RoboCup Jr. with LEGO Mindstorms

Henrik Hautop Lund

LEGO Lab
University of Aarhus
8200 Aarhus N, Denmark
http://legolab.daimi.au.dk

Luigi Pagliarini

LEGO Lab
University of Aarhus
8200 Aarhus N., Denmark
http://legolab.daimi.au.dk

Abstract

During RoboCup'99 in Stockholm, we arranged the first RoboCup Jr. Here, the aim 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 behavior-based approach 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 minutes of development time! In a user-guided behavior-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 coordination of primitive behaviors. Further, for the LEGO Mindstorms RoboCup Jr. game, we developed a field and a smart ball (with IR emitters) which allowed easy navigation and detection.

1 Introduction

During RoboCup'99 in Stockholm, we arranged the first RoboCup Jr. A main part of the RoboCup Jr. consisted of a robot soccer game with LEGO Mindstorms robots which allowed children of the age 7-14 to develop their own robot soccer players to play small tournaments within 60-90 minutes of development time. There are many fundamental problems that have to be solved in order to make such a robotic game available for children to develop and play with. Among these and probably most fundamentally, we find the problem of traditional programming languages demanding the learning of both syntax and semantics of the programming language to be used before the user can start to develop his/her own system. Further, after finally being able to develop the system, traditionally the user has to go through a long and tedious debugging phase before achieving the system and performance in mind. Since robotic systems often inherit their programming language from traditional computer systems, the robotic systems also inherit the syntax & semantics knowledge and debugging problem from the computer systems. Further, the problem of having general users to develop their own robotic systems is worsened by the integration of hardware components such as sensors and motors, since the control of these external devices traditionally demands extensive engineering and control knowledge. It is by no means an easy task to understand sensor responses, sensor fusion, motor characteristics, environmental noise, etc.

In this paper, we devise a new robot control methodology, *user-guided behavior-based robotics*, which aims at avoiding the problems mentioned above and allowing the general user (e.g. a child) with no previous robotics and programming knowledge to develop his/her own robotic system in a very fast and reliable manner. User-guided behavior-based robotics is based on recent developments in behavior-based robotics, and our general studies on using different adaptive robotic techniques (e.g. neural network controllers, interactive evolutionary robotics, building brains and bodies techniques, etc.) to construct new robot development tools for non-expert users. We show the feasibility, reliability and robustness of user-guided behavior-based robotics with the RoboCup Jr. case study, which showed how children of the age 7-14 were able to develop their LEGO Mindstorms robot soccer players to score goals and participate in friendly and fun tournaments within 60-90 minutes. Already here it must be noted that we can only describe the very positive experiences, but yet have no statistical data on the possible advantage of using our system though

we would have preferred to present such kind of data.

2 Behavior-Based Robotics

It is our hypothesis that adaptive robotic techniques such as behavior-based systems, neural network controllers, interactive evolutionary robotics, building brains and bodies techniques, etc. are suitable tools in trying to alleviate the problems involved when trying to allow non-expert users to develop their own robots. Here, we will concentrate on the use of behavior-based systems. The first behavior-based systems were developed in the mid-80s by Rodney Brooks [3] as a response to the artificial intelligence tradition of a hard division between hardware and software development, and the sense-represent-plan-act cycle that most artificial intelligence robotic systems implemented. Brooks' behavior-based systems shifted the emphasis from such a functional decomposition to a behavioral decomposition. Both approaches use the classical problem solving technique of divide-and-conquer, but with the behavioral decomposition one divides the problem into behavioral components that all have access to sensors and actuators. From a theoretical point of view, this means that the behavior-based approach promotes an explanation of intelligence that relies on the interplay between the system (animal, agent, robot, etc.) and the environment, and the embodiment of the system. Hardware has influence on all levels and there is no possible abstraction to a pure cognitive level (in contrast with the functional decomposition).

The assumption of behavior-based systems is that complex behaviors can emerge from the combination of simple behaviors. In Brooks' original subsumption architecture [3], he develops a layered structure that allows the hand-coding of one level of competence after another in an increasing order to achieve higher and higher levels of competence. Both the layers of behaviors and the integration of the behaviors are handcoded by the developer.

During the following few years after Brooks' invention of the behavior-based approach, a number of researchers like L. Steels [10], P. Maes [9], R. Arkin [1], etc. developed different architectures for the behavior-based approach to robotics. The architectures use different representations and different behavioral coordination methods. In general, simple behaviors are handcoded, and the behaviors are coordinated through competitive methods (priority-based coordination, action-selection coordination, voting-based coordination) or cooperative methods (vector summation), see e.g. [2].

In all the behavior-based approaches, the developer of the system needs to have an extensive engineering knowledge about the robot hardware and computer programming knowledge in order to be able to design the single behaviors and the coordination between the behaviors. In a recent evolutionary behavior-based approach [4], we have tried to alleviate this problem by allowing an evolutionary algorithm to first evolve the single behaviors, and thereafter evolve the arbitration between the simple behaviors. In this approach, the user needs only make the task decomposition and describe fitness functions for each simple behavior and each arbitration. But still this demands some a priori robotic knowledge.

In general, the reliance on smart engineers and computer scientists in the development of behavior-based robotic systems might pose a problem in scaling up to complex robot behaviors. However, as we will show below, it is our belief that, with small manipulations, the approach can be used to allow non-expert users to develop their own robots in a fast and easy manner.

3 User-Guided Behavior-Based Robotics

We [6] have previously explored how to convert adaptive robotics techniques into user-guided techniques, in order to allow non-experts to develop robots. In the previous case, we used evolutionary robotics and turned it into a user-guided evolutionary robotics approach, in which the user is deciding which robots in a population of robots should be allowed to reproduce. Hence, instead of having to design a fitness function in mathematical terms, the user is simply looking at the robot performances and chooses which ones to reproduce. We [6] have previously shown how this approach can be used to develop LEGO robots with simple behaviors, such as obstacle avoidance, line following, wall following, etc.

Now, we wanted to go towards more complex and interesting behaviors, such as robot soccer behaviors. It was our idea to use a similar approach, but since we had no record of such complex behaviors yet being developed with evolutionary robotic approaches, we looked to another approach using behavior-based robotics. This was partly because that in previous evolvable behavior-based experiments [4] we found indications of a possible future merging with the user-

guided evolutionary robotics approach would be fruitful (in fact, this is now work in progress). Hence, we wanted to develop a new kind of behavior-based system, namely a user-guided behavior-based system.

In the user-guided behavior-based robotics approach, the designer is developing the primitive behaviors, which include all the low level processing and integration of sensors and motors. The end-user is making the coordination of the primitive behaviors in order to have the global robot behavior in mind to emerge. Hence, the end-user is working on a high abstraction level, and does not have to concentrate his/her efforts on e.g. how to integrate sensors, how to interpret analogue values, how to send commands to the motors, how to incorporate/interpret noise, etc. This is all left to the designer of the system, who is hopefully a professional (engineer, computer scientist, roboticist) in this field, and therefore by profession is capable of making a suitable design of primitive behaviors. In the case of robot soccer, the end-user is working as a coach, telling the robot soccer player what kind of behaviors to perform - essentially like a coach would tell the left wing soccer player to run back, get the ball, run up along the left flank, and when reaching the end line to pass the ball.

There is no need to have extensive a priori expert knowledge about robotics when using the user-guided behavior-based robotics system, since the complex robotic problems are handled in the design of the system by the system designer. However, there is still the problem that the user has to understand what a specific, primitive robot behavior actually does. Therefore, for the LEGO Mindstorms robot soccer game for children, we provided video-sequences of all primitive behaviors, so that the user (child) can watch each primitive behavior and get an idea about what the robot will actually do when performing a specific behavior. This has the further advantage that children with poor (or no) reading capabilities are able to use the system by watching the visualization and using this when selecting primitive behaviors. Specifically, in Stockholm we did not have the possibility to have a Swedish translator all the time, so there were minor problems with the some of the children not understanding their coach well, but then they used the video clips to better understand the different, available commands. Currently, the programming environment provides translation into English, Swedish, Danish, and Italian.

In the LEGO Mindstorms robot soccer game for children, we made a simple coordination mechanism available in the beginner's level, since we wanted all children to be able to participate. The coordination mechanism is a simple selection of single primitive behaviors to be put in sequence. Hence, there is no subsumption, voting, or similar in this case study. It should be possible to implement such a more advanced coordination mechanism, but, in this case, our first concern was an easy system for children. So the children select primitive behaviors to be put in a sequence and this sequence is then looped over and over again in the LEGO Mindstorms robot soccer player.

Figure 1: Video sequence of a primitive behavior. © LEGO Lab 1999.

We provided a number of primitive behaviors to the user. These include eyes behaviors, reaction to bumping, going to specific regions of the field, turning and moving forward, finding the ball, circling the ball, going in specific directions on the field, etc. The whole list of primitive behaviors is seen on the left side of figure 2, which is a screen dump of the programming environment.

4 Robot and Environment Set-up

There are numerous technical problems that have to be solved before it is possible to make a RoboCup Jr. tournament for children. Especially, it is important that the robots and their interaction with the environment are easy to understand. We chose to use LEGO Mindstorms robots for the RoboCup Jr. tournaments during RoboCup'99, since these robots are fairly simple and every child feels comfortable with LEGO bricks. However, it is not trivial to make a LEGO Mindstorms robot play a robot soccer game.

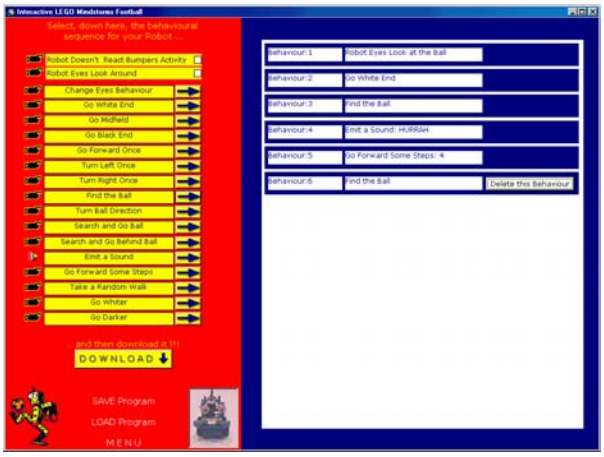

Figure 2: The programming environment. On the left is displayed all the primitive behaviors that the user can select to put on the right side in the control program. To the left of each primitive behavior is show a small video camera icon. When clicking this icon, a small video sequence will display the robot behavior. © LEGO Lab 1999.

Figure 3: One of the LEGO Mindstorms robots used for the RoboCup Jr. set-up. © P. Petrovic 1999.

For RoboCup'98, we developed a LEGO Mindstorms robot soccer demonstration [7, 8, 5] based on the availability of an overhead camera and a hardware vision system. Further, the huge set-up for RoboCup'98 included a whole stadium made out of LEGO with small cameras, rolling commercials, score board, spectators that made the "wave", etc. But for the RoboCup Jr. we found it essential to have a minimal set-up excluding things like overhead cameras in order to allow the children to understand the set-up and the game.

For the RoboCup Jr., the soccer player is a LEGO Mindstorms robot that has three light sensors and two switch sensors. Two light sensors are used to detect the ball, and one light sensor is used to detect the approximate position on the field. The switch sensors are used to detect bumps into the walls or the opponent robot. The robot soccer player has two LEGO motors that are connected to the wheels. The last output channel of the LEGO Mindstorms RCX is used to control the movements of the robot's eyes. Giving the robot eyes (or, in general, facial expression) seems to provide more affection from the children towards their robot soccer player, so this is another important aspect to investigate when making a robot soccer play for young children. In general, in the LEGO Lab we view it as important to move towards a better human-robot interface, and this seems to be facilitated with e.g. robot facial expressions.

In order to facilitate easy navigation with simple sensors (such as a light sensor), we made a special field for the RoboCup Jr. The field for LEGO Mindstorms RoboCup Jr. is a grayscale surface printed on an oversize A0. It is simply a gradient from black to white, and using a LEGO light sensor underneath the robot one can navigate around the field and e.g. find the goal of the opponent. The navigation on such a surface is very robust with the LEGO Mindstorms robot soccer player. But again, this kind of sensor interpretation and processing is done by the developer of the user-guided behavior-based system and kept hidden from the end-user. The end-user does not need to know what kind of processing is taking place within the primitive behaviors, and therefore does not need to be a professional roboticist.

The detection of a ball can also be quite difficult when using nothing else than two simple LEGO Mindstorms light sensors. The LEGO Mindstorms light sensors emit IR light and have an IR detector, so they can be used to distinguish colours at a very short distance (e.g. 0-5mm.), but are not well suited for detecting objects at longer distances. However, clever engineering can alleviate this problem. If one emits IR signals of the same wavelength as the one detected with the detector in the LEGO Mindstorms light sensor, then it is possible to sense such signals over a fairly long distance. But IR emitters often emit with a small spreading angle, so more than one IR emitter might be necessary in order to cover the surface of an object. In our case, we designed a ball out of transparent plastic, and planted 20 IR emitters inside the ball at positions so that we ensured coverage of all angles. In order to be able to emit stronger IR signals, we made

a small pulse circuit that pulses the IR emitters at a high frequency. The ball draws its current from 4 rechargeable batteries placed inside the ball, and some small weights are carefully positioned within the ball in order to balance the ball.

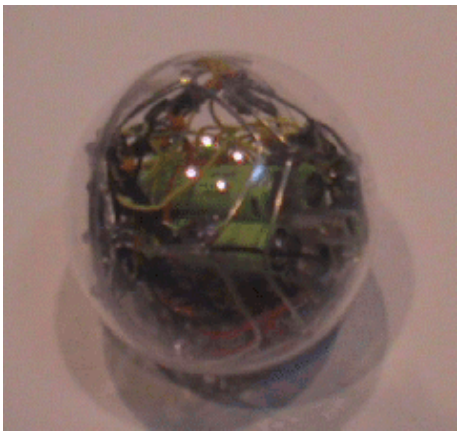

Figure 4: The smart ball developed for the LEGO Mindstorms RoboCup Jr.'99. © LEGO Lab 1999.

Finally, we had to solve the problem of recharging the batteries inside the ball. This is done by using the screws that holds the transparent plastic ball together. The batteries are placed around these two screws, so we built a LEGO recharger (see figure 5) that charges the batteries through the screws when the ball is placed in the LEGO recharger.

Figure 5: The LEGO ball recharger. © P. Petrovic 1999.

With fully charged batteries, the ball can be detected by the LEGO Mindstorms light sensors at approximately 2 meters distance. However, since the signal from the ball is pulsed and the LEGO Mindstorms light sensors are reading at a higher frequency (every 3 ms), the robot might read the light sensor when the emitter pointing towards the robot is not turned on. Hence, one needs to integrate over e.g. 3 readings. But again, this is a thing that is left for the designer of the system to figure out, and not a job for the end-user. The end-user is simply using primitive behaviors such as Find Ball, Search and Go Ball, Circle Ball, etc.

5 RoboCup Jr. Experience

Each day in Stockholm during RoboCup'99, children between 7 and 14 years of age were divided into groups of 2-4 children in each group. Each group was given a pre-made LEGO Mindstorms robot, as described above. Each group of children had a coach, who would give the children a brief (e.g. 10 minutes) introduction to the game, the robot, and the programming environment. Afterwards, the children started to program their robots with the user-guided behavior-based system for the robot soccer game. Especially, most groups would start by looking at the video sequences to understand the meaning of the available primitive behaviors. Afterwards, they would normally start a cycle of making small programs, downloading and testing the robot behavior, and refining the program. Within 20 minutes or so the children were able to score their first goal with the robot. In order to keep the attention of the children, such a fast success experience seems to be crucial. The new and easy robot programming language is essential for the success. Here, the children are not concerned with the design of primitive behaviors, but only with the combination of the primitive behaviors. They work on a higher level and design the soccer strategy of the single player, rather than design the low levels of competence such as vision processing, sending messages to motors, etc. This means that even very young children can understand and enjoy this robot soccer game.

All days of the RoboCup Jr. event, after 90-120 minutes of development, we had very nice, friendly competitions ... more in the spirit of participating, rather than winning, though some children were very keen of making the perfect robot to perform well in the games. The first day, the final ended 4-2, the next day it ended 10-7, and the final day it ended 2-1. Approximately the same amounts of goals were scored in the qualification matches, so the children definitely managed to make goal-scoring robots with the user-guided behavior-based system.

It is interesting to notice that the participating girls were at least as enthusiastic about the game as the boys, even though we played the (normally) male dominated game of soccer. We were happy to notice this,

since it is important to reach the girls and find ways to transfer technology knowledge and enthusiasm to girls as well as boys.

Unfortunately, we do not have any statistical measurements to compare this approach with other approaches, but we were definitely surprised by the children's ease and enthusiasm of using the system. Seeing children down to the age of 7 develop their own robot soccer players with 60-90 minutes gives us informal evidence that the user-guided behavior-based system facilitate the development of robots by non-expert users. The feedback from the users also verifies this:

Oksana and myself definitely had a great time and enjoyed the game very much! (which is very surprising because I do not like soccer as a game and has never been interested in it. However, the lego robot soccer tournament gave me with a different experience.) It is a good fun to program a robot and the program is indeed easy to use.

In the whole it was a great experience and we enjoyed it very much. Robots are our future and it is very exciting to see the first steps in robots development. Also it is very important to get children interested and involved in the process - and you were absolutely successful in this task. Thank you for giving children a chance to try.

11/8/1999, Elena Prokopenko (Australia), mother of Oksana

6 Conclusion

It seems evident that the problem of learning both syntax and semantics of a traditional programming language has to be solved if robot development is to be given free to non-experts. Especially, this is the case when fast development is desirable. For this purpose, we have developed the user-guided behavior-based system and tested this system with the RoboCup Jr. setup. The user-guided behavior-based system allows the user to make coordination of primitive behaviors designed a priori by the system developer. Hence, the user works on a high abstraction level, while it is the system developer (e.g. engineer, computer scientist, roboticist) who is designing the low level behaviors which include the difficult parts of communication to sensors and motors, sensor pre-processing, noise-interpretation, etc.

The RoboCup Jr. experience tells us that the user-guided behavior-based system allows the users (children of age 7-14) with no previous robotics experience to develop their own robots in a very fast manner. Indeed, the children are able to develop very complex robot behaviors such as the different robot soccer behaviors. However, it must be noted that the user-guided behavior-based system is by no means limited to a robot soccer game or a tournament set-up. Here, we simply used RoboCup Jr. as a case study. In future, we will explore the development of other complex robot behaviors by non-expert users using the user-guided behavior-based approach.

References

[1] R. Arkin. Motor Schema-Based Mobile Robot Navigation. *International Journal of Robotics Research*, 8(4):92–112, 1989.

[2] R. C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, MA, 1998.

[3] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE J. Robotics and Automation*, RA-2(1), 1986.

[4] W.-P. Lee, J. Hallam, and H. H. Lund. Learning Complex Robot Behaviors by Evolutionary Approach. In A. Birk and J. Demiris, editors, *Proceedings of 6th European Workshop on Learning Robots, LNAI 1545*, Heidelberg, 1997. Springer Verlag.

[5] H. H. Lund, J. A. Arendt, J. Fredslund, and L. Pagliarini. Ola: What Goes Up, Must Fall Down. *Artificial Life and Robotics*, 4, 1999.

[6] H. H. Lund, O. Miglino, L. Pagliarini, A. Billard, and A. Ijspeert. Evolutionary Robotics — A Children's Game. In *Proceedings of IEEE Fifth International Conference on Evolutionary Computation*, pages 154–158, NJ, 1998. IEEE Press.

[7] H. H. Lund and L. Pagliarini. LEGO Mindstorms Robot Soccer. A Distributed Behaviour-Based System. 1999. To be submitted.

[8] H. H. Lund and L. Pagliarini. Robot Soccer with LEGO Mindstorms. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II, LNAI 1604*, Heidelberg, 1999. Springer Verlag.

[9] P. Maes. Situated agents can have goals. In P. Maes, editor, *Designing Autonomous Agents*, Cambridge, MA, 1990. MIT Press.

[10] L. Steels. Towards a theory of emergent functionality. In J. Meyer and S. W. Wilson, editors, *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior (SAB90)*, Cambridge, MA, 1991. MIT Press.


JVC