Multiresolutional Intelligent Controller for Baby Robot

J. Albus, A. Lacaze, A. Meystel[1]

Intelligent Systems Division, NIST

United States Department of Commerce

albus@cme.nist.gov

Abstract

This paper presents an algorithm of unsupervised learning for applications in robotics. Minimum initial knowledge is presumed ("bootstrap knowledge"). The learning system uses the newly arrived information to extract rules of motion and construct the World Representation. The concept of recursive generalization is explored as the main tool of rule extraction and knowledge organization. The experiment in learning is described based upon simulation of a 2D and a 3D mobile system.

I. Introduction: The Baby Robot Concept

The architecture of the Intelligent Controller is accepted following the results from [1, 2]: i.e., it is based on a multiresolutional system of nested loops ("perception - World Representation - Decision Making - Actuation - World - Sensors). This system has not been equipped with learning; our paper tries to close the gap which is usually avoided by declaring all necessary knowledge available and all sensor information interpretable. The later properties do not hold in any real system, thus, a learning system is vitally important to make the control system fully intelligent.

The concept of Baby Robot was first introduced in [3, 4] and further developed in [5, 6]. The autonomous robot is presented as a "baby" – that is, a system with no *a priori* knowledge of the world in which it operates, but with behavior acquisition techniques that allow it to build this knowledge from the experiences of actions within a particular environment. The challenge is to determine whether the World Representation can be constructed without any external human-interpreter, based only upon its own experience of functioning. The learning techniques are rooted in a recursive algorithm for inductive generalization of nested schemata molded from processes of early cognitive development in humans. The algorithm extracts data from the environment, and after using clustering and abduction, it creates schemata interpreted as control rules. This system is robust enough to deal with a constantly changing environment; such changes provoke the creation of new schemata by further generalizing from experiences. The algorithm is capable of automatically generating the hierarchy of task decomposition, a job that is performed in existing systems only by organizing the knowledge of a human-expert.

The system searches in a multiresolutional knowledge base which is increase as the system acquires more knowledge. It is proved in [7] that the complexity of search is significantly reduced by using a mutliresolutional structure. Although the traditional approach to autonomous navigation involves off-line path planning with a known world map, in most of the real tasks the environment is not well known. The conditions of the assignment are ever-changing, for example: absence of gravity in the case of space; and sophisticated, hard to predict obstacles; like nonlinear hydrodynamics in the underwater case. Baby Robot gathers data from its sensors and then uses a schema-discovery system to extract concepts, form schemata and create a quantitative/conceptual semantic network.

Experimenting with Baby-Robot is important in unknown media because the rules of required behavior do not coincide with human intuitions. This concept reflects our expectation that the simulated device can acquire previously unknown schemata from the results of its own experience. When the Baby Robot is first placed onto a terrain, into space, or under water, it does not have any experiences and its sensors and actuators do not have any special meaning for Baby Robot: it will learn their meaning from its own experience. At first randomly, then in a directed manner (when new schemata are created) Baby Robot collects experiences that allow it to learn schemata to represent the function of its actuators and sensors. Then a decision making algorithm activates these schemata to achieve the goal given by its creator, or the sub-goals that it finds.

In our initial simulation experiment the goal of functioning is to minimize the distance to a beacon.

[1] On Sabbatical, from Drexel University, Philadelphia, PA 19104

The results of simulation are positive. Baby Robot displays the ability to learn a number of maneuvers and to build a multiresolutional knowledge base ready for subsequent use.

II. Structure of the Intelligent Controller

II.A. General Tools of Intelligent Controllers

Most (if not all) algorithms in the intelligent controller fit one of the following classes:

Focusing Attention algorithms They separate the relevant (the important), and assigns the rest as "beyond the current scope".

Grouping algorithms They gathers entities that are close (with respect to an assigned goodness measure) and assigns them to be a whole.

Search algorithms They simulate alternatives of possible futures.

Instantiation algorithms They take the groups stored in memory and try to rebuild the elements.

All four kind of algorithms are believed to be necessary for a system to be intelligent.

II.B. The Creation of Languages in the Intelligent Controller

The intelligent controller works as follows (Figure 1):

1. Experiences are collected. Experiences are statements in which observed events are recorded in their spatial and temporal sequentiality. The simplest kind of experiences can be defined as triplets that contain a situation (or state) an action applied, and a new situation achieved after the action. It is not possible to conclude from one experience that the changes (before and after the action) are caused by the action.
2. Focusing attention. Most realistic environments are sufficiently rich to computationally overwhelm any learning system. In order to be able to solve problems it is necessary to reduce the space (focus attention) to a manageable complexity. The focusing of attention is strongly related with the existence of a goal that the system must achieve. In our case it filters out all the mediocre (and bad) experiences. An effort has been made also to learn from bad experiences (rules about what not to do) but it will not be presented in this paper.
3. Once we have only good experiences, its intention is to find what is the reason for these experiences to yield these good results and thus create hypotheses that should allow it to achieve the goal. The first step towards the creation of hypotheses is the creation of groups of "good" experiences. The groups are created by clustering, treating the situation in which the action was applied and the action applied as a multi-dimensional vector where each dimension corresponds to a sensor or actuator. Then standard clustering algorithms can be used. The clustering algorithm assigns new labels to each of these groups.
4. Each class is composed of a group of experiences where similar actions have been applied to similar situations and all of them yielded good results. We can now induce that the cause for the good result to happens (maybe) is that when that action is applied to that situation it yield good results. It may also mean that it does not have enough data to conclude anything.
5. in the groups of experiences, it takes the groups of situations and store them as new concepts in our database. Separately it takes the groups of actions and stores them in the database. These new words are part of a new lower resolution language. The system creates it because it needs these new concepts to create rules of action. We believe that the only reason for concepts to emerge in any learning system is that they can be used to create behavior, to achieve (or strive to achieve) a goal.
6. The concepts in these database create a new mower resolution language that was created by clumping, or clustering the original (actuators and sensors) language in response to the need to achieve the assigned goal. It is important to point out that each concept in this new language can be described with with concepts of the original language. We can say that a language is "included" in the second.
7. the second product of the cause effect relationships give is the creation of hypotheses by inverting the causalities. The cause effect relationships in 5) can be expressed as follows: I applied an action (in the class) when I was in a situation (in the class) and I received "good" goodness for the assigned goal. We define a function that assigns a certainty that the action applied at that situation is the cause for the goodness perceived. This certainty function is directly proportional to the amount of experiences on the class, and inversely proportional to the amount of experiences that fit the description of the class of situations and fit the description for the class of actions but they give a "bad" goodness (these experiences had already been filtered out in step 2).

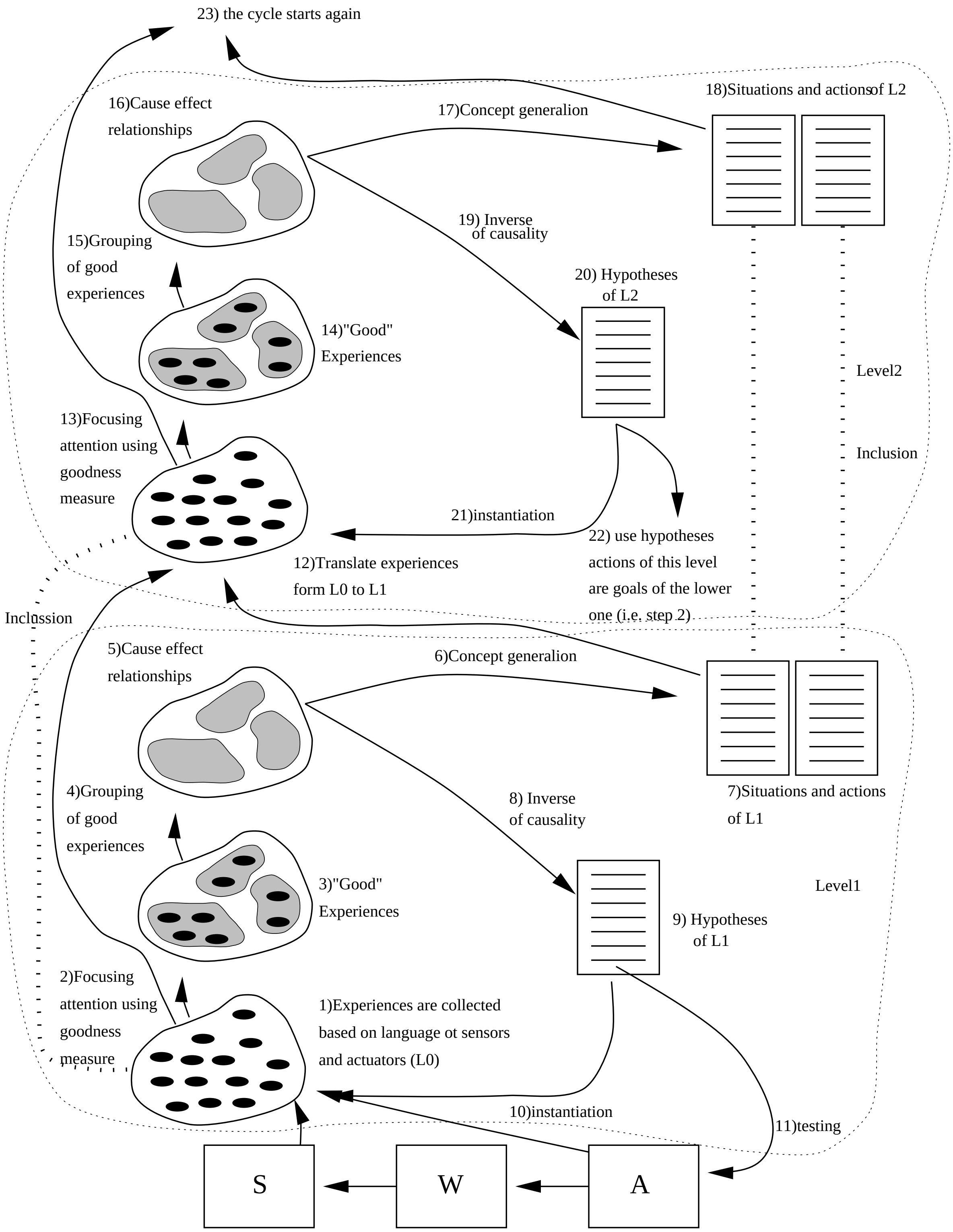

Figure 1: Architecture of a learning subsystem

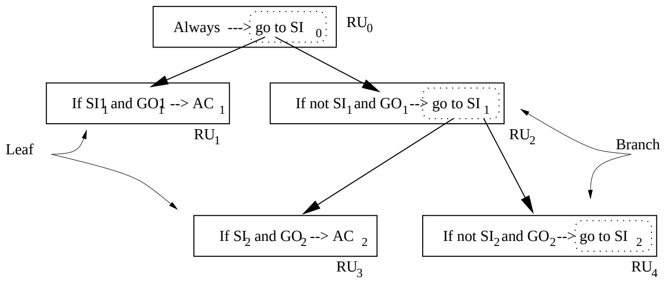

Figure 2: Format of rules in WM

Given that the this certainty function is above a certain threshold that we arbitrarily assign, then a strong argument can be made for this action being the cause of the goodness perceived (when applied to that situation). In that case we can say the following: if I am in that situation, and I want to achieve high goodness for an assigned goal then I should apply the action. A second hypotheses can also be derived: if I am not in the situation and I want high goodness for the assigned goal then I should achieve the situation. We will see in the later that this second kind of rules creates new subgoals. The causalities have been inverted into rules.

8. the hypotheses are stored in a database of hypotheses with their corresponding certainty measure. This certainty measure will be constantly updates given that new experiences are collected. The certainty measure is also used to decide which rule should be applied.
9. The hypotheses are compared agian with the new experiences that arrive to update their certainty measure. Some of them will be thrown away.
10. A behavior generation method uses the found hypotheses for control. Thus creating new experiences.
11. since we have created a new coarser language in step 7), it can describe all the experiences in 1) in this new coarser language. This creates a complete new level where the experiences are coarser. and a similar loop of experiences, to good experiences, to clusters of good experiences, to new concepts to new hypotheses is created. The difference between the two levels is that the second one processes everything with a coarser resolution, and creating an even coarser language.

II.C. Decision Making Algorithms

Lets assume that rules of action have already been created by the learning subsystem. Later in this paper, we will describe how the rules in WM are created by clustering experiences.

Rules in WM have the following characteristics (see Figure 2):

1. Rules (RU) in the database (hypotheses or schemata) fall in one of the following categories:

   Leaf: If situation (SI) and goodness (GD) for goal (GO) is desired the apply action (AC). Where SI is a set of sensor values (or groups), GO is the goal situation, AC is a vector in the actuator space, and GD is a measure of expected performance with respect to that goal. In Figure 2, RU_1 and RU_3 are two examples of leaves. In the example SI_1, SI_2, AC_1, and AC_2 are situations and actions found by the generalization algorithm.

   Branch: If not I am not in SI and my goal is GO then go to SI. In Figure 2, RU_0, RU_2 and RU_4 are two examples of branches.

   The rules at each level are stored as a goal decomposition hierarchy.
2. The following laws are used to form the hierarchy of rules:
   (a) Given a branch rule RU_n its sons are composed of leaves and/or branches with GO equal to SI belonging to RU_n. In Figure 2, RU_3 and RU_4 are sons of RU_2, thus their goal GO_2 is *go to* SI_1.
   (b) The lowest resolution level of the hierarchy has only one rule RU_0 which has the form: "If $\emptyset$ then go to SI_0", in other words "Always go to SI_0" where SI_0 is the goal assigned by the creator of the system.
   (c) Only branches can have sons.
   (d) Every branch RU_n has a brother (same father and same level) with the opposite SI and a recommendation about what to do in SI. In Figure 2, RU_1 and RU_2 are brother; same as RU_3 and RU_4

The purpose of the decision making algorithm is to find the correct consecutively assigned set of actions that will make the system achieve the first goal (the one in RU_0).

The greedy algorithm one of the many ways that it is possible to search through the goal decomposition tree. The method can be explained as follows:

1. Assign the lowest rule (RU_0) number to a variable $aux = 0$.
2. Pick a son (RU_n) of RU_{aux} that has a SI that includes the current situation and has the highest goodness among its brothers. Assign $aux = n$. Branches have goodness 0.
3. If RU_{aux} is a leaf, exit and apply the action recommended by RU_{aux}

1. Goodness	10. D-R	19. D+G
2. Distance (D)	11. H-A	20. D+R
3. Heading (H)	12. H-G	21. H+A
4. Angle to Target (A)	13. H-R	22. H+G
5. Go forward (G)	14. A-G	23. H+R
6. Rotate(R)	15. A-R	24. A+G
7. D-H	16. G-R	25. A+R
8. D-A	17. D+H	26. G+R
9. D-G	18. D+A	

Table 1: Sensors and Actuators in Baby Robot

4. RU_{aux} is a branch, thus go to 2.

This algorithm is simple and quick which is important because it has to be applied at each time step. We have implemented other search algorithms like A* but at this stage of development they do not give greater advantages over the greedy search and they are significantly slower. We also tested a non-consecutive procedure using multidimensional clustering and we found it less efficient and not consistent with the concept of early learning development. The examples given in Section 6 were implemented using greedy search.

III. Simulation of the Intelligent Controller with Unsupervised Learning

III.A. Structure of the Platform and Goal

A simulation for a mobile platform was implemented to test the generalization algorithm. The sensors return the distance to target (D), the heading (H), and the angle to target (A). The actuators are simply go forward (G), and rotate (R). The assigned goal is "make D=0".

Table 1 shows the sensors and actuators (and combinations of them) that will be used by the generalization algorithm. This notation will be used for all figures.

III.B. Step by Step Recursive Generalization

The first step that the algorithm of generalization does is to search the database of hypotheses to see if there is a hypothesis that can be applied in the current situation. The only schema present in the database says that it should "Make D=0". Since there is nothing in the database about what to do in order to "Make D=0" it assigns "D=0" as the goal, and collects a random sequence of experiences. The next step in the algorithm of generalization is the creation of the classes of experiences. The clustering algorithm discovers two classes. They correspond to the actuator "go forward" and the enhanced representation sensor "Heading - Angle to Target". These

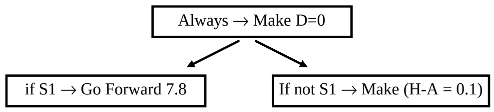

Figure 3: The database of Hypotheses at this step

two clusters create two new hypotheses: "If -0.1 < Heading - Angle to Target < 0.1 then go forward" and "if not -0.1 < Heading - Angle to Target < 0.1 then make -0.1 < Heading - Angle to Target < 0.1". These two hypotheses have an important difference. The first one gives a recommendation about what actuator to use in this situation. In the second one, the hypothesis says that if we are not in this situation we should go to this situation.

The problem with the second hypothesis is that there is nothing in the database of Hypotheses that gives instructions about how to "Make (H-A=0.1)". Thus, the generalization algorithm starts again if our current situation happens to match the situation described in the second hypotheses:

1. a new goodness measure is taken by checking whether the experience brought it closer or further to the new goal;
2. the new goal is to "make abs(H-A)=0.1";
3. all the experiences are re-ranked using the new goodness measure for the new goal;
4. the "best" experiences are selected;
5. these experiences are sent again to the classification algorithm, in other words, new eventgrams are created for the new goal and the classification algorithm creates new clusters which will be used for creating rules to follow the new goal.

In Figure 4, we show the eventgrams for the same experiences and the new goal ("make H-A=0.1").These event-grams show in each abscissa the value of a sensor or actuator in each experiences and in the ordinates the goodness of that experience. In this case, we are showing 21 (not all) sensors and actuators (and combinations) in each experience. In other words, an experience will be represented as a line of the same height (same goodness) in each of the 21 graphs. It is easy to see that in the 6^{th} graph we can see with the naked eye some clusters of good experiences.

These 300 experiences are sent to a simple clustering algorithm which finds two clusters in coordinate 6 which corresponds to the actuator "Rotate". These two clusters become two Hypotheses that get incorporated in the database as shown in Figure 5. This figure shows a real example of the goal decomposition tree shown in Figure 2. In Figure 5 "Rotate 0.9"

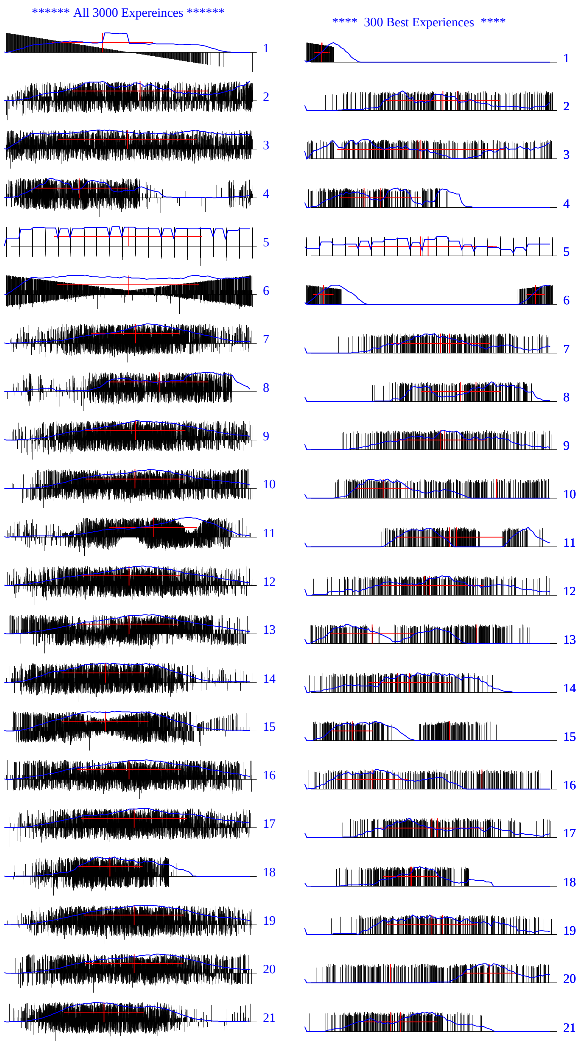

Figure 4: The eventgram of 3000 experiences (left); the eventgram of the 300 best experiences shown on the right. Note: 22 though 26 were deleted to save space

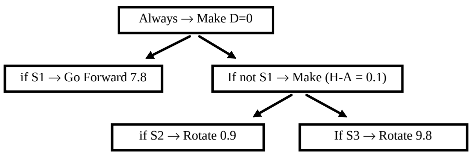

Figure 5: Database of hypotheses

and “Rotate 9.8” correspond to rotate right and rotate left. The “S2” is extracted from the experiences in which the rotate right command was applied and “S3” from the rotate left experiences.

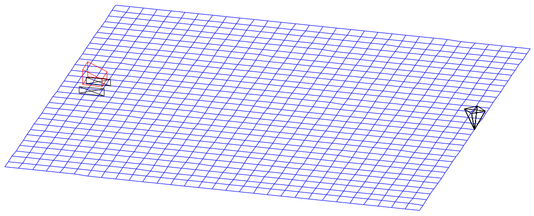

Figure 6: The Simulation Environment

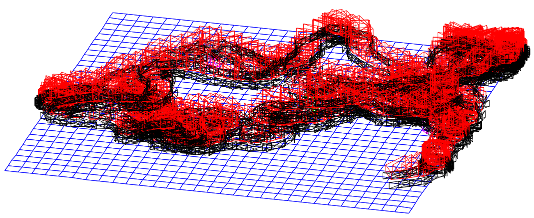

Figure 7: The initial random motion

III.C. Baby Robot for 2D Operation

The robot for 2D Operation is called the Land Baby Robot (LBR). The simulation is shown in Figure 6, on the left we can see the mobile platform and on the right it is possible to see the target. The initial goal to LBR is the touch this target.

At the begining since there are no rules in the database, the behavior is completely random. During this behavior it records experiences that later on will allow it to discover hypotheses. Figure 7 shows a random path taken initially by LBR. The traces have been left on so that the trajectory is visible.

Figure 9 shows the behavior of LBR after the first set of Hypotheses are collected. This is a typical behavior when LBR is at the development stage found in Figure 5. Hypotheses about turning and going forward have been formulated, but there is not enough experiences to correctly define the classes, that is why we see a random/spiraling movement towards the goal. The learning algorithm continues collecting numerical data from experiences that allow it to numerically improve old rules. Figure 8 shows the evolution of the ”make (H-A)=0.1” rule that was originally found and shown in Figures 3 and 5. It is possible to see that at 500 steps the rule says:”Make (H-A)=-10 degrees” and at 10000 steps we have that the same rule now is: ”Make (H-A)=-0.4 degrees”.

Figure 10 shows the numerical improvement upon the original hypothesis that yields better paths.

Figure 11 shows a quasi-optimal path that LBR converges to. At this point, LBR can successfully achieve any target in the shown region with an almost optimum path.

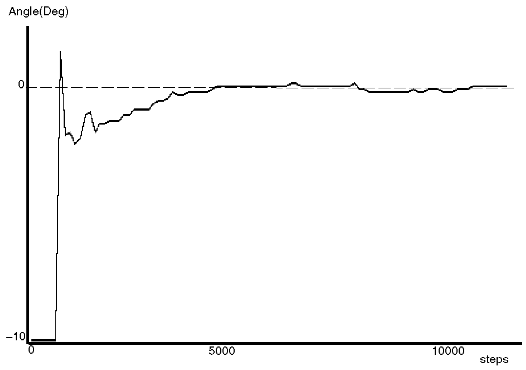

Figure 8: Numerical evolution of hypotheses as a function of time

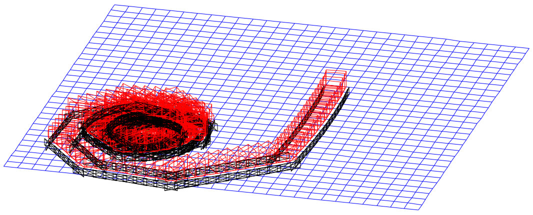

Figure 9: The first hypothesis

III.D. Baby Robot for 3D Operation

The platform for 3D operation is called underwater Baby Robot (UBR). The sensors given to the UBR are:

1. Distance to the target.
2. The three Euler angles to the target.

And the actuators given to UBR are:

1. Propulsion.
2. Angles of the two rudders.

Figure 12 shows an example of a learned hypothesis. At the beginning, there are some random movements. Then, a hypothesis is generalized, showing a spiral movement of an incorrect hypothesis of spiraling. After more experiences are collected it will realize the performance could be improved and a new hypothesis will be generalized and this one may be thrown away.

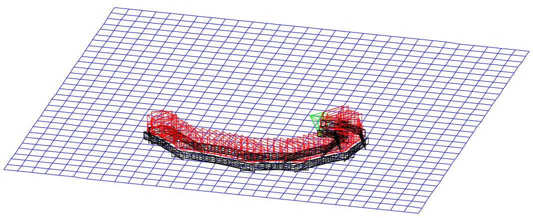

Figure 10: Numerical improvement

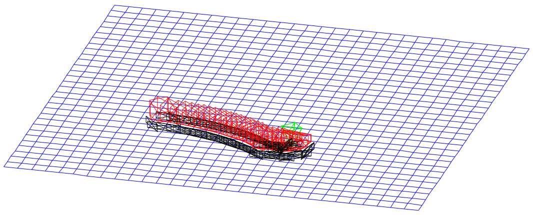

Figure 11: Quasi-optimum path

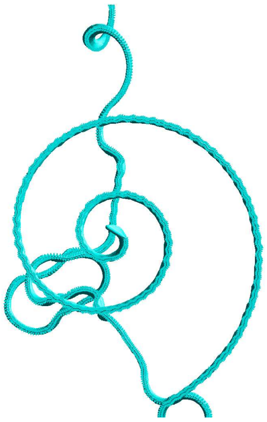

Figure 12: Random movement and wrong hypothesis of spiraling

Figure 13 shows the first trial to go to the goal. The traces were left on to show how complicated and superfluous is the path. At the beginning of this trial the learning algorithm does not have any knowledge about the environment. It is possible to see four or five different hypotheses that create different motion.

Figure 14 shows some random movements and then the bang-bang control. Note, that we did not teach our robot the concept of "bang-bang" control; it discovered it as a result of the learning process.

Figure 13: First trial

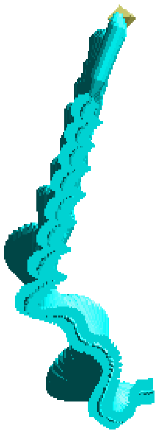

Figure 14: Bang-Bang control

After the first set of random movements we can see a sharp change in behavior (Figure 12). The first set of baby schemata is learned and we can see (Figure 14) that the submarine starts applying bang-bang control. More details about Baby-Sub and Astro-Baby can be found in [5].

IV. Conclusions

1. An algorithm that will serve as a subsystem for unsupervised learning is proposed for the Intelligent Controller Structure.
2. This system employs the method of nested clustering.
3. The system simulates and executes the process of decision making.
4. The system organizes input information into a multiresolutional structure of world representation.
5. Simulation experiments confirm the theoretical premises.

V. References

[1] J. Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21, 1991.

[2] A. Meystel. Multiresolutional architecture for autonomous systems with incomplete and inadequade knowledge representation. In H. B. Tzafestas and H.B. Verbruggen, editors, *AI in Industrial Decision Making, Control and Automation*, 1995.

[3] A Meystel. Baby-robot: on the analysis of cognitive controllers for robotics. *Proc. IEEE Int'l Conf. on Systems, Man, and Cybernetics*, 1985.

[4] J. Eilbert, A. Meystel, L. Venetsky, and S. Ziets. Baby robot: Learning to control goal-oriented behavior. In *Proc. IEEE Workshop on Intelligent Control*, pages 182–186, 1985.

[5] A. Lacaze, M. Meystel, and A. Meystel. Schemata for unsupervised learning for autonomous robots for 3d space operation. In *NASA Conference on Space Applications of AI*, pages 103–112, Greenbelt, MD, 1993.

[6] A. Lacaze. Unsupervised early conceptual learning in mobile robots. Master's thesis, Drexel University, 1994.

[7] Y. Maximov and A. Meystel. Optimum design of multiresolutional hierarchical control systems. In *Proceedings of IEEE Int'l Symposium on Intelligent Control*, pages 514–520, Glasgow, U.K., 1992.

[8] J. Albus and A. Meystel. A reference model architecture for design and implementation of semiotic control in large and complex systems. In *Semiotic Workshop of the 10th IEEE Intenational Symposium on Intelligent Control*, 1995.

[9] J. Albus, A. Lacaze, and M. Meystel. Autonomous learning via nested clustering. In *34th IEEE conference on decision and control*, New Orleans, Lousiana, December 1995.