CMAC 제어기를 위한 효과적인 메모리 매핑 함수

°권 호열*, 변 증남*, 서 일홍**
*한국과학기술원 전기및전자공학과, **한양대학교 전자공학과

An Effective Memory Mapping Function for CMAC Controller

H. Y. Kwon*, Z. Bien*, I. H. Suh**
*KAIST, **Hanyang Univ.

Abstracts

In this paper, the structure of CMAC address mapping is first revisited, and the address hashing function and the random mapping is discussed in the conventional CMAC implementation. Then the effective size of CMAC memory is derived from the modulus property of the CMAC address vector, and a new hashing function for the effective memory mapping is proposed for a CMAC implementation with feasible memory size and no troublesome random mapping. Finally, the performance of the conventional CMAC learning algorithm and that of the proposed new CMAC scheme are compared via simulations.

1. Introduction

As a means of controlling complex systems such as articulated robot manipulators, Albus[1-3] proposed the Cerebellar Model Articulation Controller(CMAC) based on the principle of cerebellar mechanism so that the overall control system behaves as desired though no a priori knowledge for the system dynamics is utilized. The CMAC is designed to accept system variables in the form of an input vector and give a response such that two similar input vectors give similar response, and through this generalization process it is capable of learning even for nonrepetitive tasks as tested in some experimental results[1-7]. Albus applied the CMAC to control a seven degree of freedom master-slave arm[1], and to learn simple test functions[2][3]. And the CMAC based control has been studied for a two degree of freedom biped walking device by Camana[4], for a position and velocity servoing of the manipulator with visual feedback[5], for a two degree of freedom articulated robot arm[6], and for five-axis industrial robot[7] by Miller and his group. Also, the CMAC was used to learn some two dimensional pattern groups for shape recognition by Miller[8].

Although the CMAC was applied to various applications as a learning control scheme, the detail analysis of the CMAC mapping and its learning capability were partially performed yet. For CMAC learning rule, the algebraic and the sequential learining rules were proposed by Shamma[9] and Hwang[10][11], respectively. Miller[6] presented some simulation results to aid the design of the CMAC parameters such as the size of CMAC associative memory A ($|A|$), the number of CMAC memory locations(C) addressed by each input state, and learning gain β. Among the CMAC parameters, particularly $|A|$ has set a difficulty in the CMAC implementation since it seems to be required for $|A|$ an impractical huge memory space in the order of the total number of discrete input sensor vectors. And, there has not been any guidelines for selecting the physical size of the CMAC memory in conventional CMAC.

In this paper, the structure of CMAC address mapping is first revisited, and the address hashing function and the effect of random mapping is discussed in the conventional CMAC implementation. Then the effective size of CMAC memory is derived using the modulus property of the CMAC address vector, and a new hashing function for the effective memory mapping is proposed for a CMAC implementation with feasible memory size and no troublesome random mapping. Finally, the performance of the conventional CMAC learning algorithm and that of the proposed new CMAC scheme are compared via simulations.

2. The CMAC revisited

In terms of modified notations the basic CMAC algorithm of Albus[2][3] will be revisited and some of difficulties of the theory will be discussed. To describe the structure of CMAC mapping, let f be the complex, multi-dimensional nonlinear function of the form :

$$p = f(s). \tag{1}$$

where $s = (s_1, s_2, \ldots, s_N)^T$ is the sensor input vector, and $p = (p_1, p_2, \ldots, p_L)^T$ is the response vector. Here superscript T denotes the transpose. The CMAC computational scheme for converting the input vector s into the response vector p shown in Fig. 1, where M and A denote the intermediate mapping matrix and the CMAC memory, respectively.

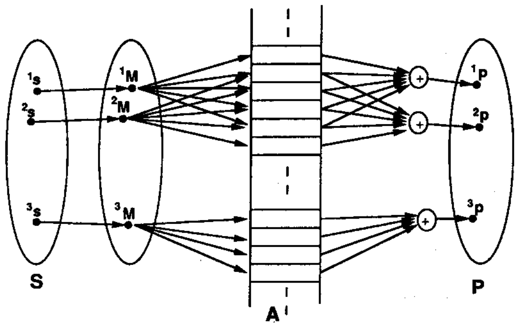


Fig. 1 The basic CMAC mapping scheme.

Let's define some notations. Given two positive integers n and C, let $\mathrm{Mod}(n, C)$ denote the remainder of n when divided by C. Also, given a row vector $g = (g_1, g_2, \ldots, g_C)$, let $\mathrm{Rot\_R}(g, n)$ be the vector obtained by rotating n times the elements of the vector g by right-shifting, e.g., $\mathrm{Rot\_R}(g, 2) = (g_{C-1}, g_C, g_1, \ldots, g_{C-2})$, etc.

Suppose that each component s_i, $1 \le i \le N$, of the input vector s takes a quantized discrete value among R_i levels. Then, without loss of generality, it can be assumed that s_i takes an integer, i.e., $0 \le s_i \le R_i - 1$. Also, suppose that a given positive integer C means the number of memory cell being associated with an input vector. Then the basic CMAC computational scheme[2] for converting the input vector s into the response vector p can be interpreted as follows :

Step 1. ($s \to M$) For an input vector s, each R_i-ary variable s_i is first mapped into a row vector m_i whose elements m_i^j, $j = 1, 2, \ldots, C$, have the value of C contiguous integers from s_i to $s_i + C - 1$, and which is rearranged in such a way that

$$\mathrm{Mod}\,(m_i^j, C) = j - 1, \qquad 1 \le j \le C. \tag{2}$$

Thus

$$\begin{aligned} m_i &= (m_i^1, m_i^2, \ldots, m_i^C) \\ &= \mathrm{Rot\_R}((s_i, s_i+1, \ldots, s_i+C-1), \mathrm{Mod}(s_i, C)). \end{aligned} \tag{3}$$

Thus, for each N-dimensional input vector $s = (s_1, s_2, \cdots, s_N)^T$, there corresponds the $N \times C$ matrix M as

$$M = \begin{bmatrix} m_1 \\ m_2 \\ \cdots \\ m_N \end{bmatrix} = \begin{bmatrix} m_1^1 & m_1^2 & \cdots & m_1^C \\ m_2^1 & m_2^2 & \cdots & m_2^C \\ & \cdots\cdots & & \\ m_N^1 & m_N^2 & \cdots & m_N^C \end{bmatrix}. \tag{4}$$

Step 2. ($M \to A$) For each $j = 1, 2, \ldots, C$, define the j-th address vector μ_j for the input s take

$$\mu_j = (m_1^j \;\; m_2^j \;\; \cdots \;\; m_N^j)^T \qquad 1 \le j \le C. \tag{5}$$

Then we may write the intermediate matrix M for input s as

$$M = (\mu_1, \mu_2, \ldots, \mu_C). \tag{6}$$

Let A^* denote the vector of physical addresses called by the input vector s, and A^* is a subset of the CMAC memory, A. And let $h(\mu_j)$ be a hashing function[12] using the key record of the address vector μ_j to give a physical address for the j-th element in A^*. Then A^* can be derived as :

$$A^* = (h(\mu_1) \;\; h(\mu_2) \;\; \cdots \;\; h(\mu_C)). \tag{7}$$

Step 3. ($A \to p$) Finally, the CMAC response p for the input vector s can be formulated as sum of weights in C locations of A^* as follows.

$$p = \sum_{j=1}^{C} w(h(\mu_j)), \tag{8}$$

where $w(h(\mu_j))$ is the L-dimensional vectored data at address of $h(\mu_j)$ in the memory A.

Consider two neighboring input vectors which have some common address vectors μ_j's in Eq. (5). In the CMAC mapping scheme, these overlapping of the vector of addresses in A^* leads to so called generalization as a kind of learning[2], and the number of the overlapping elements varies according to the distance between the two inputs in a norm sense.

3. Apparent Size of CMAC Memory A

Recall that $|A|$ means the number of locations in the CMAC memory A or simply the size of memory A that is required when the CMAC is implemented. From Eq. (3), it can be observed that a component s_i of the vector s is an R_i-ary variable while each component m_i^j of the vector m_i for s_i is an $(R_i + C - 1)$-ary variable. Then, $|A|$ may be determined directly from the range of μ_j in Eq. (5), and in this case we shall denote the size $|A|_a$,

$$|A|_a = \prod_{i=1}^{N} (R_i + C - 1), \tag{9}$$

the subscript a means the apparent size. Note also that a hashing-function $h_a(\mu_j)$ which gives the physical address in $|A|_a$ locations using an address vector μ_j in Eq. (5) as follows :

$$\begin{aligned} h_a(\mu_j) = m_1^j &+ m_2^j \bar{R}_1 + \cdots \\ &+ m_N^j (\bar{R}_1 \bar{R}_2 \cdots \bar{R}_{N-1}). \end{aligned} \tag{10}$$

where $\bar{R}_i = R_i + C - 1$.

It is remarked that $|A|_a$ is in the order of $\bar{R}^N$, and this is an impractically huge size in many real situations. To solve this difficulty of $|A|_a$ in the conventional CMAC[1-8], such a memory A is taken to be a hypothetical memory, and another random hash method is additionally used to map from the large memory A into a small memory A_p with practical size assuming that all possible states of s will not be encountered in solving a particular control problem.

However, this secondary random mapping has been introduced to avoid the curse of dimensionality of $|A|_a$ without consideration of its effects on inherent learning capability of CMAC, i.e., generalization. In fact, there are finite probability of mapping collisions which not only create undesirable interferences between distant input vectors, but also may deteriorate the learning speed to compensate the random mapping-interferences in data storage in the CMAC memory A_p[2]. Furthermore, the assumption of special task can be no longer valid when the task of the manipulator is frequently altered, and it sets a severe restiction on general of CMAC applications.

On the other hand, Albus[2] thought that the CMAC has an storage reduction by distributed memory look-up technique. He thought about the minimal $|A_p|$ which makes $s \to A$ mapping be unique for each input vector. Then it should be satisfied that the number of ways to take C locations from a total of $|A_p|$ is greater than R^N, i.e.,

$$\binom{|A_p|}{C} = \frac{|A_p|!}{C!(|A_p| - C)!} \ge R^N. \tag{11}$$

Normalizing $|A_p|$ by R after some algebraic manipulations with Eq. (11), the minimum requirement of $|A_p|/R$ can be obtained as

$$\frac{|A_p|}{R} = \frac{C}{R}\left[R^{N/C} + 1\right] \tag{12}$$

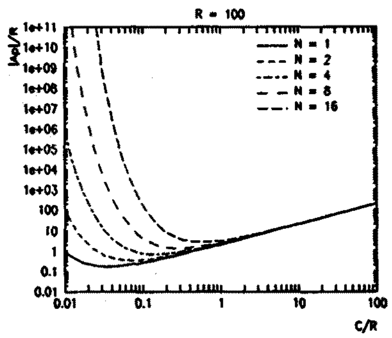


Fig. 2 Relation of the C and $|A_p|$ by Albus[2].

Fig. 2 shows that $|A_p|$ can be reduced as C increases to the value of R when $R = 100$.

However, this memory reduction model makes an under-estimation of $|A_p|$ because C locations for an input vector s can not be arbitrary chosen in total $|A|$ but rather through the highly structured $s \to A$ mapping. In the next section, the size of effective CMAC memory will be derived using the structured CMAC mapping.

4. Modulus Property of the CMAC Address Vector and Association Block

Let us consider the case when the input is two dimensional vector $s = (s_1, s_2)^T$. Let $\{m_i\}$, $1 \le i \le 2$ denote a set of elements of m_i, $1 \le i \le 2$, in Eq. (3). Then the CMAC address space for μ_j in Eq. (5) can be represented by $\{m_1\}\times\{m_2\}$. Then, the range of the elements of the address vector μ_j called by an input s constructs an association block $B_C(s)$ which defined by a two dimensional $C\times C$ region of $(m_1^1, m_1^2, ..., m_1^C) \times (m_2^1, m_2^2, ..., m_2^C)$ which is equivalent with $(s_1, s_1+1, ..., s_1+C-1) \times (s_2, s_2+1, ..., s_2+C-1)$ in CMAC address space from Eq. (3). For example, when $N = 2$, $R_1 = 11$, $R_2 = 10$, and $C = 4$, Fig. 3 shows the association blocks for inputs ${}^1s = (5, 5)^T$, ${}^2s = (6, 4)^T$, and ${}^3s = (3, 6)^T$, respectively.

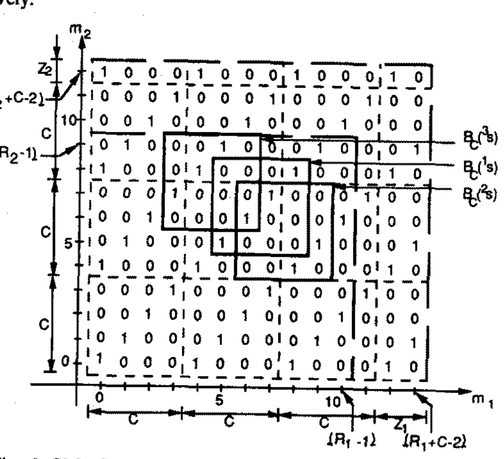


Fig. 3 CMAC address space and the association blocks.

It is remarked that the association block $B_C(s)$ contains only C effective address vectors among the $C\times C$ possible address vectors because of the constraints of the same modulus for each component of the CMAC address vector in Eq. (2). Fig. 3 shows the map of effective address vectors where the effective address denotes as "1" and non-effective address denotes as "0". Hence, the two dimensional association block $B_C(s)$ for an input s can be derived as a $C\times C$ matrix and its $k\times k$ submatrix $\tilde{I}_{k(s)}$, $0 \le k(s) < C$, as follows :

$$B_C(s) = \begin{bmatrix} \tilde{I}_{k(s)} & 0 \\ & \\ 0 & \tilde{I}_{C-k(s)} \end{bmatrix}, \tag{13.1}$$

where

$$\tilde{I}_{k(s)} = \begin{bmatrix} 0 & \cdots & 0 & 0 & 1 \\ 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 1 & 0 & 0 \\ & \cdots & & & \\ 1 & \cdots & 0 & 0 & 0 \end{bmatrix}, \tag{13.2}$$

and $k(s)$ denotes Mod (s_2-s_1, C).

For more than two dimensional input vector s, although the map of the effective address vector is more complex, the association block $B_C(s)$ for an N-dimensional input vector s is extended to N-dimensional volme of C^N elements in the CMAC address space $\{m_1\}\times\{m_2\}\times \cdots \times\{m_N\}$ while the number of effective locations is only C in it under the modulus constraints in Eq. (2), similarly. Consequently, it can be observed that the effective number of the CMAC addresses in associaton memory A decreases as the dimensionality N of s increases or C increases.

5. Effective Size of CMAC Memory A

How much reduced the CMAC address space through the consideration of effective addresses ? Let $|A|_e$ denote the effective size of locations in associative memory A considering the modulus constraint in Eq. (2). Here subscript e denotes the effective memory size. Recall that m_i^j is an $\tilde{R}_i$-ary variable in Eq. (3). Also, let Q_i and Z_i denote the quotient and remainder, respectively, of $\tilde{R}_i$ by division of C. Suppose a sub-block of an N dimensional association block can be characterized with N lengths of Z_i, $1 \le i \le N$, and $0 \le Z_i \le C$. And define a function $\lambda(Z_1, Z_2, ..., Z_N)$ be the minimum value of its elements Z_i, $1 \le i \le N$, corresponding to the number of effective elements in the block under the constraint of same modulus for each Z_i, $1 \le i \le N$, from Eq. (2) as follows :

$$\lambda(Z_1, Z_2, ..., Z_N) = \text{Min}\,(Z_1, Z_2, ..., Z_N), \quad 0 \le Z_i < C, 1 \le i \le N. \tag{14}$$

Then $|A|_e$ can be obtained by

$$|A|_{e,N=1} = Q_1\lambda(C) + \lambda(Z_1) = Q_1C + Z_1, \tag{15.1}$$

$$\begin{aligned} |A|_{e,N=2} &= Q_1Q_2\lambda(C, C) + Q_1\lambda(C, Z_2) \\ &\quad + Q_2\lambda(Z_1, C) + \lambda(Z_1, Z_2) \\ &= Q_1Q_2C + Q_1Z_2 + Q_2Z_1 \\ &\quad + \text{Min}\,(Z_1, Z_2), \end{aligned} \tag{15.2}$$

and so on.

Since the expansions of the combination such as in Eq. (15.1)-(15.2) are very complex process, it is hard to analysis the properties of them from their parameters N and C as well as each Q_i and Z_i for $1 \le i \le N$. Hence to simplify our argument, suppose that all elements s_i, $1 \le i \le N$, of the input vector s has the same resolution R so that $\bar{R}_i = R_i + C - 1$ and $\bar{R} = QC + Z$. Then the total number of effective locations in memory A can be derived from Eq. (15.1)-(15.2) as

$$|A|_{e,N=1} = QC + Z, \tag{16.1}$$

$$|A|_{e,N=2} = Q^2C + 2QZ + Z, \tag{16.2}$$

and so on. Hence, the effective size of associative memory $|A|_e$ can be obtained by mathematical induction as follows :

$$|A|_e = Q^N C + \sum_{i=0}^{N-1} \binom{N}{i} Q^i Z = Q^N C + \left[(Q+1)^N - Q^N\right] Z. \tag{17}$$

By normalizing $|A|_e$ and C with respect to R,

$$\frac{|A|_e}{R} = \left[(Q+1)^N \frac{Z}{C} + Q^N\left(1 - \frac{Z}{C}\right)\right] \frac{C}{R}$$

$$for \; Q \ge 1, \; 0 \le \frac{Z}{C} < 1. \tag{18}$$

Consider a numerical example the CMAC memory reduction when $N = 10$, $R = 100$, and $C = 80$. Then the apparent size of the CMAC memory is calculated as $|A|_a = 3.38 \times 10^{22}$ from Eq. (9) while the effective CMAC memory requires only $|A|_e = 1.18 \times 10^6$ by Eq. (18). Comparing our effective memory size with the Albus's $|A|_p = 222$ by Eq. (12) which is insufficient for unique $s \to A$ mapping.

Assuming that $R \gg 1$ and $C \gg 1$ and after some algebraic manipulations, it can be obtained a more useful approximation $|A|_{\tilde{e}}$ for $|A|_e$ using R and C instead of the Q, Z, and C.

$$\frac{|A|_{\tilde{e}}}{R} = \begin{cases} \left(\dfrac{1 + C/R}{C/R}\right)^N \dfrac{C}{R} & if \; \dfrac{C}{R} \le 1 \\ 2^N + \dfrac{C}{R} - 1 & if \; \dfrac{C}{R} \ge 1 \end{cases} \tag{19}$$

Then the error between the exact and the approximation of $|A|_e/R$ can be derived from Eq. (18) and Eq. (19)

$$ERR_{|A|_e/R} = |A|_e/R - |A|_{\tilde{e}}/R = \left[(Q+1)^N - Q^N\right] \frac{Z}{C} \tag{20}$$

Fig. 4 shows the size reduction of effective CMAC mamory according to the change of C when N and R are given. Comparing with Fig. 2, note that the approximation of $|A|_p$ by Albus[2] is an under-estimated. Also, for $N \ge 2$, the minimum memory realization of CMAC is occured at $C/R = 1$. Also, the CMAC with $C/R > 1$ can be thought as a kind of Perceptron[2] because of the generalization for an input vector influences almost of all memory cells, and where memory reduction is no longer valid.

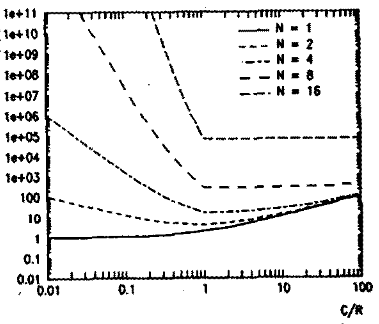


Fig. 4 Relation of the C and the $|A|_e$.

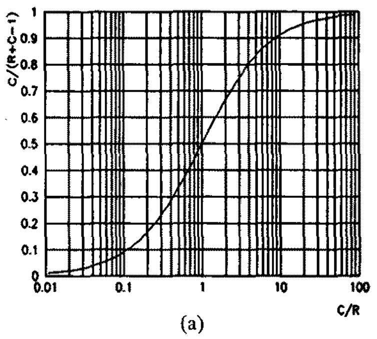
Also the ratio of the number of addresses called by an input s over total effective address space can be obtained by

$$\frac{C}{|A|_{e,1D}} = \frac{C/R}{1 + C/R} \tag{21}$$

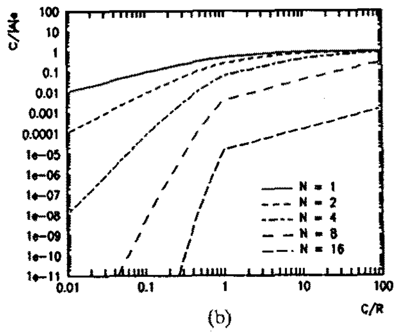
for one-dimensional case, and

$$\frac{C}{|A|_{e,ND}} = \frac{C}{|A|_e} = \left[\frac{1}{|A|_e/R}\right] C/R \tag{22}$$

for N dimensional case. Fig. 5 shows $C/|A|_{e,1D}$ and $C/|A|_{e,ND}$ vs. C/R characteristics. These two variables inversely represent the number of independent input vectors required in complete learning for given CMAC memory.



(a)



(b)

Fig. 5 CMAC memory occupation by an input s (a) 1-Dim. case (b) N-Dim. case.

Finally, consider a cost function for the CMAC implementation as

$$J = k_1 \left(\frac{|A|_e}{R}\right)^{1/N} + k_2 \frac{C}{R} \tag{23}$$

where the first term represents a CMAC memory requirement while the second represents for CMAC caculation time, and k_1 and k_2 are appropriate weighting factors. Fig. 6 shows the cost function when both k_1, k_2 are unity.

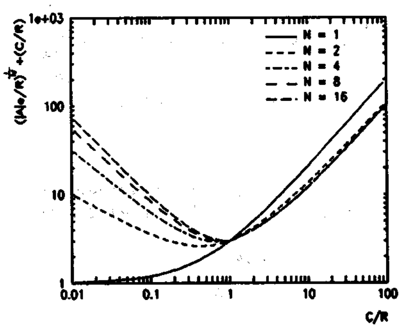


Fig. 6 A CMAC cost function for the $|A|_e$ and the C.

6. Design of Effective Memory Mapping Function

For many real environment, the CMAC parameters of N and R are often inherently fixed according to its tasks. Selecting the value of C, $|A|_e$ is simply determined in Eq. (18), but generally the The address mapping from the effective adress vector to one dimensional associative memory A requires very complex hashing function since the CMAC address space, as in Fig. 3, consists of not only regular C^N volume but also irregular volumes from various nonzero Z_i's in Eq. (18). Hence practical realization of CMAC with simple hashing function, we proposed an augmented effective associative memory $|A|_{e^+}$ defined as

$$|A|_{e^+} = C \prod_{i=1}^{N} (Q_i + 1) \geq |A|_e \tag{24}$$

Then the new hashing function $h_{e^+}(\mu_j)$ for the CMAC address vector μ_j into $|A|_{e^+}$ locations can be given by

$$h_{e^+}(\mu_j) = (q_1 + q_2 Q_1 + \cdots + q_N Q_1 Q_2 \cdots Q_{N-1}) \cdot C + z_1 \tag{25}$$

where q_i and z_i are the quotient and the remainder of m_i^j/C, $1 \leq i \leq N$, respectively. It is also noted that all z_i's, $1 \leq i \leq N$, are the same since m_i^j, $1 \leq i \leq C$, has the same modulus from Eq. (2).

The error between $|A|_e$ and $|A|_{e^+}$ are obtained as

$$\begin{aligned} ERR_{e^+} &\equiv |A|_{e^+} - |A|_e \\ &= [(Q+1)^N C - Q^N](C - Z) \end{aligned} \tag{26}$$

From above, the wasted memory space ERR_{e^+} in $|A|_{e^+}$ can be reduced by setting parameter C such that Q increases or Z approach to C. Also, note that $|A|_{e^+}$ is consistent with exact value of $|A|_e$ when $Z = C$ in Eq. (26).

7. Numerical Examples

For a simulation study, two dimensional sinusoidal test functions[3] are chosen as $\hat{p}$ is chosen as

$$\hat{p} = \sin\left(\frac{2\pi s_1}{360}\right) \sin\left(\frac{2\pi s_2}{360}\right), \quad 1 \leq s_1 \leq 360,\ 1 \leq s_2 \leq 180, \tag{27}$$

and it is shown in Fig. 7.

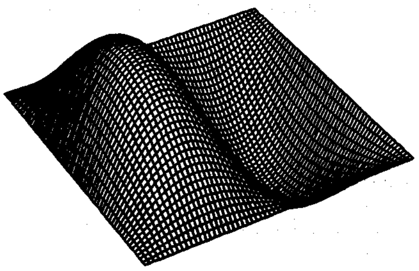


Fig. 7 2-Dim. test function.

The physical memory space is 2910 locations, the size of association block is $C = 32$, learning gain β is chosen as unity for one shot learning, and the learning strategy of maximum error correction is used.

Fig. 8 presented the CMAC learning with conventional CMAC memory mapping where $|A|_a$ is calculated as 82501 locations from Eq. (9), but physical memory space is used only 3.53 % of $|A|_a$. In this case, the hashing-function $h_a(\mu_j)$ of Eq. (10), and additional random mapping are used for the CMAC address generation. The reconstruction of the desired response function are shown in Fig. 8a-8d as the number of stored data increases with 1, 2, 16, and 50, respectively.

Fig. 9 presented the CMAC learning with proposed effective memory mapping scheme where $|A|_{e^+}$ is used as 2910 locations from Eq. (24) and the hashing-function $h_{e^+}(\mu_j)$ of Eq. (25). The reconstruction of the desired response function are shown in Fig. 9a-9d as the number of stored data increases with 1, 2, 16, and 50, respectively.

8. Conclusion

In this paper, the memory mapping function of the conventional CMAC is analyzed. Then the effective size of CMAC memory is derived from the modulus property of the CMAC address vector, and a new hashing function for the effective CMAC memory mapping is proposed with feasible memory size and no troublesome random mapping. Finally, the performance of the conventional memory mapping scheme and that of the proposed new memory mapping scheme are compared via simulations of CMAC learning for a two dimensional test function. It is remarked that our proposed scheme is stable and fast convergent without problem of random mapping collisions, and where a feasible small memory space is required.

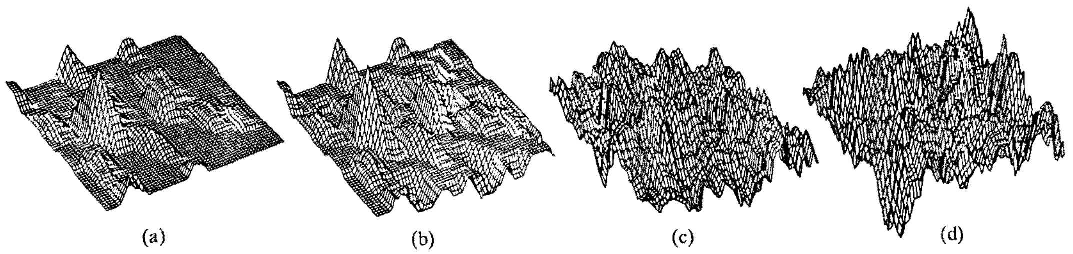


Fig. 8 CMAC learning with the conventional memory mapping scheme.

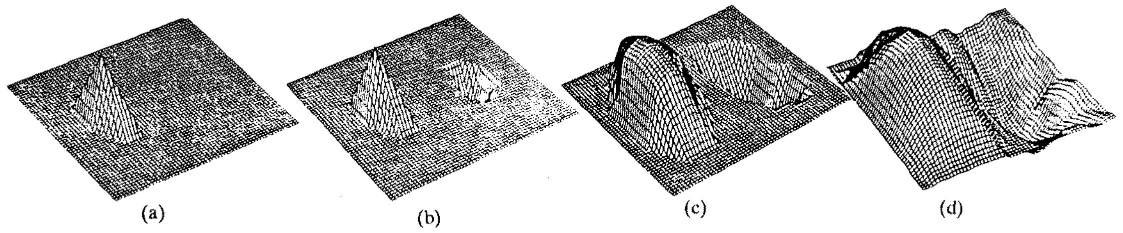


Fig. 9 CMAC learning with the effective memory mapping scheme.

References

[1] J. S. Albus, "Theoretical and experimental aspects of a cerebellar model", Ph. D. Dissertation, Univ. of Maryland, 1972.

[2] J. S. Albus, "A new approach to manipulator control : the cerebellar model articulation controller(CMAC)," *Trans. ASME J. Dynamic Syst., Meas., Contr.*, vol. 97, no. 3, pp. 220 - 227, 1975.

[3] J. S. Albus, "Data storage in the cerebellar model articulation controller(CMAC)," *Trans. ASME J. Dynamic Syst., Meas., Contr.*, vol. 97, no. 3, pp. 228 - 233, 1975.

[4] P. C. Camana, "A Study of Physiologically Motivated Mathematical Models for Human Postura Control," Ph.D. Thesis, Dep. of Electrical Engineering, Ohio State Univ., 1977.

[5] W. T. Miller III, "Sensor-based control of robotic manipulators using general learning algorithm," *IEEE J. Robotics & Autom.*, vol. RA-3, no. 2, pp. 157 - 165, 1987.

[6] W. T. Miller III, F. H. Glanz, and L. G. Kraft, "Application of a general algorithm to the control of robotic manipulators," *Int. J. Robot. Res.*, vol. 2, pp. 84 - 98, 1987.

[7] R. P. Hewes and W. T. Miller III, "Practical Demonstration of a Learning Control System for a Five-axis Industrial Robot," *Intelligent Robot and Computer Vision : Seventh in a Series*, Editor : D. P. Casasent, Proc. SPIE 1002, pp. 679 - 685, 1988.

[8] F. H. Glanz, and W. T. Miller III, "Shape recognition using a CMAC based learning system," *Intelligent Robots and Computer Vision: Sixth in a Series*, Proc. SPIE Vol.848, pp. 294 - 298, 1987.

[9] J. S. Shamma, "A Method for Inverse Robot Calibration," Appendix B., Evaluation of CMAC, M.S. Thesis, Dept. of Mechanical Engineering, M.I.T., 1985.

[10] H. Hwang and D. Y. Choi, "On learning of CMAC for Manipulator Control," *Proc. of '89 Korea Autom. Contr. Conf.*, pp. 653-662, Seoul, Korea, 1989.

[11] H. Hwang and D. Y. Choi, "Learning Performance and Desing of an Adaptive Control Function Generator: CMAC," *Proc. of '89 Korea Autom. Contr. Conf.*, pp. 675-681, Seoul, Korea, 1989.

[12] W. D. Maurer and T. G. Lewis, "Hash Table Methods," *Computing Surveys*, Vol. 7, No. 1, pp. 5 - 19, 1975.