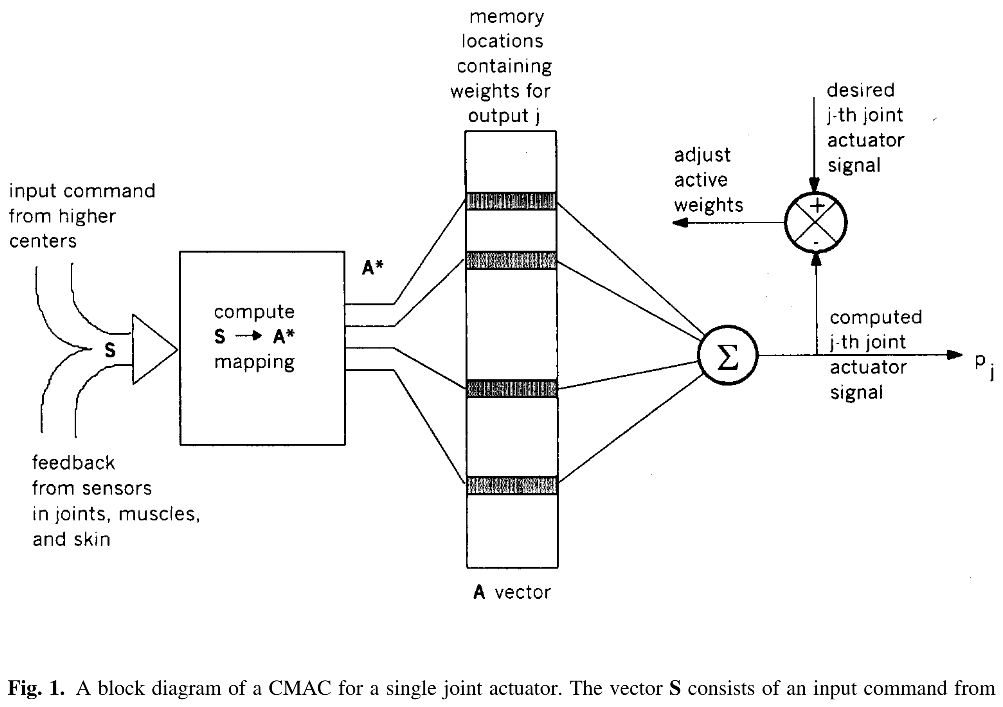
# A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller

The Cerebellar Model Articulation Controller (CMAC) [1, 2] is a neural network that models the structure and function of the part of the brain known as the cerebellum. The cerebellum provides precise coordination of motor control for such body parts as the eyes, arms, fingers, legs, and wings. It stores and retrieves information required to control thousands of muscles in producing coordinated behavior as a function of time. CMAC was designed to provide this kind of motor control for robotic manipulators. CMAC is a kind of memory, or table look-up mechanism, that is capable of learning motor behavior. It exhibits properties such as generalization, learning interference, discrimination, and forgetting that are characteristic of motor learning in biological creatures.

In a biological motor system, the drive signal for each muscle is a function of many variables. These include feedback from sensors that measure position, velocity, and acceleration of the limb; stretch in muscles; tension in tendons; and tactile sensations from various points on the skin. Feedback also includes information from the eyes via the superior colliculus and visual cortex about the positions of the hands and feet relative to their intended targets. Drive signals to the muscles also depend on higher level ideas, plans, intentions, motives, and urges. These may be specified by variables that identify the name of the task to be performed and specify the goals that are desired, the procedures and knowledge required to achieve those goals, and the priorities that have been assigned to achieving those goals.

A block diagram of a typical CMAC is shown in Fig. 1. CMAC modules are designed to accept both input command variables from higher levels and feedback variables from sensors. Each CMAC merges these two inputs into a set of memory addresses wherein are stored the correct motor response. The combined input selects a set of memory locations from a large pool of memory locations. The output is the sum of the contents



memory locations containing weights for output j
input command from higher centers
S
compute S → A* mapping
A*
adjust active weights
desired j-th joint actuator signal
+
-
Σ
computed j-th joint actuator signal
p_j
feedback from sensors in joints, muscles, and skin
A vector

**Fig. 1.** A block diagram of a CMAC for a single joint actuator. The vector **S** consists of an input command from higher motor centers combined with feedback from sensors in the joints, muscles, and skin of a limb. Each CMAC separately computes a **S** ⇒ **A*** mapping. **A*** is the set of locations in memory selected by the mapping. The selected locations in **A*** contain weights corresponding to synaptic strength between parallel fibers and dendrites on a cerebellar Purkinje cell. The computed output is the sum of the weights stored in the **A*** address set. In this example, the output is a drive signal $\mathbf{p}_j$ to the $j$-th joint actuator controller. The output signal may define a desired position, velocity, or force of the $j$-th joint actuator depending on the command.

of the selected memory locations. Feedback input from sensors causes the behavior to evolve along a goal-directed stimulus-response chain. As feedback changes, the output changes. This, in turn, causes the sensory feedback to change further. The result is a closed servo loop that makes the behavior reactive. The input command from above effectively selects a region of memory wherein the proper set of stimulus-response pairs to generate the desired behavior can be stored.

For each command, a string of stimuli produces a string of responses. For a different command, the same stimuli may produce a different string of responses. The result is that a CMAC module decomposes each different input command into a different string of subcommands. Each different string of subcommands corresponds to a different behavior. Thus, whenever the command changes, a different behavior is generated. For example, a command such as <pick up object X> will select a region of memory where motor behavior appropriate for picking up an object is stored. A different command such as <scratch itch at skin location Y> will select a different region of memory where scratching behavior is stored.

CMAC learns correct output responses for various input conditions by modifying the contents of the selected memory locations. For each input, the learning process computes the difference between the CMAC output and the desired output (provided by a teacher) as shown in Fig. 1. This difference determines a correction factor that is added to the contents of each of the selected memory locations. The rate of learning depends on a gain factor that determines the magnitude of the correction factor.

The $\mathbf{S} \Rightarrow \mathbf{A}^*$ mapping employed by CMAC has the property that any two input vectors that are similar (i.e., close together in input space) will select a highly overlapping subset of locations in the $\mathbf{A}^*$ set. Thus, the output response of a CMAC to similar input vectors will tend to be similar because of many memory locations in common. Hence, CMAC tends to generalize (i.e., to produce similar outputs for similar inputs.) The amount of generalization depends on the number of overlapping memory locations in the $\mathbf{A}^*$ set.

On the other hand, any two input vectors that are dissimilar (i.e., far apart in input space) will select a highly disjoint subset of locations in the $\mathbf{A}^*$ set. Most often, there will be no common memory locations in $\mathbf{A}^*$. Thus, the output response of a CMAC to dissimilar input vectors can be independent, making it easy for CMAC to distinguish between dissimilar input vectors. This means that CMAC can classify or recognize input patterns. The sparse nature of the $\mathbf{A}$ vector makes it possible for CMAC to learn to classify a large number of patterns.

The ability to generalize between similar inputs enables CMAC to quickly learn smooth mathematical functions such as are typical of control system operators in a memory of reasonable size. The $\mathbf{S} \Rightarrow \mathbf{A}^*$ mapping and the resulting property of generalization also gives CMAC many of the properties of a fuzzy controller.

To deal with the complexity of motor behavior required to perform tasks in the natural world without either side-stepping the computational difficulties (as with teleoperated systems) or ignoring most of the relevant variables (as with conventional automation) it is necessary to partition the control problem into manageable subproblems. This may be accomplished through a hierarchical organization such as is typical in military, government, and business organizations. In a hierarchical control structure, each agent at each level of the hierarchy takes direction from a supervisor in the level above and issues directives to subordinates in the level below. Each level has its own set of knowledge, skills, and abilities. Each agent performs task decomposition by decomposing input commands from above into output commands to one or more agents at the next lower level. Agents within the same organizational unit often work together to accomplish tasks that require cooperative behavior.

CMAC modules are specifically designed to be integrated into a hierarchy wherein tasks are decomposed into subtasks at each level. At each level, the input $\mathbf{S}$ is a combination of a command vector from a higher level plus a feedback vector from sensors or from a world model or peer agents. The output $\mathbf{P}$ is a command to a lower level module, a next state, and possibly a message to a peer.

At each level, task commands are combined with feedback to select an appropriate subtask command to be issued to the next lower level. When command input changes, the transfer function of the CMAC can change. This enables higher level commands to select among a library of transfer functions, each of which can generate strings of task commands to the next lower level. By this process, a hierarchy of CMAC modules can decompose a complex high-level task into a string of drive signals for individual actuators as shown in Fig. 2. A hierarchy of similar building blocks in the brain can be used to implement a hierarchy of behaviors such as observed by Tinbergen [3] and others in simple creatures such as fish, birds, and insects.

CMAC was based on a neurophysiological model of the cerebellum published by Albus in 1971 and 1972 [6, 7]. This model was, in turn, inspired by a series of experiments performed by Eccles [8] and others during the 1960s that showed a striking resemblance between the structure and function of the cerebellum and the Perceptron neural net developed by Rosenblatt [9] in

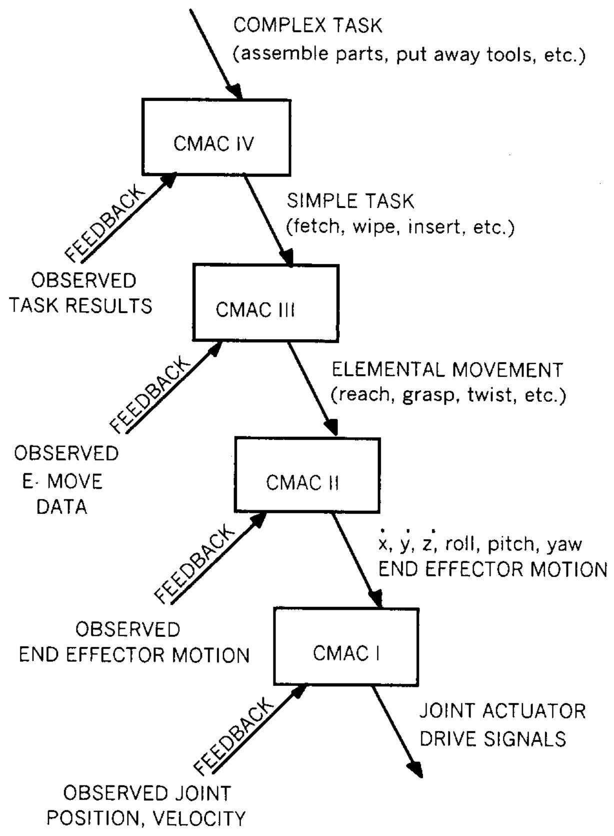

COMPLEX TASK
(assemble parts, put away tools, etc.)
CMAC IV
FEEDBACK
SIMPLE TASK
(fetch, wipe, insert, etc.)
OBSERVED TASK RESULTS
CMAC III
FEEDBACK
ELEMENTAL MOVEMENT
(reach, grasp, twist, etc.)
OBSERVED E- MOVE DATA
CMAC II
FEEDBACK
ẋ, ẏ, ż, roll, pitch, yaw
END EFFECTOR MOTION
OBSERVED END EFFECTOR MOTION
CMAC I
FEEDBACK
JOINT ACTUATOR DRIVE SIGNALS
OBSERVED JOINT POSITION, VELOCITY

**Fig. 2.** A hierarchy of CMAC controllers. In response to each input command from a higher level, each CMAC generates a string of output commands to a lower level. These output commands are stimulus-response behaviors that are selected by input commands and driven by feedback variables. This hierarchy of CMACs can be used to partition a manipulator control problem into a manageable set of sub-problems.

the 1950s. The Albus model, combined with a similar model published in 1969 by David Marr [10], have become known as the Marr-Albus model of the cerebellum. The Marr-Albus model has had a profound influence on brain research in the cerebellum and remains one of the most widely accepted and frequently cited models of the cerebellum today.

The short-term impact of CMAC research was modest. CMAC was awarded an IR-100 award and a Department of Commerce Silver Medal, but had little immediate effect on the field of neural nets. At the time of its publication, the field of neural networks had fallen into a state of disrepute. Minsky and Papert had recently published their influential book entitled *Perceptrons* in which they demonstrated some of the theoretical limitations of neural networks [11]. Over-reaction to the negative conclusions of the Minsky-Papert critique had a profound impact on neural net researchers. Virtually all interest in (and funding support for) neural network research evaporated for almost a decade. Serious interest in neural network research did not recover until the late 1980s with the invention of back propagation, adaptive resonance theory, Hopfield nets, and a number of other new discoveries.

Since 1990, CMAC has began to attract increasing levels of interest. CMAC has been used for computing plant models for feedforward control on-line in real-time. Miller has shown that CMAC can be used to implement a real-time adaptive controller for various robotic applications [12]. Miller and others have shown that CMAC learning typically converges at least an order of magnitude faster than back propagation. As the properties of CMAC have become apparent, more and more researchers have begun to apply CMAC to their areas of interest. CMAC has become well known by students and researchers in adaptive learning mechanisms and is the subject of neural network studies in a number of laboratories in several countries. The CMAC papers have been cited in the literature over 240 times.

At least as important as its contribution to the field of neural nets, is the influence CMAC has had in providing the conceptual foundation for the Real-time Control System architecture known as RCS. The fact that CMAC could learn a set of transfer functions and transition matrices meant that a CMAC could be designed to implement any state-table or expert system rule base. The CMAC input vector (command, feedback, and state) corresponds to the IF predicate. The output vector corresponds to the THEN consequent and next state. At each compute cycle, the CMAC input vector is compared with the lines on the left side of a state table. For the matching line, the right side of the state table provides the output subcommand and next state. Thus, any CMAC can be emulated by a finite state machine. RCS is a control system built from a hierarchy of finite state machine modules.

RCS was initially developed by Barbara et al. [4] to implement controllers for sensory-interactive robots. RCS enabled NBS robots to use visual feedback to acquire randomly oriented objects and pursue moving targets. During the early 1980s, RCS evolved into the hierarchical shop control system architecture for the Automated Manufacturing Research Facility (AMRF) [5]. In the AMRF, RCS was implemented on the Horizontal Machining Workstation, the Cleaning and Deburring Workstation, the Material Handling Workstation, and the Advanced Deburring and Chamfering System. During the late 1980s, RCS was developed into a control system for the DARPA Multiple Undersea Autonomous Vehicle program and was adopted by NASA for the Space Station Telerobotic Servicer. RCS was adopted by the U.S. Bureau of Mines as a control system for automated mining operations. At Martin-Marietta, Barbera and Fitzgerald developed RCS for the Army TMAP unmanned vehicle. Later at Advanced Technology Research Corporation, Barbera and Fitzgerald used RCS to build an Automated Stamp Distribution Center and to design a General Mail Handling Facility for the U.S. Postal Service. Commercial versions of RCS are currently being used for controlling machine tools and laser cutting machines. RCS was adapted by General Dynamics Electric Boat as a control system for the next generation nuclear submarine. At NIST, RCS has been used for the Enhanced Machine Controller, the Next Generation Inspection System, the Automated Welding Manufacturing System, the RoboCrane, a computer controlled Man-Lift, and the NIST Unmanned Ground Vehicle [13]. Most recently, RCS has been adopted by the Army for the Demo III Experimental Unmanned Vehicle program [14]. RCS is currently under consideration for the Army's Future Combat Systems program.

By the end of the century, RCS evolved into a canonical reference model architecture with sufficient structure to enable the development of metrics and interface standards for intelligent systems. Elements of RCS have been incorporated into standards activities in both civilian and military sectors. RCS is currently influencing the development of interface standards for machine tools under the Open Modular Architecture Controller (OMAC) group, and for military vehicles under the Joint Architecture for Unmanned Ground Systems Standards (JAUGS) project [15, 16].

For his work on CMAC and RCS applied to the AMRF, Albus was awarded the Department of Commerce Gold Medal, the NIST Applied Research Award, the Japanese Industrial Robot Association Research and Development Award, and the Robot Industries Association Joseph F. Engelberger Award. For the past ten years, Albus has been working on a theoretical foundation for the engineering of intelligent systems [17]. He is currently a Senior NIST Fellow in the Intelligent Systems Division of the Manufacturing Engineering Laboratory.

*Prepared by James S. Albus.*

## Bibliography

[1] J. S. Albus, A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC), *J. Dyn. Syst. Meas. Control, Trans. ASME* **97**, 220-227 (1975).

[2] J. S. Albus, Data Storage in the Cerebellar Model Articulation Controller (CMAC), *J. Dyn. Syst. Meas. Control, Trans. ASME* **97**, 228-233 (1975).

[3] Niko Tinbergen, *The Study of Instinct,* Clarendon Press, Oxford (1951).

[4] Anthony J. Barbera, M. L. Fitzgerald, James S. Albus, and Leonard S. Haynes, RCS: The NBS Real-Time Control System, in *Robots 8: Conference Proceedings, Vol. 2,* Detroit, Michigan, June 1984, Society of Manufacturing Engineers, Dearborn, MI (1984) pp. 19-1–19-33.

[5] J. A. Simpson, R. J. Hocken, and J. S. Albus, The Automated Manufacturing Research Facility of the National Bureau of Standards, *J. Manuf. Syst.* **1**, 17-31 (1982).

[6] J. S. Albus, A Theory of Cerebellar Function, *Math. Biosci.* **10**, 25-61 (1971).

[7] James Sacra Albus, *Theoretical and Experimental Aspects of a Cerebellar Model,* Ph.D. Thesis, University of Maryland, College Park, MD (1972).

[8] John C. Eccles, Masao Ito, and János Szentágothai, *The Cerebellum as a Neuronal Machine,* Springer-Verlag, New York (1967).

[9] F. Rosenblatt, The Perceptron: A probabilistic model for information storage and organization in the brain, *Psychol. Rev.* **65**, 386-408 (1958).

[10] D. Marr, A Theory of Cerebellar Cortex, *J. Physiol. (London)* **202**, 437-470 (1969).

[11] Marvin Minsky and Seymour Papert, *Perceptrons: An Introduction to Computational Geometry,* MIT Press, Cambridge, MA (1969).

[12] W. Thomas Miller, III, Sensor-based control of robotic manipulators using a general learning algorithm, *IEEE J. Robot. Autom.* **RA-3**, 157-165 (1987).

[13] J. S. Albus, The NIST Real-time Control System (RCS): An Application Survey, in *Proceedings of the AAAI 1995 Spring Symposium Series,* Stanford University, Stanford, CA, March 27-29, 1995.

[14] J. S. Albus, 4-D/RCS Reference Model Architecture for Unmanned Ground Vehicles, in *Proceedings IEEE International Conference on Robotics and Automation,* San Francisco, April 22-28, 2000, Institute of Electrical and Electronics Engineers, New York (2000) pp. 3260-3265.

[15] Homepage of the OMAC Users Group (http://www.arcweb.com/omac), OMAC Users Group (2000).

[16] Joint Architecture for Unmanned Ground Systems (JAUGS) (http://www.jointrobotics.com/Jaugs), U.S. Department of Defense.

[17] James S. Albus, Outline for a Theory of Intelligence, *IEEE Trans. Syst. Man Cybern.* **21**, 473-509 (1991).