ADAM: ADaptive Autonomous Machine

Daan C. van Oosten

Dutch Open University, P.O. Box 2960
6401 DL Heerlen, The Netherlands
e-mail:voo@ouhi08.ouh.nl

Lucas F.J. Nijenhuis en André W.P. Bakkers

Electrical Engineering Faculty, Control Laboratory
University of Twente, P.O. Box 217
7500 AE Enschede, The Netherlands
e-mail: bks@rt.el.utwente.nl

Wiek A. Vervoort

Informatics Department, Systems Programming and Architecture
University of Twente, P.O. Box 217
7500 AE Enschede, The Netherlands
e-mail: vervoort@cs.utwente.nl

Abstract

This paper describes a part of the development of an adaptive autonomous machine that is able to move in an unknown world, extract knowledge out of the perceived data, has the possibility to reason, and finally has the capability to exchange experiences and knowledge with other agents. The agent is not pre-programmed by its designer but was given simple rules of life i.e. what is good and what is bad. By evaluating its sensor inputs these rules of life were transformed into a rule based reactive system. Simulations of the system showed that the agent is able to learn by its own experience. By representing the learned knowledge in an appropriate way, the acquired knowledge could be judged on its effectiveness and also this knowledge could be shared with other, less experienced, agents.

1. Introduction

It is a great challenge to develop creatures that show intelligent behaviour and have a learning capacity. Since many years researchers are trying to give (artificial) life to beings that have autonomous behaviour [1-9, 11-14, 17-21]. But all trials so far have a major drawback. The developers put in these creations too much of their own knowledge in more or less static structures. As a consequence the creations do not come to artificial life in an appropriate way. That is to say their learning capacity is limited and they cannot survive in a strongly changing environment.

To meet this goal the following demands have to be met by the system: The system may not incorporate human world knowledge, everything the system learns must be based on own experiences. The agent starts as a "tabula rasa". Learning how to move is based on the state of mind of the agent. This state of mind resembles natural feelings of creatures like happiness, loneliness, sadness, etc. In order to store the information the system learns, a "dynamic data structure" is needed that enables the system, not only to store all kind of information, but new behaviours as well.

0-8186-7695-7/96 $5.00 © 1996 IEEE
Proceedings of EUROBOT '96

1.1 The new approach

The agent, as we want to see it, has a layered brain. One of the layers is the so called rule-based reactive system. It enables the robot to wander in an unknown environment. The rule-based reactive system consists of three functional blocks. The first block filters several conditions out of the raw sensor data, these conditions are used by the system to encode the situation. The last block translates an action in the corresponding actuator values. The block connecting these two blocks must learn associations between situations and actions. For this block, a simple learning mechanism is needed, based on reinforcement learning systems, that evaluate each situation action pair based on an internally generated reinforcement signal. Experiences of the system are stored by combining situations with actions. Simulations show the robot is able to learn the correct behaviour. It formulates "life-rules" that contain information and knowledge usable at a higher level of the brain.

2. Rule-based reactive system

Two words comprise the major problem of intelligent behaviour; when and what. When must the agent do something and what should it do? Natural creatures are performing all the time a process of answering this question, consciously or unconsciously. Looking at their motoric system, some situations cause a pre-defined action like a reflex, this is an innate behaviour. More complex innate behaviours are ascribed to instinct. Other behaviours have the same affect as reflexes but can be learned; these are called conditioned reflexes. Consciousness and reason have almost no effect on these behaviours, in contrast to other behaviours that require a complicated decision making process. Reflexes enable the agent to respond quickly to specific situations. Obviously these reflexes contribute to the changes for self-preservation of the agent.

2.1 Life-rules

Looking in a system manner to reflexes, the following remarks can be made: A system that can perform motoric actions is able to endanger itself, e.g. by hitting a wall. Quick reactions are needed to avoid such situations. The response should be as quickly as possible, hence almost no complex thinking process is permitted. In general, such "reflexes" should have a fixed response that can be either pre-defined or learned. In correspondence with living creatures such reflexes can be activated at all times. These kind of behaviours can be labelled "reactive behaviours" and the system that is comprising such behaviours can be called a "reactive system". If a system is build with increasing complexity and intelligence, the reactive system can be viewed as the lowest level in a hierarchy of intelligent behaviours. As it uses almost no complex decision making process, the reaction time can be very short. The answers to the previously defined questions will be called rules. A rule gives a direct answer what to do in a specific situation. A verbalised version of such a rule can be stated as follows: "IF (moving to a wall) THEN (turn left). This rule gives a direct answer when to perform what action. Once the situation is recognised, no time is needed to think about this situation, the answer is available immediately. A reactive level containing several rules can produce a system that is able to maximise the system's change for self-preservation in a specific environment. Hence, these rules can be regarded as the "rules-of-life" or "life-rules" for this particular system. "A robot that develops associations between specific sensory inputs and responses which are driven by learning criteria provided by a pre-defined value system can develop anticipatory behaviour as an emergent property". [15]. A behaviour system can be formed with condition-action pairs. Hence, a rule is formed by connecting a condition with an action. Rules can be stored as plain data and a process can continuously test conditions and -if found true- perform the corresponding action of the rule. Now, imagine several agents working together, one of them has learned a new life-rule. If this rule can be transferred using some means of communication, this information can have direct effect on the behaviour of other agents. A life-rule is clear and compact and is easy to handle. Like people could transfer a life-rule saying "if a car approaches, don't cross the street", agents could transfer a life-rule saying "IF (moving to a wall) THEN (turn to left)".

2.2 Conditions and actions

Life-rules can be regarded as associations (see quotation McFarland) between conditions and actions. These associations can be implemented in several ways. Also these associations can be generated dynamically. This opens the possibility to put a mechanism between all conditions and possible actions that will "learn" these associations. Each association has a strength that varies during its lifetime but for now only weak and strong associations will be mentioned. A strong association signifies that there is a strong relation between the associated conditions and actions. At a system level this is expressed by a high positive or negative reward for the chosen action. At an agent's level this action causes strong positive or negative feelings. A weak association signifies a weak relation between some conditions and an action. This is possible when in a real-world there is no relation between the two components or it depends highly on each specific real-world situation. On the other hand it is possible that the mechanism is inadequate to learn the proper associations.

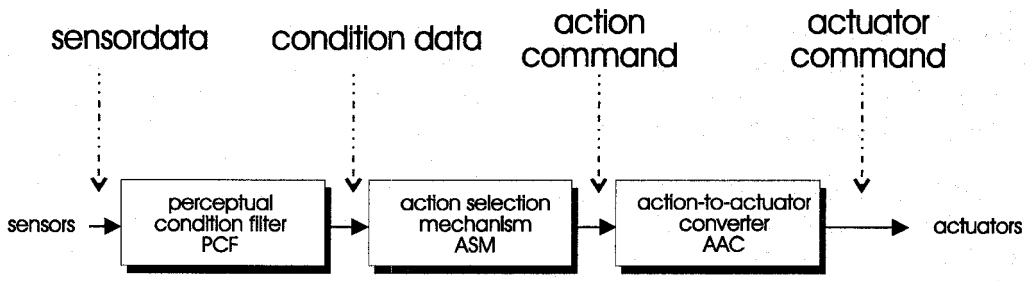

Figure 1: The functional model of the rule-based reactive system.

2.3 Functional model

The mechanism that makes the associations between conditions and actions (the middle block in figure 1) is preceded by a functional block that derives the condition from the sensor data. The chosen actions are translated in another functional block that will return the appropriate actuator commands. The first block in this model is the perceptual condition filter (PCF). The information from the sensor data is represented as perceptual conditions. The designer will have great influence in the design of this filter and the selection of the sensors. Sensor processing is only allowed in this block. Another functional block is necessary to translate the chosen action to the appropriate actuator command is called the action-to-actuator converter (AAC). If an actuator is controlled by a voltage or current signal the possible number of actions is extremely large. In the same way as in the perceptual conditions filter a reduction of possible actions can be obtained. Also in the AAC functional block, the designer can apply all kind of techniques to obtain a small action space with great functionality. States of the system, like speed of a motor or battery voltage level, can be offered to the system as conditions too. Thus, the system can make its decisions based on external and internal information. The action selection mechanism (ASM) connects the "input" block and the "output" block based on the before mentioned life-rules. Levels above the reactive level also belong to the action selection mechanism. Thus, action selection not only incorporates selecting the best action from a number of possible actions (arbitration) but, for example, learning sequences and reasoning about behaviours belong to it as well. The action selection mechanism must select the best action for every condition. The life-rules are used to make this selection, therefore one task of this mechanism is to apply the existing rules. As our goal is to make a learning system, life-rules should be created dynamically. Only in this way a system can be made that generates behavioural code dynamically and uses a dynamic data structure to store this information. For this two additional tasks are necessary. First, it must learn new rules and second it should evaluate the existing rules. Life-rules must be checked on their correctness through evaluation. But what should this evaluation be based on? Some existing systems use a feedback signal from the system, but a feedback signal is not always available. Other systems define the goals the agent should reach, but these goals are from an observer's point of view. A solution close to nature is to define a state of mind for the agent. The state of mind expresses that an agent wants to "feel happy", "not lonely", etc. It closely resembles to existing feelings of intelligent beings and it could render some emergent behaviours that could not be expected by an observer. In order to define a state of mind comprising these feelings it must be able to sense them. Thus the state of mind is only related to things the agent can sense.

2.4 Case: a simple sensory-motor system

To show the credibility of the approach of a rule-based reactive system, a simple simulation will be explained using this approach for a basic sensory-motor system. A simulated agent has two wheels to perform four different actions:

A1 STOP: no motor action
A2 FORWARD: both wheels turn with equal constant speed
A3 TURN_LEFT: both wheels move with opposite angular velocities
A4 TURN_RIGHT: same as TURN_LEFT but now in the other direction

The simulated agent has three range sensors to 'see' objects left, in front or right of it; three whiskers to 'feel' objects in these three directions and one tactile ring to feel a collision in any direction. These sensors can give world

information that is encoded (by the PCF of the functional model) in the five conditions below. We deliberately combined the three whiskers into one condition C5 in order to make the case simpler and the learning process more interesting.

C1 OBJECT_LEFT: object observed at left side
C2 OBJECT_FRONT: object observed in front
C3 OBJECT_RIGHT: object observed at right side
C4 COLLISION: the tactile ring is activated (position unknown)
C5 WHISKERS: one or more whiskers are activated (which one unknown)

These five conditions can be coded as bits in a binary number ranging from 0 to 31, where the least significant number represents the truth of C1 and so on. We call these numbers states. State 28 (11100) means that an object at the right side touches the whiskers and the tactile ring. State 8 (01000) means collision at the back side, because that is the only place without whiskers! All other states with COLLISION imply WHISKERS to be true too, so states 9 to 15 never occur. Neither will state 16 and 24, because the truth of WHISKERS implies the object to be observed somewhere.

The action selection mechanism has to apply its life-rules and learn new ones as well. The agent's initial state of mind is defined by two rules:

M1 "collision hurts so don't collide", that is NOT C4
M2 "moving is more fun than doing nothing", that is A2 OR A3 OR A4

Some discrimination is needed between these two rules; the agent must know what is more important. In general it is more meaningful to avoid life threatening situations or "negative feelings", and then seek for situations that cause "positive feelings". Hence, in this case rule M1 is more important. A mechanism is implemented that evaluates rules based on the definition of a state of mind. Each activated rule is given a reinforcement signal, that is determined by the state of mind. Because a mechanism is added that models delayed reinforcement, a part of the reinforcement signal is added to other rules that were active before as well. This way, every rule that is activated will receive a sum of several reinforcement signals. This sum is used to evaluate the corresponding rule. For this the term reward is used, and punishment in case of a negative reward. The evaluation mechanism needs a kind of forgetting mechanism. If the associations of certain life-rules become very strong, the system adapts only slowly to environmental changes.

Several simulations have shown that an agent is able to learn to wander in the environment with reducing the number of collisions. Starting with a large number of collisions, after some time the agent was able to move without collisions. As explained before, life-rules are compact and can be transferred easily to other agents. For simulation purposes, the observer could have the possibility to transfer the best life-rules from one agent to another. Then new life-rules will affect the behaviour of the receiving agent; the agents can learn from each other. Although this mechanism is of course too simple, test with this feature showed some interesting results.

3. Simulation results

The simulation was programmed in Occam, a language that was developed, together with the Transputer, to describe parallel communicating processes. As natural systems are inherently parallel, this parallelism can easily be recognised in our model. In future more levels of intelligence will be added and run in parallel. The lowest reactive level generating reactive behaviours will then be able to run in parallel with a higher level that continuously reasons with the life-rules offered by the lower level. Occam offers a convenient way to translate the system's structure directly to source code.

The results of this simulation is very promising as the agent was able to move in a simple static world consisting of posts and walls. This almost without colliding, but problems arise when the world is made more complex. Then the agent can get stuck at various places. As the action selection mechanism is fixed, it is not able to adapt to these new situations. If the agent is able to sense a situation of "getting stuck" it will try anything until the problem is solved and then it can deal with these situations. Figure 2 shows a screen dump of the simulation. The straight lines form the objects of the world. The circle nearby the right wall signifies the agent, the line inside shows the orientation or front of the agent. The movements of the agent are marked with small dots.

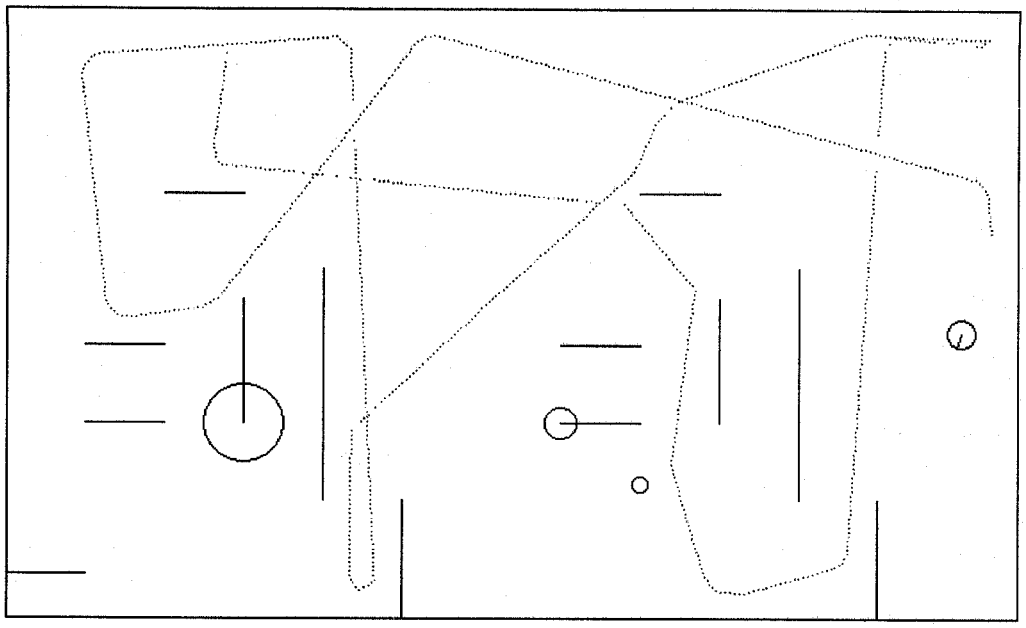

Figure 2: Screen dump of the simulation.

Figure 3 shows the mean positive reward for rules. It is obvious that some rules receive high rewards and others do not. The emptiness of states 9 to 16 was explained above. The other area in the set of possible rules (state 24 to 31, with both COLLISION and WHISKERS true) received hardly any positive reward. These states are entered several times by the agent but resulted most of the time in high negative rewards. The lonely peak at state 28 is explained below.

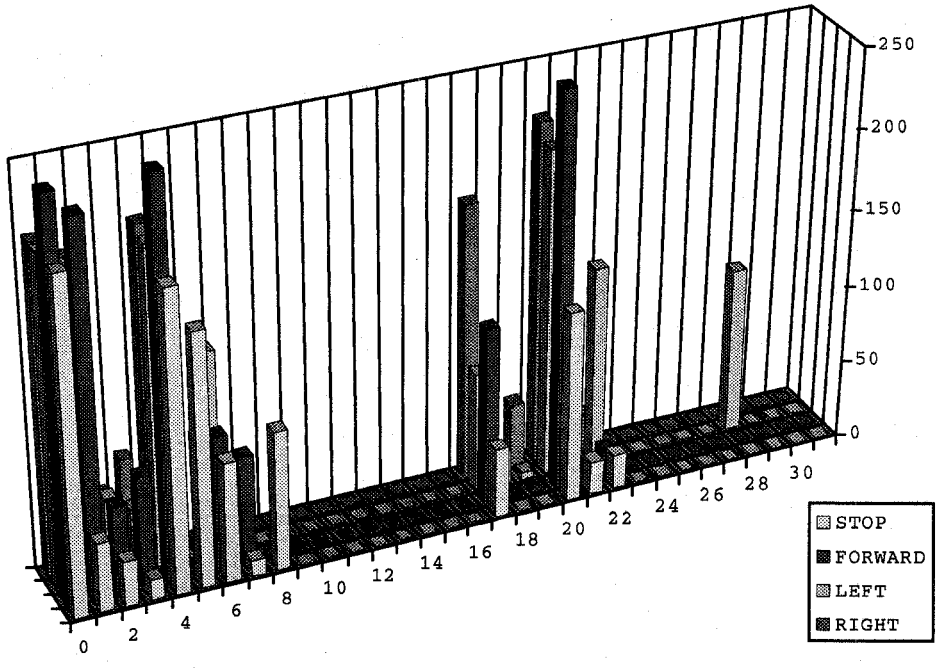

Figure 3: Chart of mean reward for each rule showing positive rewards only.

Now a closer look is taken at some specific life-rules. Eventually, 13 life-rules (with high enough positive rewards) were built up automatically. They all had some significant meaning. One exceptional life-rule is illustrated below. Figure 3 shows a peculiar peak for state 28 (11100) - action LEFT (A3). The corresponding life-rule states the following:

IF NOT OBJECT_LEFT AND NOT OBJECT_FRONT AND OBJECT_RIGHT AND COLLISION AND WHISKERS THEN TURN_LEFT

Looking at the rule, it seems that the agent is in a situation where it collides on the right side with an object, hence three conditions are active (OBJECT_RIGHT, COLLISION, WHISKERS). The agent does not see any object in the front and left direction. If the agent turns left, it likely will find a free path and can continue moving forward. From this life-rule we can conclude that the agent learns how to deal with a situation where it collides on its right side.

Let's investigate another peak in Figure 3. In state 17 (10001) action RIGHT (A4) gives the best result. The life-rule is:

IF OBJECT_LEFT AND NOT OBJECT_FRONT AND NOT OBJECT_RIGHT AND NOT COLLISION AND WHISKERS THEN TURN_RIGHT

Now the agent is near a wall (whiskers are active) that blocks the way in the left direction. The free path is found at the right side and hence the agent turns to right.

4. Conclusions

A study of existing behaviour systems led to the conclusion that a good system must satisfy the following demands. First of all, the behaviour system may not contain any human world knowledge or symbols. The knowledge should be acquired exclusively by the system itself.

Secondly, the system must use some kind of dynamic data structure where behavioural code as well as knowledge and information can be stored. This requires some kind of "general purpose memory". Moreover, if the agent wants to minimise on storing data but not on information, it must be able to generalise over data and be able to store generalised data. Classifier systems seem to give a first solution to this problem by using "don't care" symbols.

A first solution to translate "learning from experience" into behaviour can be obtained by learning what effect actions have on certain situations in the environment. By learning the effect of actions, called *associations*, it is possible to perform the proper action in each situation. The information learned can be stored as condition action-pairs. All reinforcement learning systems, including classifier systems, are based on a kind of condition-action pair. However, these systems don't go a step further when learning these pairs. As proven by the results of a simulation, such rules contain learned information and hence knowledge.

In this paper, a system of several layers of intelligence is proposed. The lowest level consists of a rule-based reactive system. A functional model is given that consists of three blocks: the perceptual condition filter (PCF), the action selection mechanism (ASM) and the action-to-actuator converter (AAC).

The perceptual condition filter offers a set of high functional perceptual conditions to the action selection mechanism. These condition data are filtered from the raw sensor data. The action-to-actuator converter offers a set of actions to the action selection mechanism and the selected actions are translated into the proper actuator commands. The action selection mechanism is thus offered a set of conditions and actions. All important learning and reasoning mechanisms are combined in this block and belong to the ASM. The PCF and the AAC are functional blocks where the designer can influence the possibilities of the resulting agent. The ASM comprises a kind of universal self-learning mechanism that is useful for a wide range of applications.

The proposed rule-based reactive level learns associations based on a so-called *state of mind*. The state of mind should resemble natural feelings of creatures like "happiness", "loneliness", "sadness", etc. Based on this state of mind a reinforcement signal is created that results in a reward for chosen actions.

The condition action pairs that emerge from the learning mechanism are called "life-rules" and contain some learned knowledge. These rules can be used by a higher level of intelligence, on top of the bottom level, to provide the source for reasoning and finding new rules. These rules may be exchanged with fellow agents on demand.

5. Future work

ADAM is in statu nascendi. ADAM will have a layered brain, as mentioned before. The lowest level of the brain is discussed in this paper. Some of the systems of the higher levels are tested, while on some other systems research is still going on. In short these systems will be described.

5.1 Space system

When an agent has a lot of sensors, the agent receives an enormous amount of data. Because the surrounding world has a certain structure, the received data must express these structures. Structures can be recognised as

redundancy in the data. Constraints describe these redundancy. So constraints express knowledge about the world. A program is developed that can determine in an efficient way different kinds of regularities in the surrounding, like e.g. IF (raven) THEN (black).

An other program is designed that looks at the incoming data for tuples that are not consistent with the set of possible functionalities. After all, it is possible that some day a non-black raven is found.

5.2 Time system

Time can be introduced if it is possible to recognise sequences of events. It was important for ancient men to see that every night is followed by a day, and that a summer is followed by autumn, winter and spring respectively. Transformations become evident by the passing of time in a search for the grand law of constancy. In order to introduce time as a factor in the system of an agent, the incoming data tuples are provided with a time stamp. This makes it possible for the system to detect possible dynamic constraints. These constraints describe the regularities in the sequences of incoming data tuples.

5.3 Reasoning system

Our goal is an autonomous system that is able to learn and can make deductions and inductions. So one of the layers is a reasoning system. When we restrict ourselves to functional dependencies, it is not difficult to write a program that is able to make deductions.

For reasoning under uncertainty a number of rules have been developed. And, of course, it is possible to use all kinds of rules that are found in literature. An easy way is to use a time window, and count the number of tuples in two projections.

5.4 Communication system

An important feature is the possibility to learn from one another. It is easy to transfer data, but it is not so easy to extract the right information out of the data. In communication we meet the symbol grounding problem [10]. We hypothesise that the world has a number of characteristics for two beings. So the constraints or rules of these two beings must show comparable patterns. Based on these patterns it is possible to make a translation from one data space to another.

Acknowledgement

This article is based on the M.Sc. thesis of L.F.J. Nijenhuis [16]

References

1. Aeken, F. van, and Sinnaeve, G. (1993) Applying Classical Conditioning to a Real Robot. In: Conferentie over leersystemen in de "lage landen" gehouden in het congrespaleis - Benelearn-93, p. 1-14.
2. Arrayo, F., Gonzalo, A. and Moreno, L.E. (1991) Neural network design for mobile robot control following a contour. In: Artificial Neural Networks, CWANN '91
3. Asteroth, A., Fischer, M.S., M"ller, K. and Schnepf, U. (1992) Tracking and Grasping of Moving Objects - A Behaviour-Based Approach -. In: Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 5th International Conference, IEA/AIE-92, Springer-Verlag, Berlin, Germany.
4. Barto, A.G., Sutton, R.S., Anderson, C.W. (1983) Neuronlike adaptive elements that can solve difficult learning control problems, IEEE Trans. on System, Man, and Cybernetics, SMC-13:834-846.
5. Barto, A.G. and Sutton, R.S. (1991) Landmark learning: An Illustration of associative search, Biological Cybernetics, vol. 42, p. 1-8.
6. Brooks, R.A. (1991) The role of learning in autonomous robots. In: Proceedings of the Fourth Annual Workshop on Computational Learning Theory, Morgan Kaufmann, San Mateo, CA, USA, p. 5-10.
7. Connel, J.H. (1990) Minimalist Mobile Robotics, A Colony-style Architecture for an Artificial Creature, Academic Press, London.
8. Gasos, J., Carcia-Alegre, M.C., and Garcia Rosa, R. (1990) Fuzzy local navigation of a simulated real robot in unknown environments. In: Proc. of the IEEE Int. Workshop on Intelligent Motion Control, Istanbul, Turkey, p. 445-449.
9. Harashima, F., Hashimoto H. and Kubota, T. (1990) Sensor Based Robor Control Systems. In: Proc. of the IEEE Int. Workshop on Intelligent Motion Control, Istanbul, Turkey, p. PL1-PL10.
10. Harnad, D. (1991) The Symbol Grounding Problem. In: Forrest, S. (ed.) (1991) Emergent Computation, Elsevier Science/MIT Press.
11. Kaelbling, L.P. (1992) An Adaptive Mobile Robot. In: Varela, F.J., Bourgine, P. (eds.) (1992) Toward a Practice of Autonomous Systems, Proc. of the First European Conf. on Artificial Life, MIT Press/Bradford Books, Cambridge Ma., p. 41-47.
12. Koza, J.R. (1992) Evolution of Subsumption Using Genetic Programming. In: Varela, F.J., Bourgine, P. (eds.) (1992) Toward a Practice of Autonomous Systems, Proc. of the First European Conf. on Artificial Life, MIT Press/Bradford Books, Cambridge Ma., p. 110-119.
13. Maeda, Y., Tanabe, M. and Takagi, T. (1993) Behavior-decision fuzzy algorithm for autonomous mobile robots, Information Sciences, vol. 71, p. 145-168.
14. Maes, P. (1994) Modeling Adaptive Autonomous Agents, Artificial Life, Vol. 1, Number ½, p. 135-162.
15. McFarland, D., Bösser, T. (1993) Intelligent Behavior in Animals and Robots, Bradford Press.
16. Nijenhuis, L.F.J., 'Modelling and simulation of a self-learning autonomau mobile robot - an artificial life approach, M.Sc. thesis, University of Twente, The Netherlands, September 1994.
17. Pfeifer, R., Verschure, P. (1992) Distributed Adaptive Control: A Paradigm for Designing Autonomous Agents.

In: Varela, F.J., Bourgine, P. (eds.) (1992) Toward a Practice of Autonomous Systems, Proc. of the First European Conf. on Artificial Life, MIT Press/Bradford Books, Cambridge Ma., p. 21-30.
18. Rooijen, N.J. van, Waerdt, J.W. van de, (1993) Exploration of an unknown environment by an autonomous mobile robot for navigation purposes, M.Sc.Thesis, University of Twente, INF 730, september 1993.
19. Steels, L. (1994) Artificial life roots of artificial intelligence, Artificial Life Journal, Vol. 1, Number ½, MIT Press, Cambridge.
20. Waay, B.v.d. (1994) Controlling agents with urges, M.Sc.Thesis, University of Twente, SPA-94-11, july 1994.
21. Zhang, Y. (1989) Transputer-Based Behavioral Module for Multi-Sensory Robot Control, Parallel Processing and Artificial Intelligence, Wiley, West Sussex, England, p. 217-232.