Small Journal Name, 9, 1–22 (1992)

Unsupervised Learning of Eye-Hand-Coordination

ANDREAS BIRK cyrano@arti.vub.ac.be
Vrije Universiteit Brussel, Artificial Intelligence Laboratory, 1050 Brussels, Belgium

WOLFGANG J. PAUL wjp@cs.uni-sb.de
Universität des Saarlandes, Computer Science Department, 66123 Saarbrücken, Germany

Abstract. Unsupervised learning of eye-hand-coordination is an interesting problem for two very different reasons. First, it is an important step in the human cognitive development. Second, when applied to a real-world set-up as we do here, it has an application potential. Demonstrating its potential on this concrete task, we present a novel approach to unsupervised learning, the so-called stimulus-response-learning or short SRL. It features an on-line evolution of simple reactive rules stored in a dynamic directed graph, such that both reactive behavior and a word-model are learned in parallel. The graph is somewhere similar to belief-networks as it represents potential consecutive activation of rules, hence allowing a simple inference of future states of the environment in dependence of possible actions of the system. But the unsupervised learning of both rules and the graph are not based on a bayesian or any related learning technique, but on a novel type of on-line evolution. Unlike common evolutionary techniques, the fitness function in this algorithm is independent of the task as it is based on general statistical measures.

Keywords: world-modeling, behavioral control, eye-hand-coordination, animat, on-line evolution

1. Introduction

During the last decade, robots have been strongly promoted as experimental platforms for AI research. Especially the bottom-up approach to investigate simple animal-like robots or animats [26] and to stepwise increase their complexity was and still is very popular within this line of work. Reactive controllers as fast, direct couplings between sensors and motors have dominated this field, mainly motivated by Brooks' well-known critique on "classical" AI [3, 4]. When it comes to research on learning robots (see e.g. [1] for an overview), the exploitation of task-dependent reinforcement or even explicit information from a teacher is the most common approach.

We likewise strongly belief that a constructive understanding of intelligence has to be grounded in a physical body interacting with an environment, or to (ab)use an ancient Roman slogan: "mens sana in copore sano", stating that a sane mind needs a sane body. But though following the "artificial life route to artificial intelligence" [22], we also belief that world-models and inference can not be neglected in the long run when trying to understand higher-level intelligence up to cognition. Furthermore, we strongly belief that intelligence is a continuous process, which is mainly based on an unsupervised acquisition of knowledge.

This acquisition or learning process is unsupervised in a strong sense, i.e., it is a general extraction of structure from data without neither any reinforcement or

any explicit information from a teacher nor an a priory bias on how this knowledge will be used. This even includes the absence of innate reinforcement mechanisms like basic drives in the form of e.g. pain, hunger, and so on. Of course, when using the knowledge learned, basic drives and other forms of bias do play a role and they influence partially which knowledge is kept in the long run. But primarily, this process is a life-long construction and up-dating of a so-to-say general theory of the world in the spirit of [20], without necessarily regarding if or how this knowledge is used in the sense that there is no a priori focus of inference on the knowledge.

This view is partially supported by psychological experiments indicating that even animals with simple non-human intelligence are capable of an unsupervised construction of world-models. Seward reports for example in [24] that rats learn the structure of a maze without reinforcement.

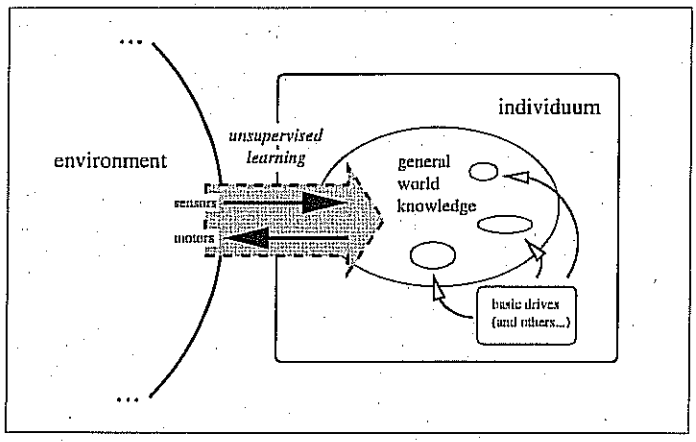

Figure 1. Our view on "intelligence" as a continuous process boot-strapped by an unsupervised acquisition of knowledge as a general and dynamic "theory of the world". This acquisition or learning process is unsupervised in a strong sense as it is independent from any external or innate reinforcement or any explicit information from a teacher. It is even not related to the fact if or how this knowledge is used in the sense that there is no a priori focus of inference. Basic drives and other forms of bias depending on the task/environment come only later on into play, using the acquired knowledge and partially influencing which knowledge is kept in the long run.

The rest of this article is structured as follows. In section 2, the basic concepts of SRL are defined and motivated. Section 3 introduces the set-up for learning eye-hand-coordination, which is then used for concrete examples to further illustrate the principles of SRL. Related research, especially work by Drescher, is discussed in section 4. Results with SRL are presented, which significantly out-perform this previous work. In section 5, further results are presented from simulated and real-world set-ups. Varying problem-sizes are used for a rough analysis of SRL's computational behavior. Section 6 concludes the article and sketches potential future work.

2. Stimulus Response Learning

2.1. Overview

Stimulus response learning is an attempt towards bridging the gap between reactive control and world-modeling. It permits animats to explore and internally model an environment by means of unsupervised learning. The basic elements of this world-model are reactive in the sense that they establish a close link between sensor-data and motor-activations. But they do not form a strict *must*-do-control. Instead, they supply the system with a *can*-do-knowledge as they allow a simple form of inference.

The central data structure of the world-model used in SRL is a dynamic directed graph, which is called the *m-net* as motivated later on. Its nodes are simple rules for behavioral-control, the so-called *stimulus response rules (SRRs)*, which are composed of predicates and actions. In its simplest form such a rule is a classifier [14, 13], i.e., it has the form c/a, representing that action a can be performed if condition c holds. Performing a when c holds is called *execution* of the rule. In addition to its "control-character", a rule is seen as a prediction, namely that the world changes (in respect to sensor-data) due to execution of the rule. In a more elaborated version, SRRs resemble schemas [8], having a result r describing the sensor-data after execution.

A directed edge between two rules represents a possible consecutive execution of the rules. Looking at a path in the m-net therefore allows to anticipate the future consequences of actions. An m-net is learned by a novel type of evolutionary algorithm where the fitness of rules and edges is measured by two simple, purely statistical quality measures, namely *reliability* and *applicability*. Reliability is the relative number of times the prediction associated with a rule or edge holds. Applicability is the number of times an edge or rule is actually used. Note that no reinforcement is used in this framework. The system maintains solely relations between sensor-data and motor-activations in the m-net which have sufficient statistical support from iterated "experiences".

By building up a directed graph, the system constructs a spatial — but in general not Euclidean — model of the world. In the graph we keep track of the SRR executed last. This SRR is called the *standpoint*. It models the system's current position in the world. It is introduced for two reasons. First, simple inference of future environment states, which can be used for planning, reduces to a search of paths from the standpoint to an SRR with a desired property. Second, continuous processing of the whole knowledge learned so far is avoided.

The underlying evolutionary algorithm uses in each learning-step rules and edges that are connected to the standpoint, i.e., the standpoints' neighborhood in the graph theoretic sense. In a large m-net this greatly reduces the complexity of learning steps and captures the following intuition: if we want to explain a new phenomenon at some place in the world (modeled by the standpoint in the m-net), then we expect existing knowledge about that part of the world (the neighborhood

Small Journal Name, 9, 1–22 (1992)

Unsupervised Learning of Eye-Hand-Coordination

ANDREAS BIRK cyrano@arti.vub.ac.be
Vrije Universiteit Brussel, Artificial Intelligence Laboratory, 1050 Brussels, Belgium

WOLFGANG J. PAUL wjp@cs.uni-sb.de
Universität des Saarlandes, Computer Science Department, 66123 Saarbrücken, Germany

Abstract. Unsupervised learning of eye-hand-coordination is an interesting problem for two very different reasons. First, it is an important step in the human cognitive development. Second, when applied to a real-world set-up as we do here, it has an application potential. Demonstrating its potential on this concrete task, we present a novel approach to unsupervised learning, the so-called stimulus-response-learning or short SRL. It features an on-line evolution of simple reactive rules stored in a dynamic directed graph, such that both reactive behavior and a word-model are learned in parallel. The graph is somewhere similar to belief-networks as it represents potential consecutive activation of rules, hence allowing a simple inference of future states of the environment in dependence of possible actions of the system. But the unsupervised learning of both rules and the graph are not based on a bayesian or any related learning technique, but on a novel type of on-line evolution. Unlike common evolutionary techniques, the fitness function in this algorithm is independent of the task as it is based on general statistical measures.

Keywords: world-modeling, behavioral control, eye-hand-coordination, animat, on-line evolution

1. Introduction

During the last decade, robots have been strongly promoted as experimental platforms for AI research. Especially the bottom-up approach to investigate simple animal-like robots or animats [26] and to stepwise increase their complexity was and still is very popular within this line of work. Reactive controllers as fast, direct couplings between sensors and motors have dominated this field, mainly motivated by Brooks' well-known critique on "classical" AI [3, 4]. When it comes to research on learning robots (see e.g. [1] for an overview), the exploitation of task-dependent reinforcement or even explicit information from a teacher is the most common approach.

We likewise strongly belief that a constructive understanding of intelligence has to be grounded in a physical body interacting with an environment, or to (ab)use an ancient Roman slogan: "mens sana in copore sano", stating that a sane mind needs a sane body. But though following the "artificial life route to artificial intelligence" [22], we also belief that world-models and inference can not be neglected in the long run when trying to understand higher-level intelligence up to cognition. Furthermore, we strongly belief that intelligence is a continuous process, which is mainly based on an unsupervised acquisition of knowledge.

This acquisition or learning process is unsupervised in a strong sense, i.e., it is a general extraction of structure from data without neither any reinforcement or

any explicit information from a teacher nor an a priory bias on how this knowledge will be used. This even includes the absence of innate reinforcement mechanisms like basic drives in the form of e.g. pain, hunger, and so on. Of course, when using the knowledge learned, basic drives and other forms of bias do play a role and they influence partially which knowledge is kept in the long run. But primarily, this process is a life-long construction and up-dating of a so-to-say general theory of the world in the spirit of [20], without necessarily regarding if or how this knowledge is used in the sense that there is no a priori focus of inference on the knowledge.

This view is partially supported by psychological experiments indicating that even animals with simple non-human intelligence are capable of an unsupervised construction of world-models. Seward reports for example in [24] that rats learn the structure of a maze without reinforcement.

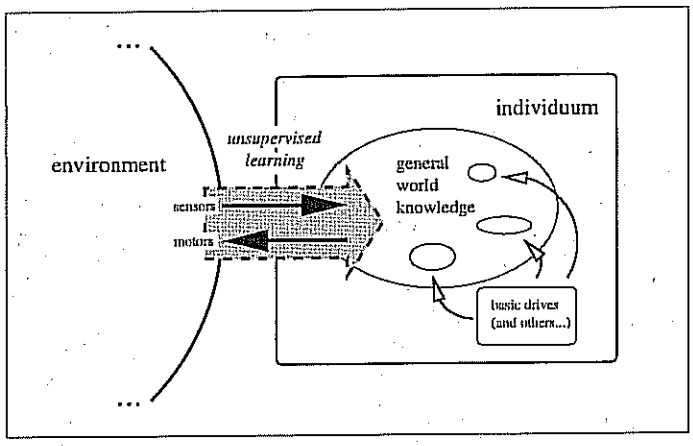

Figure 1. Our view on "intelligence" as a continuous process boot-strapped by an unsupervised acquisition of knowledge as a general and dynamic "theory of the world". This acquisition or learning process is unsupervised in a strong sense as it is independent from any external or innate reinforcement or any explicit information from a teacher. It is even not related to the fact if or how this knowledge is used in the sense that there is no a priori focus of inference. Basic drives and other forms of bias depending on the task/environment come only later on into play, using the acquired knowledge and partially influencing which knowledge is kept in the long run.

The rest of this article is structured as follows. In section 2, the basic concepts of SRL are defined and motivated. Section 3 introduces the set-up for learning eye-hand-coordination, which is then used for concrete examples to further illustrate the principles of SRL. Related research, especially work by Drescher, is discussed in section 4. Results with SRL are presented, which significantly out-perform this previous work. In section 5, further results are presented from simulated and real-world set-ups. Varying problem-sizes are used for a rough analysis of SRL's computational behavior. Section 6 concludes the article and sketches potential future work.

2. Stimulus Response Learning

2.1. Overview

Stimulus response learning is an attempt towards bridging the gap between reactive control and world-modeling. It permits animats to explore and internally model an environment by means of unsupervised learning. The basic elements of this world-model are reactive in the sense that they establish a close link between sensor-data and motor-activations. But they do not form a strict *must*-do-control. Instead, they supply the system with a *can*-do-knowledge as they allow a simple form of inference.

The central data structure of the world-model used in SRL is a dynamic directed graph, which is called the *m-net* as motivated later on. Its nodes are simple rules for behavioral-control, the so-called *stimulus response rules (SRRs)*, which are composed of predicates and actions. In its simplest form such a rule is a classifier [14, 13], i.e., it has the form c/a, representing that action a can be performed if condition c holds. Performing a when c holds is called *execution* of the rule. In addition to its "control-character", a rule is seen as a prediction, namely that the world changes (in respect to sensor-data) due to execution of the rule. In a more elaborated version, SRRs resemble schemas [8], having a result r describing the sensor-data after execution.

A directed edge between two rules represents a possible consecutive execution of the rules. Looking at a path in the m-net therefore allows to anticipate the future consequences of actions. An m-net is learned by a novel type of evolutionary algorithm where the fitness of rules and edges is measured by two simple, purely statistical quality measures, namely *reliability* and *applicability*. Reliability is the relative number of times the prediction associated with a rule or edge holds. Applicability is the number of times an edge or rule is actually used. Note that no reinforcement is used in this framework. The system maintains solely relations between sensor-data and motor-activations in the m-net which have sufficient statistical support from iterated "experiences".

By building up a directed graph, the system constructs a spatial — but in general not Euclidean — model of the world. In the graph we keep track of the SRR executed last. This SRR is called the *standpoint*. It models the system's current position in the world. It is introduced for two reasons. First, simple inference of future environment states, which can be used for planning, reduces to a search of paths from the standpoint to an SRR with a desired property. Second, continuous processing of the whole knowledge learned so far is avoided.

The underlying evolutionary algorithm uses in each learning-step rules and edges that are connected to the standpoint, i.e., the standpoints' neighborhood in the graph theoretic sense. In a large m-net this greatly reduces the complexity of learning steps and captures the following intuition: if we want to explain a new phenomenon at some place in the world (modeled by the standpoint in the m-net), then we expect existing knowledge about that part of the world (the neighborhood

of the standpoint) to be more useful than existing knowledge about remote parts of the world.

2.2. The M-Net as Controller and World-Model

As mentioned before, we are interested in a constructive understanding of intelligence by investigating animal-like robots or animats, starting with simple systems and step-wise incrementing their complexity. Within AI, there has been and to some extent still is a big debate about re-active versus model-based approaches. Instead of engaging in yet another discussion, we simply denote here any kind of software which helps the animat to get along in its environment as an animat-"mind", let it be reactive or model-based- or both.

Within SRL, the animat-"mind" is based on a dynamic directed graph, hence called the "mind"-net or short the m-net. The nodes are as mentioned before the so-called *stimulus response rules (SRR)*. Concretely, the concept of an SRR captures following three types of rule: a slightly extended version of the condition action rule *CAR* of classifier systems [13], the *schema* as used by Gary L. Drescher [8], and the *TOTE*. The TOTE was introduced as psychological concept by George Miller, Eugene Galanter and Karl Pribam in [10] and has, as far as we know, not yet been used in AI.

The basic elements of SRRs are *tests* and *actions*. A test t is a predicate on sensor-states represented by so-called *sensor-channels*. Actions activates effectors via *motor-channels*. The set of tests is denoted with $\mathcal{T}$ and the set of actions is denoted with $\mathcal{A}$.

- A *CAR* is a pair $c/a \in \mathcal{T} \times \mathcal{A}$. It is the most simple SRR consisting of a test c, the so-called *condition*, and an *action* a. If the condition holds, the action can be executed.

- A *schema* is a triple $c/a/r \in \mathcal{T} \times \mathcal{A} \times \mathcal{T}$. It is a CAR extended by a test r, the so-called *result*. The result is supposed to represent the state of the environment after execution of action a.

- A *TOTE* is a quadruple $c/a/f/r \in \mathcal{T} \times \mathcal{A} \times \mathcal{T} \times \mathcal{T}$. The letters TOTE are an abbreviation for test_1, operate, test_2, exit. A TOTE is an extension of a schema, where test_1 corresponds to the condition, the operate corresponds to the action and the exit corresponds to the result. The action of a TOTE is repeatedly executed until test_2, the so-called *feedback* holds.

For an SRR s we use $c(s), a(s), f(s)$ and $r(s)$ to denote condition, action, feedback and result of s.

Assume t_{true} is a test which is always fulfilled. The CAR c/a corresponds to the schema $c/a/t_{true}$ and the schema $c/a/r$ corresponds to the TOTE $c/a/t_{true}/r$ with the above semantics. Therefore, the set S of SRRs is the set of TOTEs containing

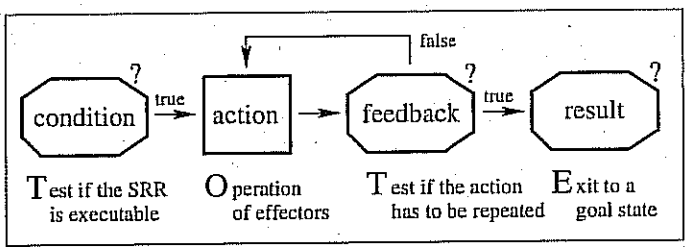

Figure 2. The TOTE as the most general SRR. When the condition holds, the rule can be activated, i.e., the action is repeatedly executed until the feedback becomes true. The result of the rule represents the change of the environment after activation of the rule. Known approaches to rules, namely schemas and CARs of classifier-systems are a subset of this type of rule.

implicitly schemas and CARs. Figure 2 shows the working-principle of a TOTE as most general form of SRR.

TOTEs are like do-until loops; their execution only terminates when the feedback turns true. In order to prevent the system from getting stuck, the execution of a TOTE is aborted if the action is repeated more than a fixed rep_{max} times.

A *m-net* is a triple (V, E, sp) where (V, E) is a directed graph, V is a set of SRRs and $sp \in V$ is a node called the *standpoint*. An edge $(s_1, s_2) \in E$ between two SRRs represents a possible consecutive execution of s_1 and s_2. It is tempting to include in E all edges (s_1, s_2) such that the result $r(s_1)$ logically implies the condition $c(s_2)$. But for practical purposes this does not work: if the set $\mathcal{T}$ of tests is sufficiently powerful, then the question whether $r(s_1) \Rightarrow c(s_2)$ quickly becomes NP-hard or even undecidable. Alternatively, one can severely restrict the set $\mathcal{T}$ of tests. This is the route taken by Drescher ([8], see also section 4 on related work). There, tests are characterized by monomials. This makes the decision $r(s_1) \Rightarrow c(s_2)$ trivial, but tests cannot be negated. Most of the effort in [8] is spent struggling with the consequences of this.

We chose an extremely simple and brute force way out of this dilemma which is purely based on statistics. One way would be to keep track of how often $c(s_2)$ holds in situations where $r(s_1)$ holds, but this would require to update statistics about all SRRs in the graph all the time. A faster way is to count how often $c(s_2)$ holds after execution of s_1. Thus, implication is based on simple statistics and may of course be unreliable. Inferences on the m-net are far less powerful than in logic-based or common probabilistic approaches, but they are computationally inexpensive and simple. They can be learned and executed in a very fast manner, and they allow anticipation and planning capabilities for the animat, which are far beyond simple reactive behavior.

2.3. The Statistical Measures

The statistical measures for the quality of rules and edges are based on the fact that both an SRR s and an edge (s_1, s_2) represent assumptions. It is easy to test these assumptions by executing the SRR s or by trying to execute s_2 if the standpoint is s_1. Thus, if the assumption holds very often, then we will consider the SRR or the edge to be very *reliable*. Furthermore, if the assumption is often used, then we will consider the SRR or the edge to be very *applicable*.

Concretely, the execution of an SRR s is called *successful* iff after the execution:

1. the condition $c(s)$ does not hold, i.e., the action $a(s)$ has changed the perceived state of the environment, and
2. the feedback $f(s)$ became true before the action a was repeated more than rep_{max} times, and
3. the result $r(s)$ holds.

Note that for a schema condition two and for a CAR conditions two and three are always fulfilled. Also note that in the common usage of CARs in classifier-systems, it is usually not explicitly required that the action changes the state of the world as perceived by the condition.

SRL proceeds in iterated *time steps* t. We denote by $mnet_t = (V_t, E_t, sp_t)$ the m-net after time step t. An edge $e = (s_1, s_2)$ is *traversed* in time-step t iff s_2 is executed in time-step t. This implies that the standpoint changes from s_1 to s_2 during time-step t. The (traversal of the) edge is *successful* iff the execution of s_2 is successful.

The *reliability* $rel()$ of an SRR s, respectively edge e is defined as

$$rel(s \text{ (or } e)) = \frac{\text{number of successful executions of } s \text{ (or } e)}{\text{number of executions of } s \text{ (or } e)}$$

Let the *lifetime* of s (or e) denote the number of time steps of the system since creation of s, respectively e. The *applicability* $app()$ of an SRR s, respectively edge e is defined as

$$app(s \text{ (or } e)) = \frac{\text{number of executions of } s \text{ (or } e)}{\text{lifetime of } s \text{ (or } e)}$$

The *score* $sco()$ of a SRR s, respectively edge e is defined as the product of reliability and applicability:

$$sco(s \text{ (or } e)) = \frac{\text{number of successful executions of } s \text{ (or } e)}{\text{lifetime of } s \text{ (or } e)}$$

Let $\#V_t$ and $\#E_t$ denote the number of SRRs respectively edges in the present m-net. An SRR s, respectively edge e is considered to be a bad representation and

it is therefore removed from the m-net iff its score falls below a threshold t_{remove} with:

$$t_{remove} = 0.3 \cdot \frac{1}{\#V_t(\text{ or } \#E_t)}$$

The threshold t_{remove} can be motivated as follows. A useful SRR or edge should have a reliability close to One. Furthermore, exactly one SRR and edge is executed in one time-step by the system. So a "normal" SRR or edge is executed on average every $\#V_t$, respectively $\#E_t$ time-steps. The empirically found constant 0.3 is used to weaken the condition for removing an SRR or edge.

2.4. Learning an M-Net

The system starts at time $t = 0$ with an empty graph (V_0, E_0) and an undefined standpoint. In every following time step, the system randomly chooses between

- a training step using the current m-net
- creation of a new edge
- creation of a new SRR

Let $N_t = \{s \in V_t | (sp_t, s) \in E_t\}$ denote the set of SRRs reachable in $mnet_t$ from the standpoint by a single edge, i.e., the immediate neighborhood of the standpoint in the graph theoretic sense. A training-step on the current $mnet_t$ is done by selecting an SRR $s \in N_t$ with fulfilled condition $c(s)$. If such an SRR s exists, it is executed and the standpoint is set on s. Thus training is one way to update the scores of SRRs and edges.

The creation of a new edge is done as follows. An SRR $s \in V_t \setminus N_t$ is searched, such that its condition $c(s)$ holds. If the search is successful, the edge (sp_t, s) from the standpoint to s is included in E_{t+1}, the SRR s is executed and the standpoint is set on s.

The creation of a new SRR is done via a novel type of on-line evolution. In the four standard classes of evolutionary computation, namely genetic algorithms [13, 11], genetic programming [16, 17], evolutionary strategies [21, 23] and evolutionary programming [9], the fitness-function as a crucial element of the concrete algorithms is task-dependent and has to be carefully designed. In SRL in contrast, the fitness of elements in the evolutionary process is defined via the general statistical measures for rules and edges.

Concretely, the basic elements for a new rule, i.e., tests and action, are created by an evolutionary process, which operates on basic elements of rules in the current neighborhood of the standpoint. The fitness of the basic elements is simply based on the scores of their rules. So, the fitness-function is general in the sense that it is defined in respect to the task of world-modeling in general and that there is no adaption needed to fit the animat's concrete environment.

2.5. Creating Tests and Actions

Assume the animat has n_s sensor- and n_m motor-channels, each of which can take at time step t a single specific value from a finite set M_s of sensor- and a finite set M_m of motor-values. Tests and actions are defined via vectors, which can contain sensor- and respectively motor-values as well as the wild-card symbol '$*$'.

A vector $v_m[1 : n_m]$ with n_m entries from $M_m \cup \{*\}$ represents an action as follows:

- If the entry at place i is a legal motor value, i.e., $v_m[i] \in M_m$ for $i \in \{1, ..., n_m\}$, then this value $v_m[i]$ is sent to the according motor channel,
- otherwise, i.e., if the entry $v_m[i]$ is a wildcard '$*$', no value is sent to the motor-channel i.

A vector $v_s[1 : n_s]$ with n_s entries from $M_s \cup \{*\}$ represents a test as follows. The predicate defined by v_s is true if and only if for all places $i \in \{1, ..., n_m\}$

- the entry $v_s[i]$ is a wild-card '$*$', or
- $v_s[i]$ is equal to the current value of sensor-channel i.

Tests and action for a new SRR are created by one iteration of an on-line evolutionary algorithm. The same algorithm is consecutively used to separately construct the different basic elements of SRRs, i.e., conditions, actions, feedbacks, and results. In doing so, their fitness is, as mentioned before, defined via the score of their SRR. The algorithm, or more precisely each of the up to four invocations of the algorithm uses basic elements from N_t as population $\mathcal{P}$.

The evolutionary algorithm uses following operators to create new tests:

Adaption produces a vector v which is a snapshot of the current environment, i.e., $v[i] = ch[i]$ for all sensor-channels $ch[i]$.

Mutation produces a new vector v from an existing one v' by replacing an entry in v' by a wildcard at a random place.

Crossover produces a new vector v from two existing vectors v' and v'' by copying the head of v' and the tail of v'' with respect to a random place j, i.e., $v[i] = v'[i]$ for all $i \leq j$ and $v[i] = v''[i]$ for all $i > j$.

The operators for the creation of new actions are exactly the same, except the implementation of adaption. There, a vector v is created with randomly selected motor-values. In addition, it is tested if v changes the currently perceived state of the environment when v is sent to the motor-channels. If this is not the case, an other vector is created in the same manner.

Every time the system tries to create a new SRR s, the basic elements of s, i.e., the tests and action, are constructed with the above operators. In doing so, the

probability to select a basic element from the population $\mathcal{P}$ as input of an operator is proportional to the score of SRR from which the basic element is taken.

The likelihood with which an operator is used also depends on the scores of SRRs. Let w_{scores} denote the sum of the scores of the SRRs in N_t, the neighborhood of the standpoint. Note that N_t can be empty, leading to an empty population $\mathcal{P}$. Then, only adaption can be used as mutation and crossover need a non empty population. In this case, the corresponding probability $prob_{adap}$ of using adaption must be One. The probabilities $prob_{mut}$ of using mutation and $prob_{cross}$ of using crossover are defined such that otherwise, i.e., when $\mathcal{P} \neq \emptyset$, their probabilities increase the more and the better SRRs are in N_t. Concretely, the probabilities of using an operator are defined as follows:

$$prob_{adap} = \frac{w_{scores}}{w_{scores} + w_{adap}}$$

$$prob_{mut} = (1 - \frac{w_{scores}}{w_{scores} + w_{adap}}) \cdot w_{mut}$$

$$prob_{cross} = (1 - \frac{w_{scores}}{w_{scores} + w_{adap}}) \cdot w_{cross}$$

The constant w_{adap} weights the probability of using adaption against usage of mutation or crossover. The constants w_{mut} and w_{cross} are used to weight the probability of using mutation against crossover; they sum up to One. In all experiments reported in this article, we used following settings: $w_{adap} = 0.5$; $w_{mut} = 0.2$; $w_{cross} = 1 - w_{mut} = 0.8$.

Though some of these concepts resemble the common forms of evolutionary algorithms, especially genetic algorithms (GA), there are some substantial differences:

- The fitness-function is independent of the task in the sense that it is not adapted to the concrete animat or its environment, instead it is based on general statistical scores.
- Initial populations in GA are purely random. Here, adaption boot-straps the evolution with non-random, but usually much too specific tests as representation of states of the environment.
- Mutation does not make a random change on tests, but it increases their generality by ignoring some information about the environment.
- Mutation and crossover do not destroy their input when producing a new test or action.
- The cardinality of the population is not fixed.

2.6. Inference on an M-Net

With a graph as world model, a simple form of inference on the m-net is done via a path-search. The standpoint *sp* marks the system's current position in the world. It is always set to the SRR executed last. Given a desirable state g of the environment, the system can search the m-net for an SRR s with a result $r(s)$ which is fulfilled on g. To achieve the goal g, the system can execute the SRRs along a shortest path from the standpoint to s. Of course, it is also possible to search paths from any SRR with a desired property, e.g. an SRR with a certain condition c representing an environment state, to any other SRR with a desired property.

Note that when this inference is used for planning, i.e., the SRRs along the path are actually executed to achieve a goal, the scores of these SRRs get updated. Therefore, the provision of goals and planning influences the further development of the m-net. At this point, basic drives and other forms of task-dependent bias come into play, as they are responsible for the "desire" to achieve a certain goal. So, when the inference is followed by an execution of SRRs to achieve goals, the scores of SRRs on paths that are used often are updated often. As a consequence, SRL refines the m-net in these regions, it so-to-say focuses on getting knowledge that helps to achieve the goals. At the same time, the unsupervised learning still continues. So, despite the fact that for certain acquired knowledge a "purpose" is discovered, the system keeps on building a "general theory of the world" regardless of its usage.

3. Eye-Hand-Coordination as Example

3.1. The Basic Set-Up

In this section, the basic concepts of SRL are described by examples based on unsupervised learning of eye-hand-coordination. The concrete results from experiments on this task are presented later on in section 5.

Eye-hand-coordination is not a standard subject in the context of research on animats, as normally mobile robots are preferred. We chose this task for two reasons. First, it is an interesting problem as it forms an important early stage in the human cognitive development as reported by the Swiss psychologist Piaget [19]. Second, a real-world set-up with a robot-arm and vision forms a complex, but still somewhat controllable experimental framework. Infrastructural constraints are less severe for this set-up, as for example space-requirements are rather small, energy-supply is not a critical issue, and so on. Nevertheless, the set-up includes all the basic problems of real-world perception and motor-control.

Concretely, the experimental set-up consists of a robot-arm and a camera as illustrated in figure 3. Images from the camera are processed into a $n \times n$ grid of averaged pixel values. These values are fed into a linear vector $ch[1 : n^2]$ of $n_s = n^2$ sensor-channels. In doing so, there is no particular order on the sensor

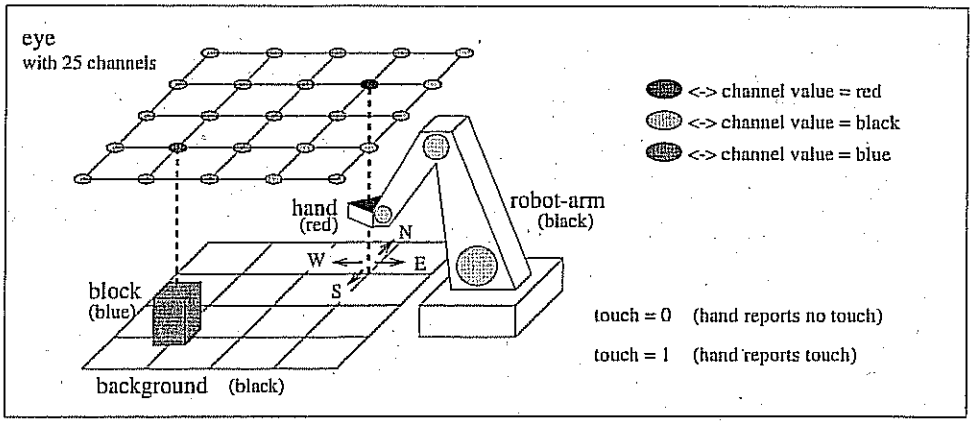

Figure 3. The basic set-up with a robot-arm and 25 vision-channels. The arm can move in the plane and grip colored building-blocks. Starting from scratch, SRL evolves a network of rules relating sensor values to possible actions and their consequences. Unlike common evolutionary algorithms, the fitness function in this process is independent of the task as it is based on general statistical measures.

data, i.e., no information about the grid-structure is part of the representation of the sensor-data.

The camera is fixed approximately perpendicular to a black background without any calibration or other positioning constraints. A blue building-block is sometimes put on the background. A red robot-gripper, the so-called hand can move in the four directions of the plane. Furthermore, the hand can grip the building block if it is directly under the hand and carry it around.

Typical values for a sensor channel $ch[i](1 < i < n^2)$ are therefore `red`, `blue` or `black`. In our real-world experiments, each of the 8 colors used by the vision can occur caused by reflexes, shadows, and so on. In addition, a sensor-channel $ch[n^2 + 1]$ is devoted to sensing touch. It reports the value 1 when the building-block is underneath the hand, otherwise it has the value 0.

The actions are based on a single motor-channel, which can take values to move the hand in the four directions and to grip and ungrip. So, the set of actions is $\mathcal{A} = \{\texttt{north}, \texttt{south}, \texttt{west}, \texttt{east}, \texttt{grip}, \texttt{ungrip}\}$. We also did experiments with two motor-channels, one for movements and one for gripping. But as these two channels can not be activated in parallel in this set-up, the results are identical to experiments with a single channel. Note that this is due to the semantics of the adaption operator for the creation of actions, which yields in the case of the two *independent* channels to the action-vectors $(\texttt{north}, *), (\texttt{south}, *), (\texttt{west}, *), (\texttt{east}, *), (*, \texttt{grip}), (*, \texttt{ungrip})$.

3.2. The Eye-Hand M-Net

The m-net representing hand-movements with a 4×4 grid of vision-channels is shown in figure 4. The following conventions are used:

A black frame is drawn around SRRs. Several edges are drawn together by one symbol (figure 5): n edges from a source-SRR s_0 to a set $S = \{s_1, ..., s_n\}$ of target-SRRs are symbolized by a dotted box around S and one arrow leading from s_0 to S. Actions are abbreviated by their first letter.

Let $i(x, y)$ be the sensor channel at position (x, y) of the grid. Following mnemonics are used for a test vector v: $A(x, y)$ stands for

- $v[i(x, y)] = \texttt{red}$
- $v[i] = *$ for all other places i.

Thus, $A(x, y)$ stands for 'hand (as red spot) at position (x, y)'. Note that the two dimensional coordinates in this notation are only used for the convenience of the reader. The system just sees the sequence of sensor channels in some arbitrary but fixed order.

The m-net for hand-movements plus taking, carrying and releasing a block consists of two parts similar to the above "eye-hand-net". One part is responsible for movements of the empty hand. The other one is used for carrying a building-block around. The two nets are interconnected by SRRs for proper taking and releasing of the block, i.e., `grip` is only used if the block is under the hand, `ungrip` is only used if the block is held. So the SRRs of this m-net fall into the following groups:

- Movements of the empty hand
  - $A(x, y)/\texttt{north}/A(x, y + 1)$
  - $A(x, y)/\texttt{south}/A(x, y - 1)$
  - $A(x, y)/\texttt{west}/A(x - 1, y)$
  - $A(x, y)/\texttt{east}/A(x + 1, y)$
- Taking a building-block
  - $B(x, y)/\texttt{grip}/C(x, y)$
- Releasing a building-block
  - $C(x, y)/\texttt{ungrip}/B(x, y)$
- Carrying a building-block around
  - $C(x, y)/\texttt{north}/C(x, y + 1)$
  - $C(x, y)/\texttt{south}/C(x, y - 1)$
  - $C(x, y)/\texttt{west}/C(x - 1, y)$

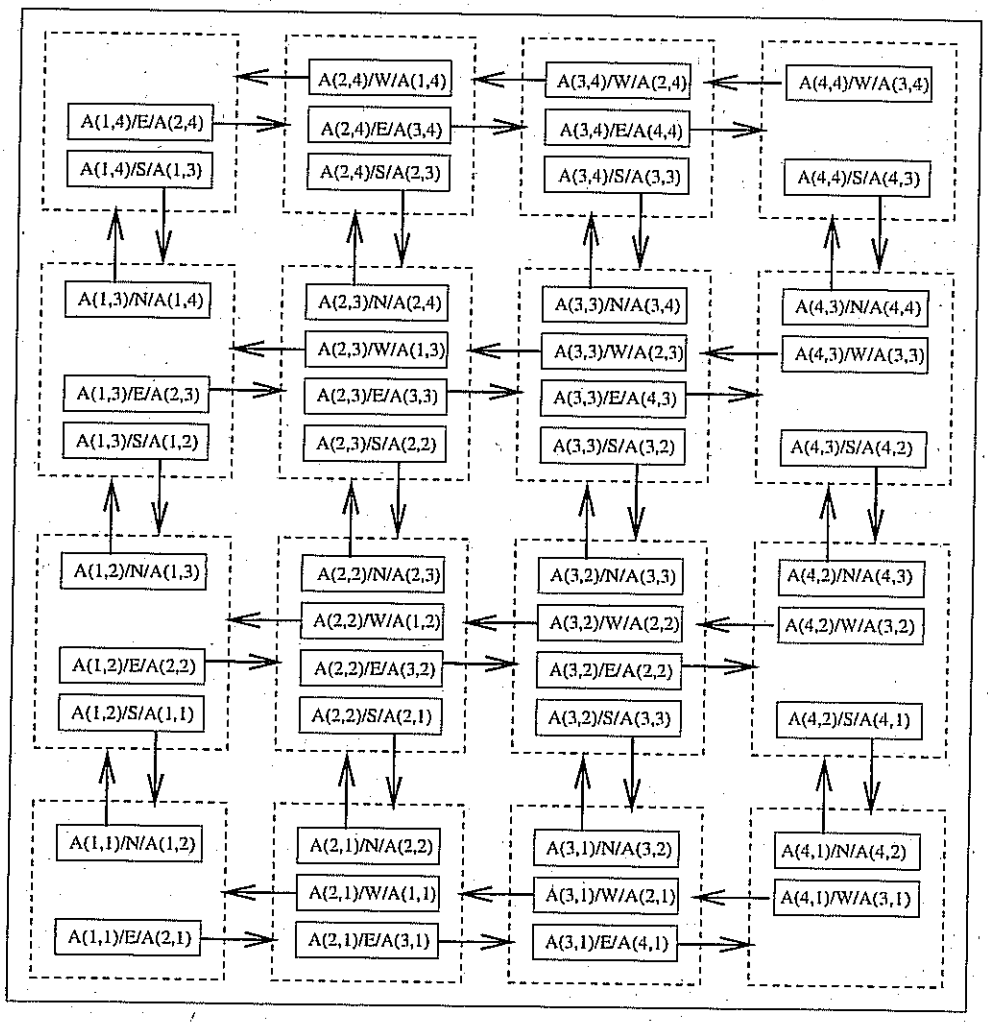

Figure 4. An m-net representing hand-movements based on a 4×4-grid of vision-channels. A detailed explanation can be found in the text.

– $C(x,y)/\texttt{east}/C(x+1,y)$

where $B(x,y)$ denotes the vector v testing whether the hand is at position (x,y) and whether it feels a touch through the sensor channel n^2+1, i.e.,

- $v[i(x,y)] = \texttt{red}$
- $v[n^2+1] = 1$

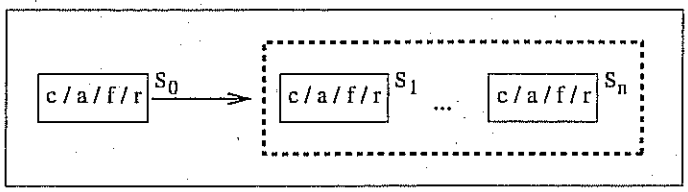

Figure 5. Symbolization of several edges, each of them leading from the SRR s_0 to the SRRs s_1 to s_n in the dotted box.

- $v[i] = *$ for all other places i

and $C(x, y)$ denotes the vector v testing whether the hand holds a block at position (x, y), i.e.,

- $v[i(x, y)] =$ `blue`
- $v[i] = *$ for all other places i.

Be again reminded that coordinates are used only in this mnemonic representation for the convenience of the reader. Sensor channels and test vectors are linear, so the grid structure of the vision is not "known" to the system. In order to learn the m-net described above, the system has to to discover this grid-structure.

Furthermore, the system has to solve an even more difficult task. It has to learn that hand-movements are independent of the position of the block. Let us illustrate the problem by describing the development of a proper SRR s for a hand-movement. Assume the hand is at position (2,1) and the block is at position (3,4) when s is created with action `west`. If no SRRs have been created in similar situations before, the condition c and the result r of s are created via adaption. So c is a representation of 'red spot at (2,1), blue spot at (3,4) and the rest are black spots'. The result r is a representation of 'red spot at (1,1), blue spot at (3,4) and the rest are black spots'.

During the further run, the system creates copies of s with wild-cards in c and r due to the use of mutation and crossover. In doing so, the system finds out what is essential for hand-movements. The red spot is important as SRRs have low *reliability* if they ignore it. The blue and black spots are unimportant as SRRs paying attention to them have low *applicability*. So, the SRR s' with condition "red spot at (2,1) and the rest does not matter", action `west` and result "red spot at (1,1) and the rest does not matter" will be the one getting a high score, and thus staying in the m-net as a proper representation of a hand-movement.

After the complete eye-hand m-net is learned, it can be used by the system for planning via the simple inference mechanism based on a search of a shortest path between rules in the net. It is for example possible to "tell" the system to move the hand to a certain position, to grip a building-block, or to transport the block

to a certain position, by supplying an according vector v as goal. Of course, the inference can already be tried while the system is still learning, i.e., while the eye-hand m-net is not yet complete and while it still contains a lot of "false" knowledge. Then planning simply does not always succeed.

4. Related Research and First Results

Learning Classifier Systems or short classifier systems, introduced by Holland in [13], are the within the field of evolutionary computation the approach which is most similar to SRL. Classifier systems in general are described in some detail in [15, 14, 11]. They already have been successfully used to learn several simple tasks for animats, reported for example in [7, 6, 12, 5].

Though some of this work is quite impressive, it all relies on carefully designed, task dependent fitness functions or even explicit information from a teacher. We are in contrast interested in an unsupervised acquisition of an animat-"mind". In this respect, the work of Drescher is the only directly related research up to our knowledge. He describes in [8] an approach, which is very close to SRL, at least in respect to the basic motivations and intentions.

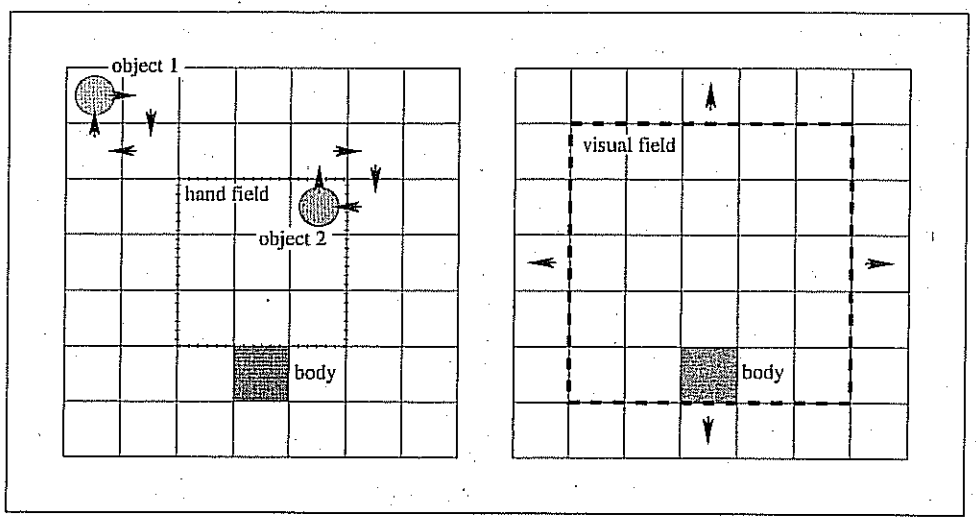

Figure 6. The simulated world used by Gary L. Drescher.

As illustrated in figure 6, Drescher uses a simulated environment which is somewhat more complex than the one described in the previous section. The robot-hand in his set-up and the center of a 5×5-grid of vision-channels can be placed on a 3×3-grid. So, hand- and eye-movements are possible. In addition, his environment

features an extra number, say η of input nerves, which are only used for speculations and not to produce results. Furthermore, two objects are present, which move occasionally. This eases their distinction from the background.

Drescher's best run on a Thinking Machines CM2 (16K processors, 512 Mbyte main memory) ended after one day with memory overflow. His system learned approximately 70% of a world-model for eye-hand-coordination. A corresponding world-model — including eye-movements and random signals on additional η extra input nerves — is found with SRL in 25 seconds on a SUN Sparc 10 completely. The total amount of memory used is less than 250 Kbyte.

We consider four points to be the main reasons why we are able to outperform Drescher's system in such a significant way:

- Fitness functions. Drescher uses a complicated mechanism to rate the usefulness of his rules. This mechanism requires a huge amount of memory, approximately 32 Kbyte per rule. Our scores just need a few bytes per rule, because computation of applicability and reliability requires only two variables per rule, namely the number of successful executions and the lifetime.
- Chaining. It is necessary to construct chains of rules which can be executed consecutively. Drescher uses logical implication between predicates with a very restricted set of tests and then works hard to 'embellish' his mechanism. We solve this problem by quick and easy statistical implication.
- Local learning. Most of our learning process takes place in a small subset of the whole world-model, namely the neighborhood of the standpoint in the m-net. Drescher has to process all the rules learned so far.
- Trying to change the perceived state of the environment. SRL creates only new rules with actions affecting the tested senor-input. The condition must not be fulfilled after execution of the action. This prevents the creation of rules like $t_{true}/a/t_{true}/t_{true}$ where t_{true} is the always fulfilled test and a is an arbitrary action. Drescher's system lacks such a mechanism.

Note that the first three items drastically affect the *asymptotic* runtime of the system as a function of n_s, i.e., the number of sensor channels.

5. Further Results

5.1. Simulations

In addition to the experiments dealing with Drescher's work, we did further research with the simulated environment. An interesting aspect is of course SRL's performance on a varying problem-size. In the Drescher-experiment from the previous section, the hand can only be positioned on a 3×3-grid, leading to an accordingly small m-net. By varying the size of the grid of vision-channels, which increments the number of possible, i.e., perceived hand-positions at the same time, it is possible

to test SRL on different problem-sizes in respect to the number of sensor-channels. Concretely, we did experiments with $n \times n$ grids of vision-channels, where n ranges from 2 to 9. Note that the problem-size as number of sensor-channels n_s is quadratic in n as $n_s = n^2 + 1$

The search space in terms of possible SRRs and edges is exponential in the number of sensor-channels n_s due to the combinatorial explosion of possible basic elements for rules, i.e., tests and actions. The concrete numbers are shown in table 1 in the columns 'possible SRRs' and 'possible edges'. The number of SRRs and edges in the desired m-net for eye-hand-coordination is in contrast quadratic in n, i.e., linear in the number of sensor-channels n_s. The concrete numbers are shown in table 1 in the columns 'eye-hand SRRs' and 'eye-hand edges'.

Table 1 shows furthermore the run-times of SRL for the learning of a complete m-net for proper hand-movements, grasping, and moving of building-blocks as described in subsection 3.2. The experiments were run on a SGI Indy, a rather low-end computer. The absolute time values given are averages of respectively five runs for each $n \times n$ grid.

Table 1. Learning Hand-Movements and Grasping

grid-size	possible SRRs	possible edges	eye-hand SRRs	eye-hand edges	ø-time h:m:s
2×2	$3.9 \cdot 10^5$	$1.5 \cdot 10^{11}$	24	72	0.7
3×3	$4.1 \cdot 10^{11}$	$1.7 \cdot 10^{23}$	66	250	3.6
4×4	$1.1 \cdot 10^{20}$	$1.2 \cdot 10^{40}$	128	528	32.6
5×5	$9.6 \cdot 10^{30}$	$5.8 \cdot 10^{61}$	210	906	1:53
6×6	$1.3 \cdot 10^{44}$	$1.8 \cdot 10^{88}$	312	1384	5:04
7×7	$6.0 \cdot 10^{59}$	$3.6 \cdot 10^{119}$	434	1962	10:27
8×8	$1.6 \cdot 10^{68}$	$2.6 \cdot 10^{136}$	576	2640	48:47
9×9	$2.1 \cdot 10^{98}$	$4.2 \cdot 10^{196}$	738	3418	2:03:59

In each run, the systems succeeded in learning a complete and proper m-net. Figure 7 shows the run-times in terms of system-steps t for different $n \times n$ grids. They appear to be quadratic in n (left side), i.e., they appear to be linear in the problem-size n_s (right side). The absolute run-time of a system-step $t \rightarrow t + 1$ showed to be rather constant within each particular learning experiment. This observation can be explained by SRL's local learning on a limited subset of the whole m-net, namely the neighborhood of the standpoint. Concretely, the absolute run-time of a system-step is in the range of a few milli-seconds on low-end computers. This is important as we are interested in using SRL for on-line learning, i.e., learning takes place in real-time while an action is executed, on animats, i.e., on concrete robotic devices.

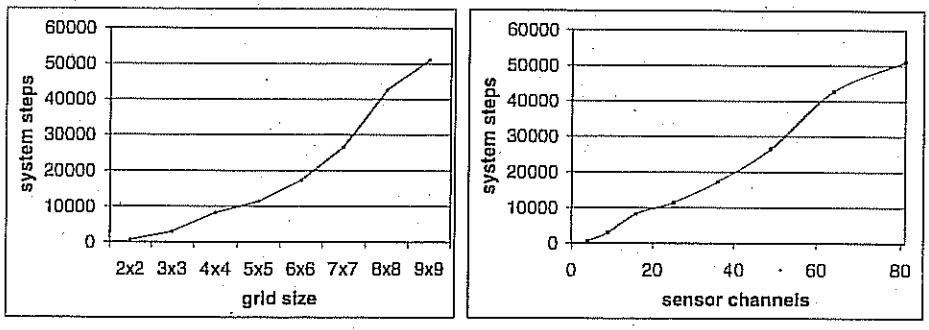

Figure 7. Runtimes of learning eye-hand-coordination with SRL on varying problem-sizes, i.e., with different $n \times n$ grids (left side). Note that the problem-size in terms of the number of sensor channels n_s is quadratic in n, therefore the run-times are also displayed in dependence of n_s (right side).

5.2. The Real-World Set-Up

In addition to the experiments with the simulated set-up, SRL was also tested in real-world experiments with a camera and a COBRA robot-arm. A simple preprocessor transfers the high-resolution camera images into data for the $n \times n$ vision-channels. The image is partitioned in n^2 squares and color information is reduced to tree bits, i.e., eight colors are possible. For each square the most frequent color is determined and fed into the according sensor channel.

The run time of these experiments is clearly dominated by the speed of the robot-arm. Compared with the simulated environment, the absolute run-time increases by a factor of approximately 500. As mentioned before, the absolute run-time of a learning-step is in the range of a few milli-seconds, so learning while performing an action is obviously feasible. In all experiments with the real-world set-up, SRL successfully dealt with noise and errors.

An additional interesting outcome of the real-world experiments is the usefulness of the 'feedback' of TOTEs as SRRs. In the simulations, schemas with condition, action, and result seemed to be sufficient representations for SRRs. In simulations, it is easy to let the hand "jump" from one field of the vision-grid to the next one. In a real-world set-up this would require very careful calibration of the system, making a learning mechanism of very limited use. The feedback of TOTEs offers an easy way out of this situation.

In the real-world set-up, an action $a(s)$ of a SRR s moves the hand only a fraction of the length of a square of the vision preprocessor. If the feedback $f(s)$ is the negation of the condition $c(s)$, then the calibration is not needed. Instead, it is sufficient to set the constant rep_{max} slightly higher than a rough upper bound for

the number of actions needed to move the hand from one field of the vision grid to the next one.

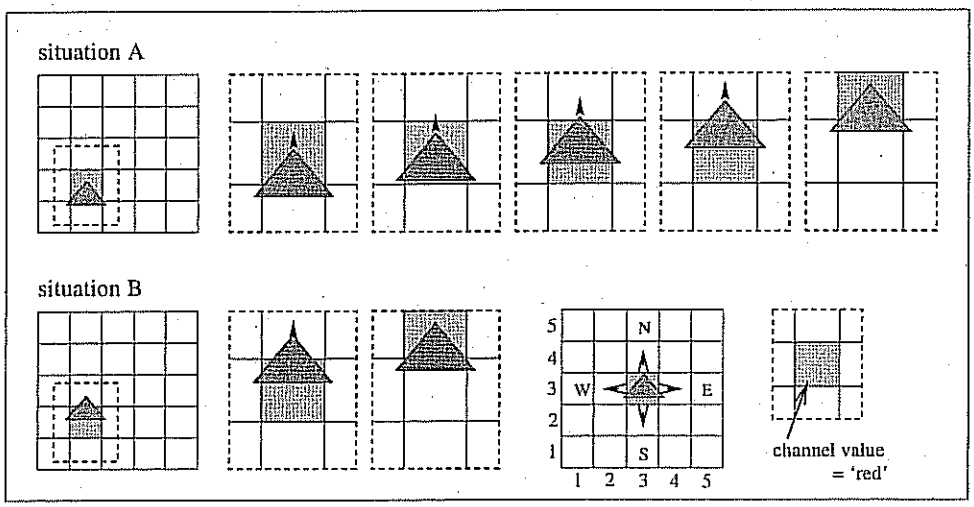

Figure 8. Using the 'feedback'-test of TOTEs in the real-world set-up.

The basic principle is illustrated in figure 8. It shows two situations where $A(2,2)$ holds, because a part of the hand of sufficient size is in the corresponding square of the preprocessor. In situation A, the hand is in a southern part of the square after execution of the TOTE "$A(2,1)/N/notA(2,1)/A(2,2)$". In situation B, the hand is in a "northern" part of the square after execution of the TOTE "$A(2,3)/S/notA(2,3)/A(2,2)$". As shown in figure 8, the hand is moved by the TOTE "$A(2,2)/N/notA(2,2)/A(2,3)$" to square $(2,3)$. In situation A, four activations of the action north are needed, only two are sufficient in situation B.

6. Conclusion and Future Work

In this article, stimulus response learning or short SRL is presented. It allows an embodied system or animat an unsupervised learning of the structure of its environment. Experiences, as relations between arbitrary sensor and motor data, are represented in a data-structure which consists of a dynamic directed graph, the so-called m-net. The nodes of the m-net are simple rules for situated action control, the so-called stimulus response rules or short SRRs. An edge between two rules represents possible consecutive execution of the rules. The rule executed last is called the standpoint. Its location in the m-net models the system's current (abstract) position in the world. A simple form of inference is done via a search of

a shortest path between two SRRs with desired properties. This inference can be used for planning by executing rules along a path from the standpoint to a goal.

The basic elements of the rules are generated with an evolutionary algorithm. This is done locally, i.e., in every time-step only a small portion of the world-model learned so far is processed to produce new rules. This process as well as the evolution of edges is guided by the two simple measures, namely reliability and applicability. Both measures are purely statistical and independent of the environment.

The experimental results presented deal with hand-eye-coordination, i.e., control of a robot-gripper via a camera in a block world. One class of experiments is done with a simulated environment in the spirit of Drescher [8]. Complete world models are quickly found in situations were the original schema mechanism ran out of computational resources. This in a sense successfully completes the experiments described in [8].

Further results from experiments in simulations and a real-world set-up are presented. In these experiments, the problem-size is varied. Though the search-space increases exponentially, the results with SRL are very encouraging.

The experiments described in this article deal with a single set-up. They are complex, situated in the real world, and allow to demonstrate the potential of simple planning. The next step is to use SRL in additional scenarios.

In the VUB AI-lab, autonomous mobile robots are situated in an artificial ecosystem [25, 18], where they face concrete tasks. The robots have to autonomously re-charge, they have to avoid physical damage, and so on. In [2] work is described where the robots learn basic reactive behaviors including photo-taxis towards the charging-station (which is marked by a white light), obstacle-avoidance, etc. The learning algorithm uses explicit reinforcement based on the energy-level of the robot.

The application of SRL within this framework bears a promising potential for future research. Similar to the eye-hand-grid, a mobile robot can build up a kind of "map" which *dynamically* relates "objects" (in terms of sensor-input) to motor-activations through *unsupervised* learning.

Such a world-model can significantly improve the performance of a robot. Take for example the reactive behavior of photo-taxis towards the charging-station. It is limited to a range of 1 to 2 meters around the charging-station where the white light can be perceived by the sensors. Therefore, a "hungry" robot which is not within this circle or has the wrong orientation has to perform a random-walk. With a world-model this waste of energy could be avoided. Instead, it would be possible to infer a chain of actions that result in "being in the charging-station", an environment state which can be represented by a test on a sensor-channel for in-flowing electrical current.

Acknowledgments

Part of this research was funded by the Leibniz program of the DFG. The experiments described in this paper have been carried out at the Universität des Saarlandes, Saarbrücken.

References

1. Andreas Birk and John Demiris, editors. *Learning Robots, Proceedings of EWLR-6.* Number 1545 in LNAI. Springer, 1998.
2. Andreas Birk. Robot learning and self-sufficiency: What the energy-level can tell us about a robot's performance. In *Proceedings of the Sixth European Workshop on Learning Robots*, LNAI 1545. Springer, 1998.
3. Rodney Brooks. Achieving artificial intelligence through building robots. Technical Report AI memo 899, MIT AI-lab, 1986.
4. Rodney Brooks. Intelligence without reason. In *Proc. of IJCAI-91.* Morgan Kaufmann, San Mateo, 1991.
5. Marco Colombetti and Marco Dorigo. Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence Journal*, 1993.
6. Marco Colombetti and Marco Dorigo. Training agents to perform sequential behavior. *Adaptive Behavior 2(3)*, 1994.
7. Marco Dorigo. Alecsys and the autonomouse: Learning to control a real robot by distributed classifier systems. *Machine Learning, 19*, 1995.
8. Gary L. Drescher. *Made-up minds, A constructivist approach to artificial intelligence.* The MIT Press, Cambridge, 1991.
9. L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligence through Simulated Evolution.* Wiley, New York, 1966.
10. Karl Pribam George Miller, Eugene Galanter. *Plans and the structure of behavior.* Holt, Rinehart & Winston, New York, 1960.
11. David Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning.* Addison-Wesley, Reading, 1989.
12. J. Grefenstette and A. Schultz. An evolutionary approach to learning in robots. In *Proceedings of the Machine Learning Workshop on Robot Learning.* New Brunswick, NJ, 1994.
13. John H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, 1975.
14. John H. Holland. Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In Tom M. Mitchell Jaime G. Carbonell, Ryszard S. Michalski, editor, *Machine Learning: An Artificial Intelligence Approach, Volume II.* Morgan Kaufmann, 1986.
15. John H. Holland. *Adaption in Natural and Artificial Systems (2nd ed).* MIT Press/Bradford Books, Cambridge, 1992.
16. John R. Koza. *Genetic programming.* The MIT Press, Cambridge, 1992.
17. John R. Koza. *Genetic programming II.* The MIT Press, Cambridge, 1994.
18. David McFarland. Towards robot cooperation. In Dave Cliff, Philip Husbands, Jean-Arcady Meyer, and Stewart W. Wilson, editors, *From Animals to Animats 3. Proc. of the Third International Conference on Simulation of Adaptive Behavior.* The MIT Press/Bradford Books, Cambridge, 1994.
19. Jean Piaget. *The origins of intelligence in children.* International Universities Press, New York, 1952.
20. Wolfgang J. Paul and R. Solomonoff. Autonomous theory building systems. *Annals of Operations Research*, 1994.

21. Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Fromman-Holzboog, Stuttgart, 1973.
22. L. Steels and R. Brooks, editors. *The "artificial life" route to "artificial intelligence".Building situated embodied agents.* Lawrence Erlbaum Associates, New Haven, 1993.
23. Hans Paul Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutions-Strategie.* Birkhäuser, Basel, 1977.
24. J.P. Seward. An experimental analysis of latent learning. *Journal of Experimental Psychology*, 39:177–186, 1949.
25. Luc Steels. A case study in the behavior-oriented design of autonomous agents. In Dave Cliff, Philip Husbands, Jean-Arcady Meyer, and Stewart W. Wilson, editors, *From Animals to Animats 3. Proc. of the Third International Conference on Simulation of Adaptive Behavior.* The MIT Press/Bradford Books, Cambridge, 1994.
26. Stewart W. Wilson. The animat path to ai. In *From Animals to Animats. Proc. of the First International Conference on Simulation of Adaptive Behavior.* The MIT Press/Bradford Books, Cambridge, 1991.

Received Date
Accepted Date
Final Manuscript Date