot soccer demonstration that we showed in Paris, this competence provided the robot soccer players with fairly simple strategies. It allowed the red players to play a central, attack friendly game, and the blue players to play a more defence-like game. Essentially, the red player closest to the ball would move towards the ball and the other red player would place itself approximately 50 cm behind the ball (and the other red player). If a red player was in very close proximity to the ball (e.g. touching the ball or just behind it), it would move towards the opponent goal, and by this trying to score a goal. The blue players would move around in each their side of their defence zone, but if the ball was in their own side of the field (in front of themselves), they would move forward to kick it forward towards the opponent goal. In this sense, the red players played an attacking, forward game, while the blue players played a more defensive, wing-back game. The game the two teams played, as far as we could observe it, was quite balanced and therefore each competition had a fairly unpredictable result.

In total, the behaviour-based control system gave a robot soccer play that allowed the robots to play a good robot soccer game, where goals were scored in most periods. In the demonstration tournament, we played games with five periods of up to 2 min (or until a goal was scored), and on an average three to four goals were scored in each match. This is higher average of goals/period than in most other robot soccer games, and it was essential for us to achieve this good performance in order to provide a spectacular game for the public audience (Figs. 8–10).

Certainly, the strategy of the game can be improved. For the demonstration tournament, our main objective was to make a lively game and ensure a convincing performance of the robots. We used most of the development time on designing the distributed behaviour-based approach, on ensuring fast and reliable vision tracking and communication. We did not use much time on implementing different strategies for the single robot soccer players. However, this is where the distributed behaviour-based system is at its strongest. The implementation of new strategies can be done in very short time indeed, since all that has to be changed is the highest competence level, namely the action planner. Since this is a simple subsumption system, everything from the lower levels can be re-used (and most of it should be re-used). A few lines of code (in the order of 5–10 lines of code) will change the behavioural strategy of a robot (with the same hardware structure) totally, e.g. from having an attacking behaviour to having a midfield behaviour.

Further, we can imagine an interesting higher layer of competence being implemented in a simple way on top of the other layers. This layer would be a team play competence, which makes the players move around and pass the ball in accordance of the position of the other players (team players as well as

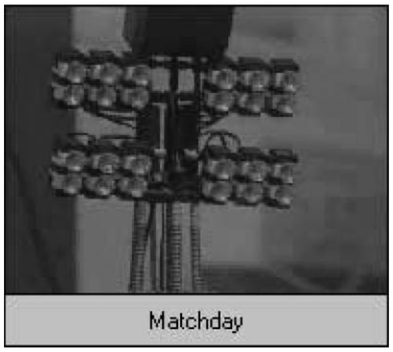


Fig. 9. Another figure of the robot soccer game.

Fig. 8. The LEGO robot soccer World Cup’98 game.

Fig. 10. The 1500 LEGO spectators make the ‘Mexican wave’.

opponents). This layer could be implemented in the host computer on top of the action planner. In a sense, the new layer would act as a coach who directs all the independent players in a team to have a co-ordinated game of soccer.

However, simple that we might find it to add higher levels of competence to the distributed behaviour-based system, it must be noted that we inherit the problem of "level design" from the subsumption architecture. The design of higher levels of competence is critically dependent on the design and implementation of lower levels of competence. We have not solved this problem by going to a distributed system. I believe that this problem is one of the reasons why Brooks [4] finds that only few experiments with behaviour-based robotic systems have achieved high complexity or high level of performance. On the other hand, with the distributed behaviour-based system, we were able to make a very well performing and robust behaviour-based system with many levels of competence. In the few cases of success, it seems valid to ask whether the success of behaviour-based systems is due to the design paradigm or due to competent engineers. Regardless of the advances that we have made in this example, in my point of view, this still remains an open question.

4. Interactive LEGO football for RoboCup Junior

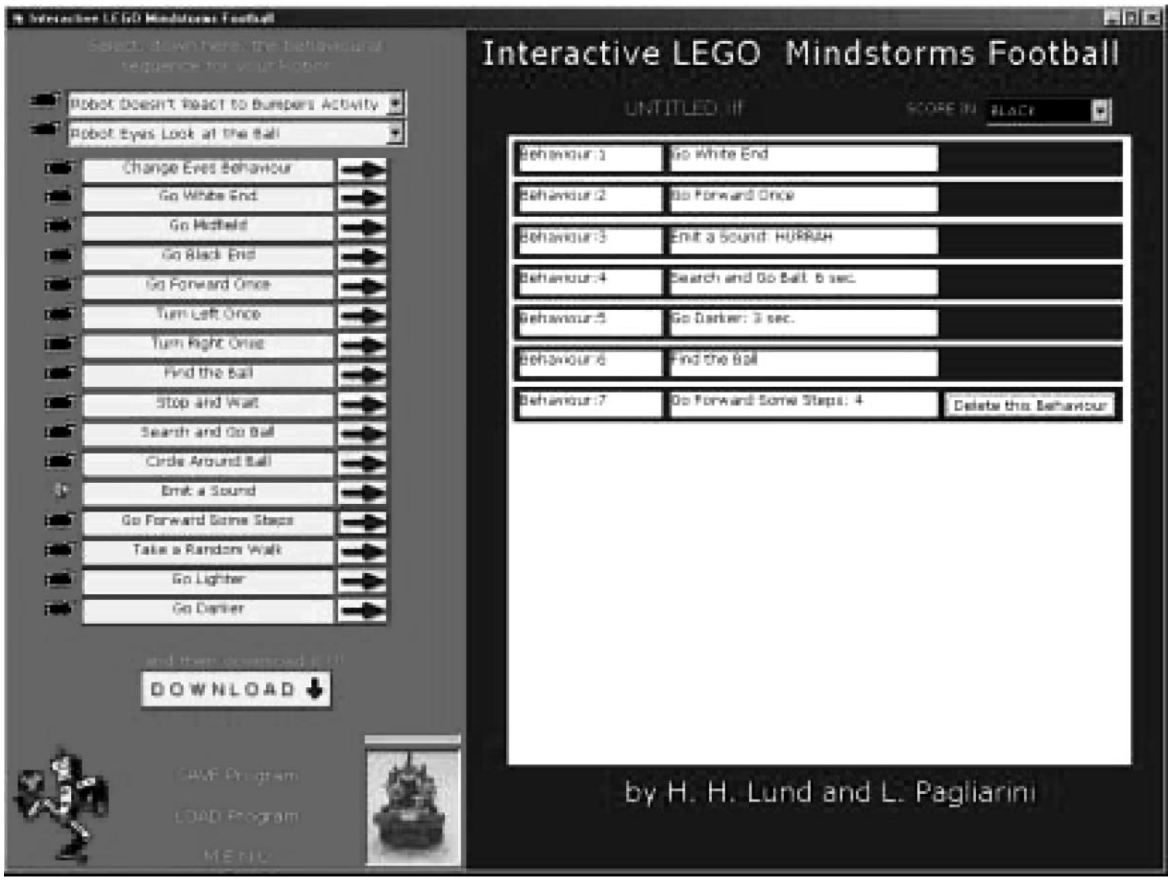
Though the robot soccer system was a nice demonstration, there was the drawback that it did not allow the interaction by the users. This is a major drawback when we are concerned with children learning by getting hands-on experience. Indeed, there may be a conflict between much modern research on developing autonomous systems, and the educational research putting emphasis on interaction, e.g. in guided constructionism [10]. Also, classical constructionism with its roots in the work by J. Piaget suggests that the best way to learn about an artefact is to actually build the artefact. Another issue to be solved was that the set-up with an overhead camera was complicated, and so is not feasible for schoolteachers to set up.

Therefore, for the RoboCup Junior during RoboCup'99 in Stockholm, I developed another well functioning LEGO MINDSTORMS robot soccer game together with L. Pagliarini. The use of overhead camera was avoided by making a special transparent ball with infra-red transmitters of approximately the same wavelength as the detectors in the LEGO MINDSTORMS light sensors. This facilitated the perception of the ball, while the recognition of position in the field was facilitated by colour (grey) codes on the floor.

The aim of our RoboCup Junior game was to allow children to get hands-on experience with robotics, and for this purpose we set up a LEGO MINDSTORMS robot soccer game for children. We developed the *user-guided behaviour-based approach* [16] in order to allow non-expert users to develop their own robots in an easy and fast manner. Indeed, using this approach, children of the age 7–14 were able to develop their own LEGO MINDSTORMS robot soccer players to play in nice and friendly tournaments with 60–90 min of development time! In a user-guided behaviour-based system, it is the system developer who takes care of the difficult robotic problems, while the end-user is working on a higher abstraction level by making the co-ordination of primitive behaviours.

As mentioned, the programming environment for the LEGO MINDSTORMS RoboCup Junior was made with emphasis on allowing children (between 7 and 14 years of age) to develop their own robot soccer players. We found the behaviour-based approach to be an excellent inspiration for achieving this. Especially, we used the concepts of low and high levels of competence, or primitive behaviours and arbitration. We, as developers, provide the primitive behaviours to the children, while they work (play) on a higher level with the arbitration of the primitives. Hence, the difficult task of designing low level primitives that includes sensor interpretation is done a priori by the programmer (so the children get to do the easier and funny part of co-ordination rather than doing low level programming). For instance, the interpretation of analog values on the input channels is done in the primitive behaviours, which might provide the user with a behaviour such as 'Find Ball'. The designer of the system programs the motors to allow the robot to, for example, turn around and stop when receiving values such as 637 and 655 on two of the input channels. But the user is simply co-ordinating the primitive behaviours. This user-guided behaviour-based system is described in further details in [16] (Fig. 11).

The user-guided behaviour-based system for RoboCup Junior is called interactive LEGO football



Robot Doesn't React to Bumpers Activity
Robot Eyes Look at the Ball

UNTITLED .ilf SCORE IN BLACK

Behaviour:1	Go White End
Behaviour:2	Go Forward Once
Behaviour:3	Emit a Sound: HURRAH
Behaviour:4	Search and Go Ball. 6 sec.
Behaviour:5	Go Darker: 3 sec.
Behaviour:6	Find the Ball
Behaviour:7	Go Forward Some Steps: 4

Fig. 11. The programming environment for LEGO MINDSTORMS RoboCup Junior. We used the behaviour-based approach, and developed primitives (the behaviours on the left), and allowed the children to make higher level strategies using these primitives. With this system, children from 7 to 14 years of age were able to develop their own robot footballers within 30–60 min. Copyright H.H. Lund and L. Pagliarini, 1999.

(ILF), and has been used with great success at tournaments during RoboCup'99, MindFest'99, RoboCup EURO 2000, and numerous local events. Figs. 12 and 13 show children participating in the RoboCup Junior game and the performance of the robots that they have developed during maximum 1 h.

5. Breeding robotics

In the case of ILF, the easy development of these robot behaviours is dependent on the available tool. In the RoboCup Junior set-up, the children's task was facilitated by our programming environment, in which children would co-ordinate primitive behaviours rather than hand-coding complex behaviours from scratch. However, one major drawback remains, namely that the children have to be able to read in order to use the system. Therefore, we have worked on developing user-guided evolutionary robotics to fully avoid the necessity to learn syntax and semantics of a programming language before being able to develop robot behaviours. Essentially, we are exploring the concept of *development without programming* by children, and especially at the case of developing robot control systems, so this is a case study of breeding robotics. In breeding robotics, the machines are products of the interaction of the artificial evolutionary process and

Fig. 12. Some children playing with the LEGO robot soccer players that they have developed with ILF within 60 min.

the breeders (in this case children) that try to help, direct and select [13]. The evolutionary robotics approach has shown that in some cases, given a mathematically described fitness function, it is possible to achieve an automatic development of robot controllers. However, it is questionable how one is to construct the mathematical fitness function. So together with my colleagues, I applied an interactive genetic algorithm to the problem of developing robot controllers and achieved an evolutionary robotics approach that allows children without any programming knowledge to develop controllers for LEGO robots [14]. We used neural networks as robot controllers, and found that combining the interactive genetic algorithm with a kind of reinforcement learning — development at the evolutionary time scale combined with life-time development — reduces the development time drastically. Hence, we overcome one of the major drawbacks of the interactive genetic algorithm, namely the development time.

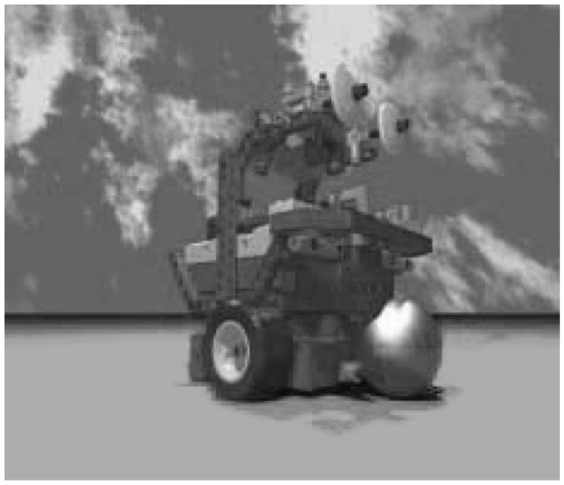


Fig. 13. The LEGO robot soccer player for ILF. We provide building plans for making this robot in order to facilitate the participation in the game. See LEGO robot soccer player building instructions (http://www.mip.sdu.dk/~hhl/RoboCupJr/Build/).

The general idea is a *user-guided evolutionary robotics* approach by which children can develop robot controllers in the simulator by choosing among different robot behaviours that are shown on the screen, and then, when they are satisfied with the simulated robot's behaviour, download the developed control system to the real LEGO robot and further play with it in the real environment.

The user-guided evolutionary robotics approach is inspired by our previous work using interactive genetic algorithms to evolve simulated robot controllers, facial expressions and artistic images (see [23,26]). In this approach, there is no need of programming knowledge, since all the end-user has to provide is a specification of preference of the solutions suggested graphically on the screen. Hence, there is no description of a fitness function, but the selection in the genetic algorithm is performed by the user.

In order to use the user-guided evolutionary robotics approach, it is necessary to simulate the robot in its environment, make selective reproduction in the simulator, and then transfer to the physical robot. As described in [12,18], it is possible to build an accurate simulator that allows very good transfer from simulation to reality by basing the simulator on the robot's own samplings of sensor and motor responses. The disadvantage is that data has to be collected. In the construction of the simulator, this data had to be collected for the different sensors and different motor configurations. For instance, we had to measure the motor response for each individual LEGO robot design that we wanted to use in the simulator. This is the disadvantage of the approach.

The sensor and motor data was collected in a similar way to that described in [12,18], and the collected data was put into look-up tables that is used by the simulator to look up specific sensory readings and displacements of the simulated LEGO robot.

Our first experiments showed that we could develop simple robot behaviours such as obstacle avoidance, line following, etc. for LEGO robots with the user-guided evolutionary approach [14]. Here, children chose a subset (three) of simulated robots in a population (of nine simulated robots) to reproduce generation after generation before downloading the final result to the real LEGO robot. In order to show the feasibility of the user-guided evolutionary robotics approach, we wanted to test it with more complex tasks such as the RoboCup Junior game. Hence, we extended the approach to allow children to evolve complex behaviours for the LEGO MINDSTORMS robots. We call this implementation the Toybots Breeder. Inspired by the successes of previous work on evolvable behaviour-based systems [7] and the user-guided behaviour-based system for the RoboCup Junior [16], we decided these two approaches to be the starting point of our user-guided evolutionary robotics approach for allowing children to develop complex behaviours.

Essentially, we use the primitive behaviours (e.g. ‘Go Forward’, ‘Find Ball’ and ‘Go Midfield’) that we developed for the ILF program as the building blocks in the genotype, and allow the interactive evolution to develop the co-ordination of these behaviours. In the simplest case, the co-ordination can be sequencing a number of the primitive behaviours. In this case, in order to build a simulator, we simulate each of the primitive behaviours, and we simulate the ball movement — using the simulation technique described above. As before, we can now show a population of simulated LEGO MINDSTORMS robots on the screen with the simulation of the field and the ball, and allow the

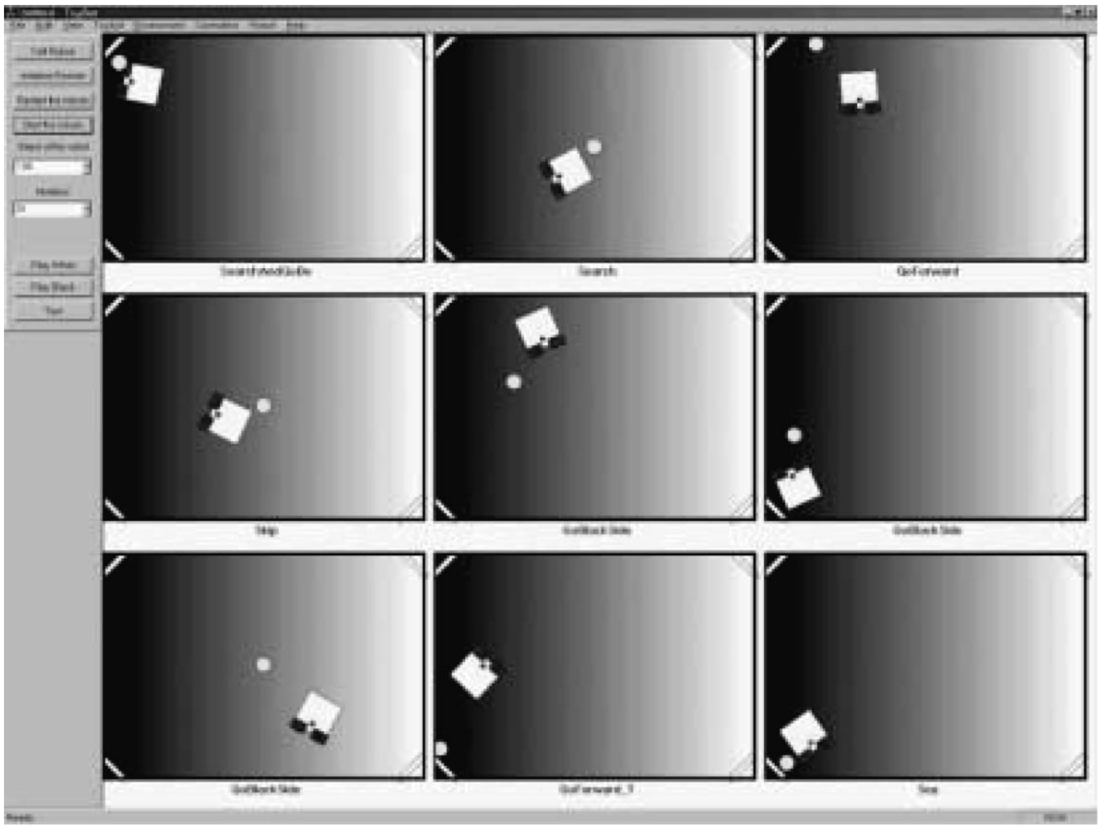


Fig. 14. The Toybots simulator for the LEGO MINDSTORMS RoboCup Junior. Here, children can develop complex robot soccer behaviours before downloading these behaviours to the real LEGO MINDSTORMS robot to be used in competitions. Copyright H.H. Lund, 1999.

children to select the ones that play the kind of soccer that they are interested in, see Fig. 14. User tests in our lab showed that children of 8 years of age were able to develop robot soccer players and enjoy the game with the Toybots Breeder.

There are a number of research issues that have to be addressed in order to ensure feasible user-guided evolutionary robotics. The most important issue is user fatigue: how can we avoid demanding the user to select from a big population for numerous generations. If the selection process becomes too long, the user will experience fatigue. In our case, by using primitive behaviours as the building blocks, we achieve fairly fast evolution, which is essential when children are involved. However, it is not given that the building blocks should necessarily be at this level. For the simple behaviours such as obstacle avoidance and line following, we used connection weights in a neural network as the building blocks, but for the more complex task of robot soccer we found it necessary to increase the complexity of the building blocks to become primitive behaviours.

6. Co-evolutionary Robot Soccer Show (CERSS)

The experiments with user-guided evolutionary robotics for RoboCup Junior showed that indeed children could use evolution to develop interesting robot behaviours. Based on this experience, we decided to develop an educational tool for teaching evolution, and therefore developed the CERSS. CERSS aims at explaining Darwinian evolution along with aspects regarding technology (robotics) and modern artificial intelligence. It is believed that the game might help in understanding of the important biological concept of Darwinian evolution, since it gives the user hands-on experience with the concept, and hopefully makes the concept less abstract for the user. Such hands-on experiments are often impossible in “traditional” biology classes (an exception might be work with the fruit fly, *Drosophila*, which however is a very long process), and is therefore believed to be an important supplement to the more traditional biology classes.

The CERSS allows the user to get hands-on experience with the concepts of evolution, cross-over, mutation, fitness, etc. by looking at how they influence the evolutionary process. This is done by setting different parameters (e.g. selection pressure, mutation rate, cross-over rate and fitness) and then allowing the evolution to run and observe the outcome. Afterwards, one is encouraged to use the experience gained from the first run to modify the parameters and run other evolutionary experiments for obtaining the best possible result.

As was the case with the Toybots Breeder, also the CERSS is an instance of off-line evolutionary robotics, in which the evolution takes place in simulation before the result is transferred to the physical robot. In this case, the physical robot is a Khepera miniature robot, which is circular and measures 55 mm in diameter. The robot has two independent motors connected to small wheels (one on each side of the robot), and eight infra-red sensors that can sense objects in a very short distance. Further, the robot is equipped with a simple, linear camera that can sense object 36° in front of the robot in different grey levels. The controllers developed for playing soccer in the simulator can be downloaded to the physical Khepera robot that can then play the soccer game in a physical arena with a yellow tennis ball and an opponent. For instance, downloading and physical games with the evolved robots took place in Amsterdam during RoboCup European Championships 2000.

In the CERSS, there are two populations (the red population and the blue population) that compete against each other, and who are dependent on each other. This is known as co-evolutionary robotics (see [20]). When one population of robots gets better (e.g. scores more goals), the other gets worse (e.g. more goals are scored against it). This is in some way similar to the natural phenomenon of predators and prey. If the predators start running faster, then the prey must find a new strategy in order to survive. If the prey switches to a new strategy, then the predator must find another, new strategy as well, in order to catch the prey. And so the co-evolution can continue with one strategy after another. In our case, we hope that the populations of robot soccer players can use the co-evolution to find new robot soccer strategies. The underlying controller to be developed is a behaviour-based system where activation flows between layers in a hierarchical behaviour control (see [22]). Comparison tests showed that the co-evolved individuals were more robust in handling different

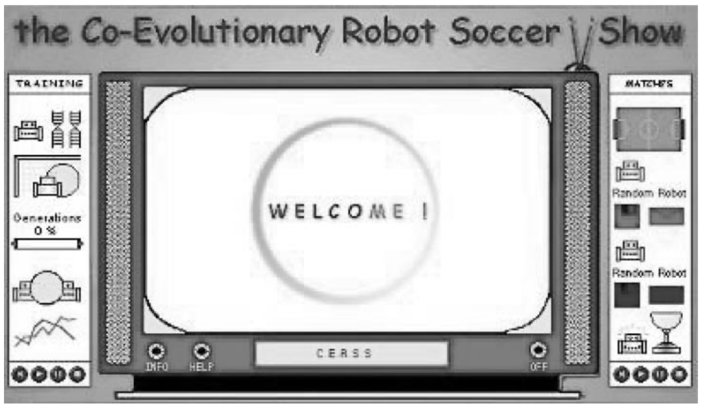


Fig. 15. The entry for the Co-evolutionary Robot Soccer Show. The software can be downloaded for free at the website http://legolab.daimi.au.dk/cerss/.

opponent strategies than individuals evolved with traditional evolution (Fig. 15).

For each population in the co-evolutionary robotics set-up, the user can set and experiment with a number of parameters such number of generations, elitism, number of selected individuals, crossover rate, mutation rate, and number of test steps. Further, the user sets the fitness function deciding number of points for scoring goals, number of (negative) points for scoring own goals, number of points for being fast, etc. The CERSS ran as an Internet competition, and everyday users would submit their best players to the server, which would play them against each other every night in almost 100,000 matches a night, in order to generate the high score list in the morning. Also, the best co-evolved robot controller from numerous experiments was used in a Khepera robot that participated in the Danish Championship in robot soccer 1999, where the evolved robot won four matches against robots that had been hand-coded. Eventually, the evolved controller lost in the final of the championship (Fig. 16).

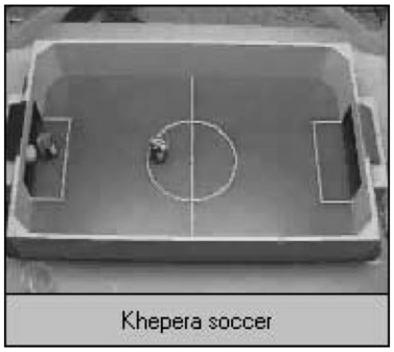


Fig. 16. Evolved Khepera robot soccer player. The red robot is evolved in simulation and the controller is transferred to the physical robot. The blue robot is hand-coded.

7. Context development

Based on the positive experience with the robot soccer experiments, RoboCup has now developed a full RoboCup Junior league. Before RoboCup 2000 in Melbourne, almost 40 school classes used RoboCup Junior as an activity either during school hours or after. However, based on experience and interview with teachers (see [25] for a report describing response by teachers regarding the use of RoboCup Junior), we have noticed an undesirable gender gap. It turns out that most participants for the robot soccer games are boys, probably because of the strong use of “male-dominated concepts” such as technique, soccer, cars, and competition.

Therefore, we introduced other games such as robot dancing and RoboCup Junior Rescue that puts emphasis on combining technical skills with other skills, putting things into the right context, co-operation and performance (rather than competition). Indeed, for the robot dancing, we experienced a much more equal distribution of participants between the two genders (Fig. 17).

Fig. 17. Robot Dance Performance made by children from Australia during RoboCup Junior 2000. Here, the children design both the story line, the environment, the robot, etc. See the RoboCup Junior Official Site (http://www.artificialia.com/RoboCupJr/).

Indeed, the issue of putting things into the right context is crucial in entertainment robotics. For instance, the first robot soccer demonstration would not have resembled soccer play without the stadium with spectators, rolling commercials, sounds, etc. It is therefore important, that we are able to put the robots into the right context, and often giving the user the chance to design the context. So, the user should not only observe the robot or program the robot behaviour. Ideally, the user should have the freedom to design the environment, design the robot morphology, program the robot behaviours, and interact with the robot (Figs. 18 and 19).

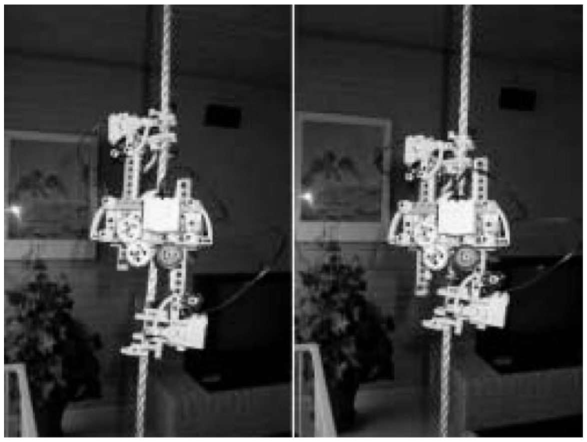


Fig. 18. LEGO robot rope climbing robots for the LEGO MINDSTORMS Book of Records.

Fig. 19. A LEGO MINDSTORMS robot for the cross-country skiing competition.

Another example that works toward such possibilities is the LEGO MINDSTORMS Book of Records that allows children to explore development of robot morphology and robot programming, and ideally also context development by adding new record categories. In the LEGO MINDSTORMS Book of Records, we developed games such as rope climbing, cross-country skiing, shot put, Robin Hood precision shooting, etc.

8. Conclusion

The numerous practical experiments described here have shown that it is possible to use modern artificial intelligence in entertainment robotics. However, most often the methods have to be re-designed in order to meet the demand of providing both fun and educational experiences for the users. Hence, we have developed a number of user-guided approaches based on behaviour-based systems, evolutionary computation, neural networks, and multi-agent systems. Further, research in the field of embodied artificial intelligence suggest that we should put emphasis not only on optimisation of the robot controllers but look at the interaction between controller, morphology, material, and environment. So we should allow the user the freedom to manipulate as many of these parameters as possible, and not only interact with a "static" robot or only develop the robot controller. This corresponds with the educational practises of allowing the user/student to construct the artefact, the functionality and the context in which the artefact is placed.

For further reading see [17].

Acknowledgements

Luigi Pagliarini collaborated with the author on making many of the applications described in this article, he engaged in numerous discussions about edutainment with a psychological view, and he provided valuable feedback. A number of experiments were developed in collaboration with students. They include Kasper Støy, Esben Østergaard, Jesper Blynel, Mikkel Yde Kjær, Rene Schalburg, Henrik Madsen, Jesper Rasmussen, Jakob Vesterstrøm, and M. Rooker. All views presented in this article are those of the author, and do not necessarily present those of any sponsor or funding agency.

References

[1] R. Arkin, Behavior-Based Robotics, MIT Press, Cambridge, MA, 1998.
[2] V. Braitenberg, Vehicles: Experiments in Synthetic Psychology, MIT Press, Cambridge, MA, 1984.
[3] R.A. Brooks, A robust layered control system for a mobile robot, IEEE Journal of Robotics Automation 2 (1) (1986) 14–23.
[4] R.A. Brooks, Evolutionary robotics where from and where to, in: Evolutionary Robotics: From Intelligent Robots to Artificial Life ER'97, AAI Books, Ont., Canada, 1997, pp. 1–19.
[5] D. Cliff, I. Harvey, P. Husbands, Explorations in evolutionary robotics, Adaptive Behavior 2 (1) (1993) 73–110.
[6] D. Floreano, F. Mondada, Automatic creation of an autonomous agent: genetic evolution of a neural network driven robot, in: D. Cliff, P. Husbands, J. Meyer, S.W. Wilson (Eds.), Proceedings of the 3rd International Conference on Simulation of Adaptive Behavior (SAB'94), From Animals to Animats, Vol. 3, MIT Press, Cambridge, MA, 1994.
[7] W.-P. Lee, J. Hallam, H.H. Lund, Learning complex robot behaviors by evolutionary approach, in: A. Birk, J. Demiris (Eds.), Proceedings of the 6th European Workshop on Learning Robots, LNAI 1545, Springer, Heidelberg, 1997.
[8] F.L. Lewis, Neural network control of robot manipulators, IEEE Expert/Intelligent Systems and their Applications 11 (3) (1996) 64–75.
[9] H.H. Lund, AI in children's play with LEGO robots, in: Proceedings of the AAAI 1999 Spring Symposium Series, AAAI Press, Menlo Park, CA, 1999.
[10] H.H. Lund, Robot soccer in education, Advanced Robotics Journal 13 (8) (1999) 737–752.
[11] H.H. Lund, J.A. Arendt, J. Fredslund, L. Pagliarini Ola, What goes up, must fall down, in: Proceedings of the Artificial Life and Robotics 1999 (AROB'99), ISAROB, Oita, 1999, pp. I-9–I-15.
[12] H.H. Lund, O. Miglino, From simulated to real robots, in: Proceedings of the IEEE 3rd International Conference on Evolutionary Computation, IEEE Press, NJ, 1996.
[13] H.H. Lund, O. Miglino, Evolving and breeding robots, in: Proceedings of the 1st European Workshop on Evolutionary Robotics, Springer, Berlin, 1998.
[14] H.H. Lund, O. Miglino, L. Pagliarini, A. Billard, A. Ijspeert, Evolutionary robotics — a children's game, in: Proceedings of the IEEE 5th International Conference on Evolutionary Computation, IEEE Press, NJ, 1998.
[15] H.H. Lund, L. Pagliarini, Robot soccer with LEGO MINDSTORMS, in: Asada (Ed.), Proceedings of the RoboCup'98, Springer, Heidelberg, 1999.
[16] H.H. Lund, L. Pagliarini, RoboCup Jr. with LEGO MINDSTORMS, in: Proceedings of the International Conference on Robotics and Automation (ICRA 2000), IEEE Press, NJ, 2000.
[17] O. Miglino, H.H. Lund, M. Cardaci, Robotics as an educational tool, Journal of Interactive Learning Research 10 (1) (1999) 25–48.
[18] O. Miglino, H.H. Lund, S. Nolfi, Evolving mobile robots in simulated and real environments, Artificial Life 2 (4) (1996) 417–434.
[19] T.M. Mitchell, S.B. Thrun, Explanation-based neural network learning for robot control, in: S.J. Hanson, J.D. Cowan, C.L. Giles (Eds.), Advances in Neural Information Processing Systems, Vol. 5, Morgan Kaufmann, Los Altos, 1993, pp. 287–294.
[20] S. Nolfi, D. Floreano, Co-evolving predator and prey robots: do 'arm races' arise in artificial evolution? Artificial Life 4 (4) (1998) 311–335.
[21] S. Nolfi, D. Floreano, O. Miglino, F. Mondada, How to evolve autonomous robots: different approaches in evolutionary robotics, in: R. Brooks, P. Maes (Eds.), Artificial Life, Vol. IV, MIT Press, Cambridge, MA, 1994, pp. 190–197.
[22] E.H. Oestergaard, Evolving complex robot behaviour, Master's thesis, University of Aarhus, May 2000.
[23] L. Pagliarini, H.H. Lund, O. Miglino, D. Parisi, Artificial life: a new way to build educational and therapeutic games, in: Proceedings of the Artificial Life, Vol. V, MIT Press, Cambridge, MA, 1996.
[24] R. Pfeifer, C. Scheier, Understanding Intelligence, MIT Press, Cambridge, MA, 1999.
[25] E. Sklar, J. Johnson, H.H. Lund, Children Learning From Team Robotics: RoboCup Junior 2000, Educational Research Report, Department of Design and Innovation, Faculty of Technology, The Open University, Milton Keynes, UK, 2000.
[26] V. Vucic, H.H. Lund, Self-evolving arts — organisms versus fetishes, Muhely (The Hungarian Journal of Modern Art) 104 (1997) 69–79.
[27] S. Woodcock, Game AI: the state of the industry, in: Game Developer, Freeman, New York, 1999.

[28] S.-Y. Yoon, B.M. Blumberg, G.E. Schneider, Motivation driven learning for interactive synthetic characters, in: Proceedings of the 4th International Conference on Autonomous Agents, ACM Press, New York, NY, 2000.

Henrik Hautop Lund is a professor in sensor and actuator technology at the Maersk Mc-Kinney Moller Institute for Production Technology, University of Southern Denmark, Odense. He holds a PhD in computer systems engineering, an MSc in computer science, and a BSc in mathematics. In 1997, he founded the LEGO Lab at the Danish National Center for IT-Research/University of Aarhus. The LEGO Lab was financed by the Danish Government and Danish Industries, such as LEGO. The LEGO Lab was occupied with development of LEGO MINDSTORMS robots and how to put artificial intelligence into our daily life. In 2000, he moved his activities to become professor at the Maersk Institute. The activities of his group are sponsored by the LEGO group, with whom the research group has close relations. He worked previously as research associate from 1992 to 1995 at the Institute of Psychology, The National Research Council, Rome, Italy, doing research in the fields of artificial life, neural networks, and evolutionary computation. Further, he worked as research associate during 1996 and 1997 in the Department of Artificial Intelligence at University of Edinburgh, UK, doing research on biologically inspired robotics and evolutionary robotics. He has been interested in the relationship between robot behaviour and robot morphology since 1992. He has published more than 40 peer reviewed papers in international, scientific journals and conference proceedings.