N90-29889

A PROCEDURE CONCEPT FOR LOCAL REFLEX CONTROL OF GRASPING

Paolo Fiorini & Jeffrey Chang
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

Abstract

This paper proposes an architecture for the control of robotic devices, and in particular of anthropomorphic hands, characterized by a hierarchical structure in which every level of the architecture contains data and control function with varying degree of abstraction. Bottom levels of the hierarchy interface directly with sensors and actuators, and process raw data and motor commands. Higher levels perform more symbolic types of tasks, such as application of boolean rules and general planning operations. Layers implementation has to be consistent with the type of operation and its requirements for real time control. In the paper we propose to implement the rule level with a Boolean Artificial Neural Network characterized by a response time sufficient for producing reflex corrective action at the actuator level.

1. Introduction

The set of tools available to robotics researchers to build grasping strategies includes path planners, many types of control algorithms, and sensor data fusion techniques. Many authors have proposed different architectures for the integration of these tools to assure a smooth flow of information from sensing to central computing and back to actuation. A proposed system is based on **Logical Sensor Specification** [6] and consists of software layers between physical sensors, actuators and the central processor, each one of them able to perform some local processing on sensor data and to directly modify commands from the top level to the actuators. A proposal also aimed at reducing the computational load of a controller is that of **Reflex Control** [1] in which a limited number of actions are carried on by the mechanical hardware to react to specified external situations. Other authors have proposed **Expert Systems** at various levels of a layered architecture for hand control. In [15] an expert system for configuring grasp postures is proposed, and this is integrated in [9] with an **Artificial Neural Network** for learning the relations between postures and object shapes. In [7] a **Learning Expert System** is presented for the discovery and refinement of control skills for fine manipulation tasks. These proposals represent efforts to improve performance and applicability of Control Theory and can be associated with the research in **Intelligent Control** [10],[13],[14], whose main effort is the integration of Control Theory, Artificial Intelligence and Operation Research in a homogeneous structure capable of autonomous reasoning and control.

From this brief overview, it is clear that architectures based on hierarchies of processing stages are undergoing extensive study, but present results do not take full advantage of the possibilities offered by this approach. For example, reflex control alone is not sufficient to handle a large number of operating conditions, and expert systems are too slow for a direct interface with a real time process. An efficient **Intelligent Controller**, must assure immediate reactions to unexpected external conditions and thus bypass the long processing time of the higher levels. A fast reaction would compensate for the different execution times of planning and control, and fill in possible voids in the command stream. This can be of great usefulness in the case of space servicing with time delay, when the remote operator cannot provide immediate feedback, and also to free some of astronauts time.

2. Hierarchical Controller

The needs described in the above paragraphs can be satisfied with hierarchical architectures and with circuital implementations of the various layers, that can assure better flexibility and performance than current systems. This proposed control architecture consists of three levels: a planner, a rule base and the actuator controller. Each level can be composed of different layers depending on the particular application, and on the requirement of data fusion. In the implementation that we propose, each level consists of a single layer, and the rule base is included between planner and actuator controller, to have response time and amount of knowledge intermediate between those of the other two. The purpose of this stage is to store expected relations among single feedback signals or subsets of them, and to use the results to understand the evolution of a grasping task. In particular this design allows for a flexible reflex control in various grasping tasks and permits the actuator controller to determine autonomously the best reaction to a given pattern of feedback signals. Both planner and actuator controllers use feedback signals and rules output to start the generation of a new plan and to modify actuator trajectory. When a new strategy is generated by the planner, the associated rules are blended with the current rule base, to assure a smooth transition between plans.

This architecture requires several additions to the standard implementations of planner and actuator controllers. The first one, in fact, has to generate the set of conditions that qualify the task evolution, while the second has to store a set of alternate trajectories. The rule base processes the feedback signals and generates a set of boolean variables, applies the rules supplied by the planner to these variables and determines the current evolution of the task. When a replanning occurs, it assures consistency in its rule base. In this implementation, two basic assumptions have been made: that a small set of actuator trajectories can cover most of grasping situations, and that the evolution of a grasping task can be represented by logical conditions. In this case, the rule base is an adaptive boolean network in which sensor conditions are stored as if-then rules expressed as boolean conjunctions. A range detector converts data from the sensors into logical levels, and the adaptation mechanism supervises the loading of a new plan.

The complete system will include two boolean networks, managed by an adaptation unit, to ensure that adaptation does not interfere with the correct processing of the feedback signals. When the backup network is successfully updated, the adaptation controller will switch it on line and will start updating the other one. Figure 1 is a block diagram of the proposed architecture of the complete system. In the following paragraphs, the justification for such an approach will be presented, together with the description of functions and implementation of the rule base.

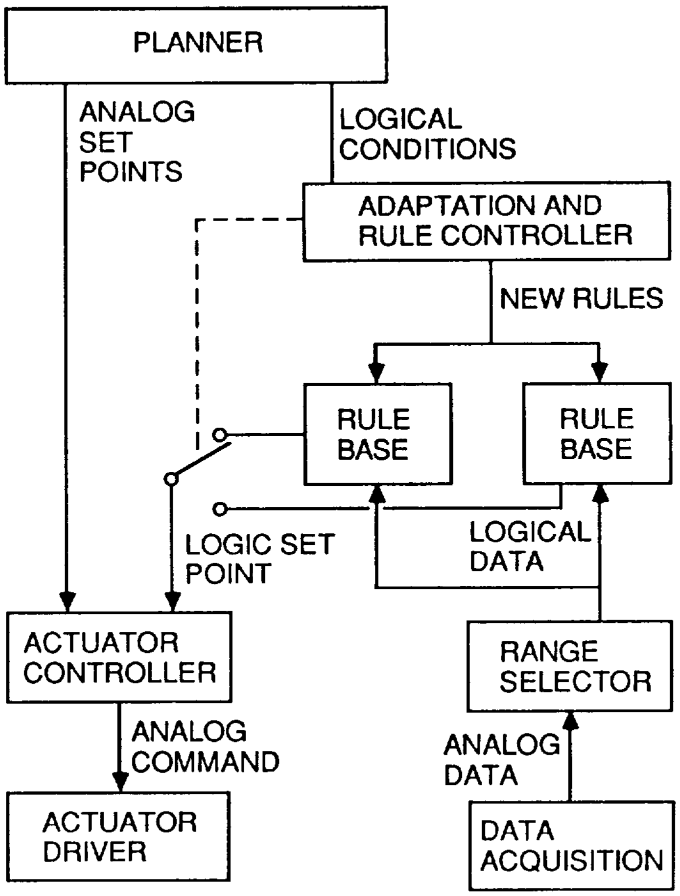

Fig. 1: Hierarchical architecture for robotic control

3. Design Approach to a Control Architecture

In defining a structure for a hand system, two main steps have to be taken: first the analysis of function allocation in the hierarchy, and second their match with candidate implementations to satisfy the needs of real time control. After the initial achievements in designing good mechanisms and control algorithms [3] [4], much effort is now directed towards the definition of an architecture that can include and organize all components of a multifingered hand system. In this situation, it comes quite natural to turn to the analysis of human behavior as a possible model for a control architecture.

In the functional analysis of [8],[11],[12] a causal hierarchy is defined, describing the different types of control actions of human operators supervising industrial plants. The layers of the hierarchy are based on increased level of symbolic representation of the information, from raw sensor readings to descriptions of plant states. At each level, the degree of complexity of the system representation is approximately kept constant. The plant operator is the processing power acting at each level of the hierarchy on different data abstractions. Depending on the case complexity, he/she can initiate a control action at every layer and, in particular, at the one whose data representation best describes the current situation. Such a system then has a capability for a particular reflex action, in the sense that commands to the plant can be generated at all levels of the hierarchy, without the need of reaching the top level of abstraction for deciding the next action. Learning of new skills is also necessary to fine tune the response of the system. A hierarchical structure has the potential for implementing a distributed supervised scheme, in which every layer can receive updates from the next higher level, and can modify the logic of the next lower level in the dual role of supervisor and learner.

The second step in designing a control architecture is the match of the functions assigned to a layer of the hierarchy with a specific implementation. In particular, the interaction between the actuator control and the strategy generated by the grasp planner is not well defined. During task execution , no provision is made for using sensor feedback to update in real time the initial plan, or for switching to alternate ones. The normal approach is to halt the operation when an error condition is recognized, and generate a new plan based on current sensor data. To answer the need for a fast activation of an alternate plan, qualitative conditions are mixed to the robot programs, defining the expected logical conditions of the task. To be more general, these qualitative or logical conditions should be generated by the task planner together with the quantitative information pertinent to the trajectory definition. In the same way that analog information is stored in the actuator controllers, logical conditions should be stored in qualitative controllers that would supervise the task evolution and would manage strategy changes. In this paper we present an example of this by assigning to one of the layers of the control hierarchy a particular class of artificial neural networks called **Dynamically Programmable Logical Arrays** [2], [16], and by implementing it in VLSI technology.

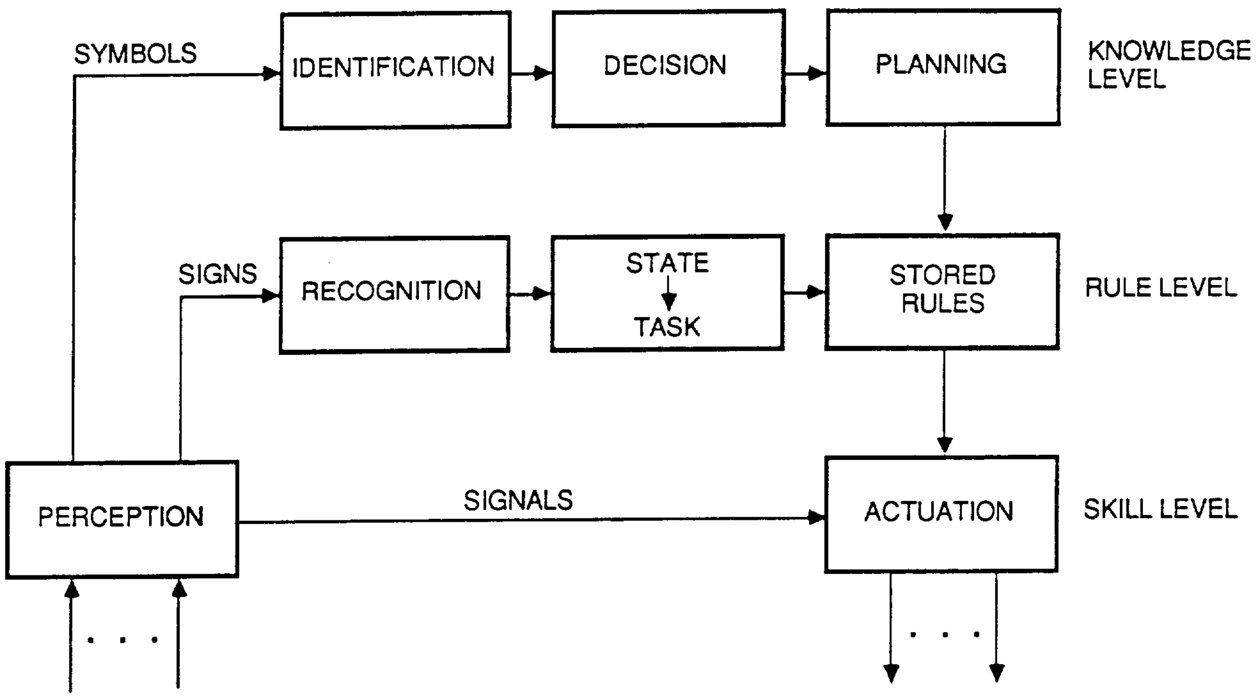

Fig. 2: The Rasmussen-Lind model

4. Functional Hierarchy

The model developed by Rasmussen and Lind and presented in [11] describes the operator's decision making process during the control of complex plants, such as nuclear or power facilities. The proposed data organization closely follows the needs of the human mind in terms of quantity and quality of information presented. Three parameters have been determined to play a major role in the modeling of a control action: causal relations, complexity of representation, and expectation on feedback data. Operators organize their mental model based upon the causal relations existing between elements of the plant, and generate several plant descriptions, each one with a different level of abstraction. In this way, the number of elements in each level, i.e. the level's complexity, can be kept approximately constant. Particular situations direct the operator's attention, and the corresponding control actions are determined by the presence or by the absence of specific feedback signals. Each hierarchical level corresponds to a different kind of mental process. At the bottom layers actions are immediately activated by important feedback signals, middle layers reasoning can recognize typical patterns in the feedback signals and command more complex reactions, and top layers mental process is dedicated to a symbolic type of reasoning in which sequences of patterns can be analyzed.

A schematic representation of this model is in Fig. 2, where three levels of this hierarchical structure are visualized. The bottom level is defined as the skill level,where no conscious action takes place, and feedback/reaction are governed by fixed and automatic relation, e.g. maintaining a level within ranges, or reacting to a particular alarm signal. At this level there is no abstraction on the feedback data, but each signal is considered alone for its particular meaning. In the middle level, some processing of the feedback information is necessary before a control action can take place. This data processing can be quite elementary, and can be visualized as a set of rules combining subsets of feedback signals. At this level, actions are decided on the basis of abstract entities derived from the rules stored at this level. The next higher level represents the knowledge level, where complex inferences have to be made. The type of processing at this level is not characterized by sets of rules, but by general planning functions. If we consider the above description as a possible model for a hierarchical control structure for a robotic system, we see that the decision process performed by the plant's operator can be used as a guide to identify both the degree of data aggregation needed at a certain level of the architecture and the type of processing most suitable for that level. This model can also be used to describe a typical sequence of actions in human manipulation.

The model can be mapped directly into a possible architecture of a control system for an anthropomorphic hand. In a simple three-level structure, the skill level corresponds to the actuator controllers, the rule level corresponds to an intermediate processing of feedback data and actuator commands, and the knowledge level corresponds to the grasp planner, such as the one presented in Figure 1.

5. Adaptive Boolean Networks

This type of network is built with node modules capable of manipulating small sets of input variables with logical operators that can be dynamically programmed to change the boolean function implemented in the node. The overall network is then a combinatorial circuit and its outputs are boolean functions of the inputs. These logical operations can be considered equivalent to propositional logic calculation, compiled into the network as logical rules relating symbolic inputs to symbolic outputs. Due to the nature of its boolean constituents, this processing is completely combinatorial, non-numeric and asynchronous. The architecture is regular with limited connectivity and modules can be easily structured by aggregating groups of functions. Adaptation is an additional feature of these networks that allows them to take an active role in configuring the connections of the logic gates, with the purpose of optimizing some performance index, such as minimality and consistency of the rule base. Dynamic programmability distinguishes these networks from conventional **Programmable Logic Arrays** which realize fixed functions after the initial programming step. They are also different from **User Programmable Gate Arrays** [5], in which the logic function embedded in the circuit can be altered by storing in the array a different set of connections for the logic gates. These arrays play no active role during reconfiguration, but they are reprogrammed on line by an external source.

The underlying concept for this class of combinatorial dataflow architectures is the same than that of **Artificial Neural Networks** (ANN); it is useful to make a brief comparison of the two structures. The prototype structure of an ANN element is a weighted sum of inputs modified by a nonlinear threshold function, while these type of networks have fully input, output and processing boolean. The main consequence of this is the simpler design of both the computing device and its control circuits. In an ANN, learning is achieved by a process of convergence following a gradient path in a multidimensional state space. In adaptive boolean networks the learning process is accomplished by adapting the current rule base to new rules presented to the network by an outside source. This can be viewed as a type of supervised learning in which the network takes the active role of blending the new rule with the old ones to assure consistency and minimality (Fig.3).

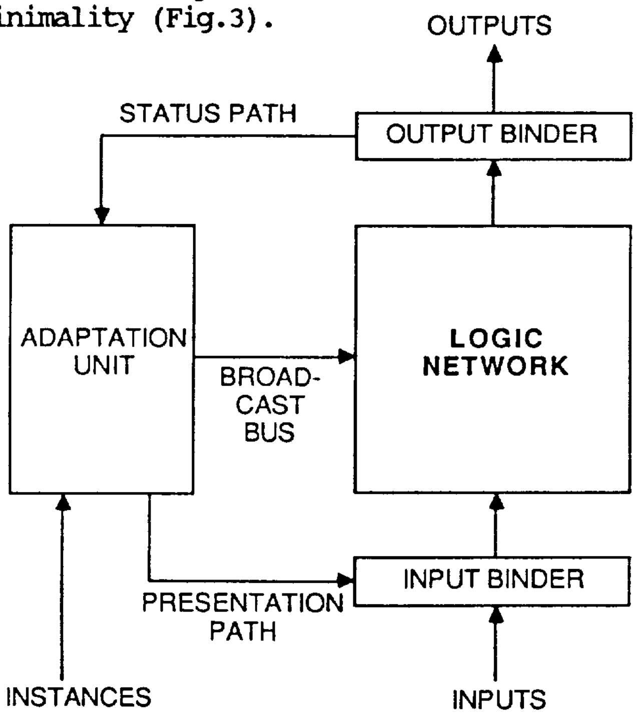

Fig. 3: Adaptive Boolean Network

Problem specification is done by incrementally entering into the system if-then rules expressed as boolean conjunctions. During processing, the network acts as a Programmable Logic Array. The network inputs are discrete variables supplied by the environment. The output of the processing is the truth values for the logic predicates which have been previously stored in the network as logic rules. During adaptation, the network structure changes to update the overall network functions. As new rules are added, the network automatically reconfigures to a logic circuit that seeks to maintain a minimal and consistent rule base. There is no explicit programming of the network, and the internal configuration of the network is not unique and depends on the initial state and on the history of the previous adaptations. The system accepts new rules that are sequentially presented to it by an external controller. This process allows each node in the network to determine its relation with the new rule and determine whether it should be involved in the adaptation process. The adaptation may involve addition or deletion of nodes, or compaction of subnetworks. A central controller is used for coordination, but the adaptive process itself is completely distributed in the network, and modifications to network are performed with considerable concurrency.

Rules consist of a conjunction of boolean variables as antecedent and of a single boolean variable as consequent, e.g.:

$$ABC \rightarrow Z$$

which means that if the antecedent is true then the consequent must be set to true. These rules differ from ordinary boolean functions, because of the characteristic of propositional logic of not specifying anything besides what the rule states. This means that no condition is set for the truth value of (not Z) and that Z can also be true in the absence of the given antecedent. Thus whenever the antecedent of a rule is not matched by the input environment, the instance provides no information about what the output of the system should be. Rule consistency is achieved by resolving all conflicts between two instances that contradict each other, i.e. if they are both eligible to fire for some state of the environment, and they have discordant output. Minimization occurs between concordant instances and only guarantees relative minimality. Minimal representation is achieved by heuristics methods, and not through procedures to derive a theoretical optimum. Minimality is said to have been achieved when no two concordant rules can be equivalently represented by one rule only, or by two rules with fewer variables.

This type of digital neural network can be implemented in several architectures [16], depending on the degree of connectivity among the nodes and the type of communication allowed between nodes and the external controller. A totally connected architecture has been extensively studied [2], and it consists of two function arrays, separated by a controller column, as in Figure 4. The first array is an AND plane, in which all nodes implement a logical AND function. The second array consists of nodes implementing the logical OR functions necessary to build the complete rule out of the minterms built in the AND plane. The central column consists of an array of nodes called D-nodes, which collect the output of a row of the AND plane, thus generating an output corresponding to a minterm of a rule. Variables are associated to nodes in the AND plane by setting a status bit in the node located at the intersection of the column, representing the variable and the row corresponding to the minterm. In a similar way, the OR plane collects the minterms forming a rule by using a chained OR to represent a logical function as a sum of products.

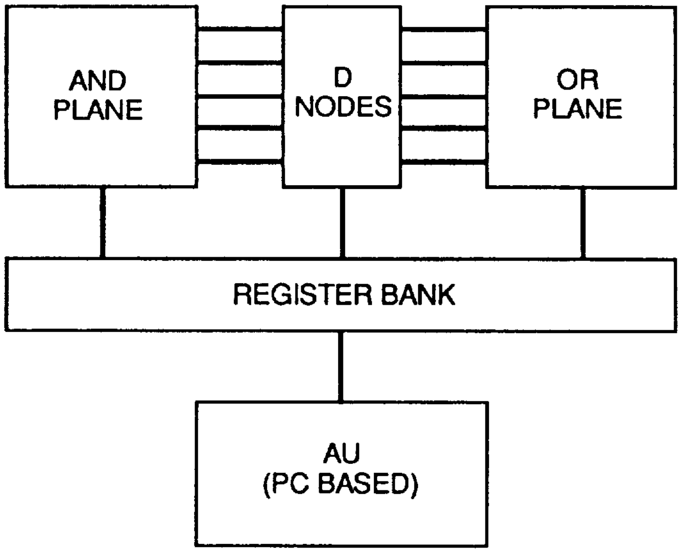

Fig. 4: System Diagram of the Prototype

The adaptation process in this architecture is a two step process. First the new rule is presented to the network by the Adaptation Unit, where each node performs a self classification to determine how it should be operated on, to bring about the correct adaptation of the network. Second, the Adaptation Unit guides the adaptation of the node pool so that a prioritized sequence of operations can perform the network adaptation. Rules are presented to the circuit as a sequence of component minterms, and they are assigned to a specific output, by enabling one of the columns of the OR plane. Complex operations, such as fusion of two rules, are done by the Adaptation Unit to which the network schedules the offending minterm for external minimization.

To experiment with integration of the rule base with an real actuator controller, the network has been implemented in VLSI technology so that it can be located with the controller and will not affect the communication bandwidth of the system. During processing, the network receives the output of range selectors that transform the analog output of the sensors into boolean variables, and then it processes them according to the memorized rules. During adaptation, the network structure changes to update the overall network function. In the present model, we use a recency law for resolving rule conflicts during the transition from a grasping strategy to the following one. The implemented prototype is a four input, four output, eight minterm network, calling for a 4 by 8 AND plane, an 8 node control column and an 8 by 8 node OR plane. The whole chip measures 7900 by 9200 microns and was designed in a 3 micron p-well CMOS technology. The chip is mounted in a 64 pin package, and is currently interfaced to a personal computer for functional testing. All input output lines are buffered to avoid racing conditions among the feedback signals. Figure 5 is a microphotograph of the chip.

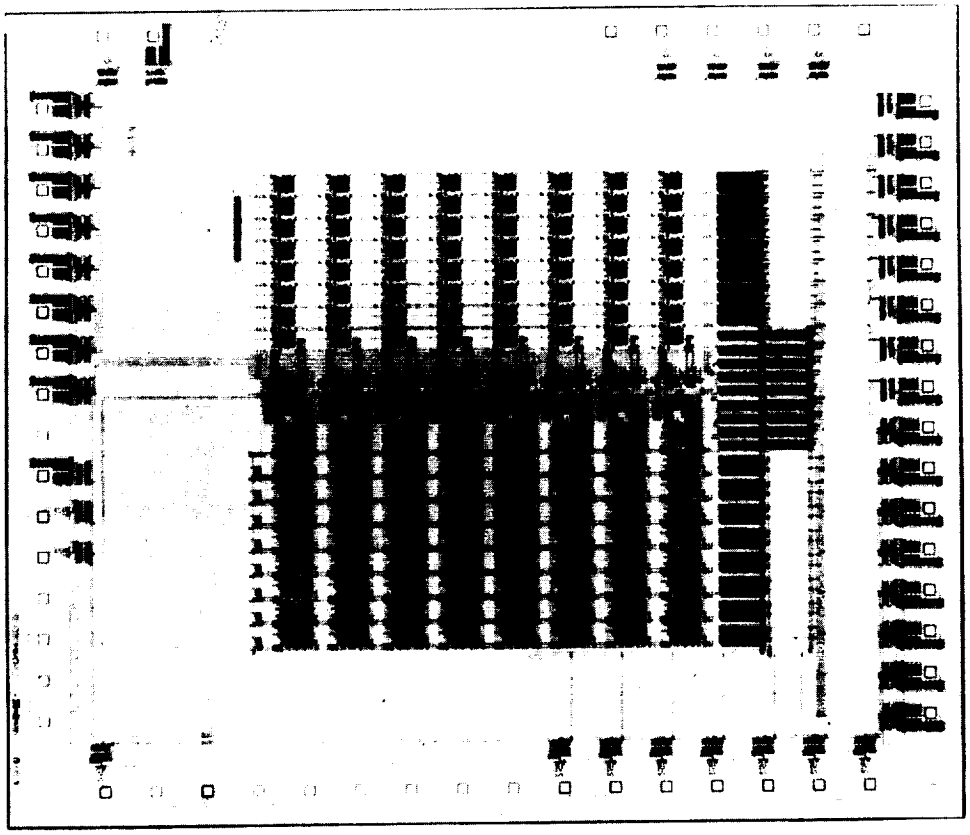

Fig. 5: Microphotograph of the Boolean Neural Network

6. Conclusion

In this paper we have presented the first steps of the design and implementation of a hierarchical architecture for the control of robotic devices such as mechanical hands. The justification for this approach is found in the analysis of human behavior during control functions, and it is an improvement over similar design proposed for intelligent controllers. The three fundamental criteria that this structure satisfies are: causal connections between layers, constant complexity of each layer, and directed focus of attention. The implementation of each layer must obey the same design criteria, and therefore it must change from one layer to the other, to fulfill the requirements of processing type and execution speed of that level. An implementation of a layer has been described, which will act as a rule base, or logical controller, in a three layer architecture, and the characteristics of this implementation as a boolean artificial neural network have been presented.

7. Acknowledgements

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

8. References

[1] Bekey, G., Tomovic, R.,"Robot control by reflex action", Proc. Int. Conf. on Robotics and Automation 1986, pp. 240-247.

[2] Chang, J., Vidal, J. J.,"Inferencing in Hardware", MCC University Research Symposium, Austin, Tx, July 1987.

[3] Cutkosky, M.,"Robotic Grasping and fine manipulation", Kluwer Academic Publisher, 1985.

[4] Fiorini, P., "A versatile hand for manipulators", IEEE Control Magazine, October 1988.

[5] Freeman, R., "User-programmable Gate Arrays", IEEE Spectrum, December 1988, pp. 32-35.

[6] Henderson, T., Hansen, C., Bhanu, B.," The specification of distributed sensing and control", Journal of Robotic Systems, 2(4), 387-396(1985).

[7] Lee,S., Kim, H.M., "Learning Expert Systems for robot fine motion control", IEEE Intelligent Control Conference, Philadelphia, PA, 1988.

[8] Lind, M., "The use of flow models for automated plant diagnosis", 1981 NATO Symposium on Human Detection and diagnosis of system failures.

[9] Liu, H., Iberall, T., Bekey, G., "Building a generic architecture for robot hand control", Proc. IEEE Int. Conf. on Neural Network, July 24-27 1988, San Diego, CA, pp. II 567-574.

[10] Meystel, A., "Intelligent control in robotics", Journal of Robotic Systems, 5(4),269-308 (1988).

[11] Rasmussen, J., Lind, M.,"Model of human decision making in complex systems and its use for design of system control strategies", American Control Conference, Arlington, VA, 14-16 June 1982.

[12] Rasmusen, J., Lind, M., "Coping with Complexity", RisO-M-2293, 1982, European Annual Conference on Human Decision and Manual Control, Deft 1981.

[13] Saridis, N. G., "Knowledge implementation: structures of intelligent control", Journal of Robotic Systems, 5(4), 255-268 (1988).

[14] Stephanou, H.E., Erkmen A.M.," Evidential classification of dexterous grasps for the integration of perception and action", Journal of Robotics Systems, 5(4), 309-336(1988)

[15] Tomovic, R., Bekey, G., Karplus, W., "A strategy for grasp synthesis with multifingered robot hands", Proc. Int. Conf. on Robotics and Automation 1987, pp. 83-89.

[16] Vidal, J. J., "Implementing Neural Nets with programmable logic", IEEE Trans. Acoustics, Speech and Signal Processing, Vol. 36, N. 7, July 1988.