様式 6

論　文　目　録

報告番号	[✓ 甲 工] 乙 工 工 修 第 26 号	氏　名	何　晨
学位論文題目	Non-Uniform Cellular Neural Network and its Applications		

論文の目次

参考論文

主論文

1. “Iterative middle mapping learning algorithm for cellular neural network”, Chen HE and Akio USHIDA, Institute of Electronics, Information and Communication Engineers Trans., Vol.E-77A, No.4, pp.706-715, 1994.

2. “A modified predictor-corrector tracing curve algorithm for nonlinear resistive circuits”, Chen HE and Akio USHIDA, Institute of Electronics, Information and Communication Engineers Trans., Vol.E-74, No.6, pp.1455-1462, 1991.

副論文

1. “Convergence analysis of synchronous-updating CNN and related DTCNN”, Chen HE and Akio USHIDA, Proc. of 1993 Int. Symp. on Nonlinear Theor. and its Appl., pp.29-34, Hawaii, American, 1993.

2. “Cellular associative memory with middle mapping learning algorithm”, Chen HE and Akio USHIDA, Proc. of 1993 Int. Symp. on Neural Network and Signal Processing, pp.160-165, Guangzhou, China, 1993.

3. “An efficient algorithm for solving nonlinear resistive circuits”, Chen HE and Akio USHIDA, Proc. of 1991 IEEE Inter. Symp. on Circuit and System, pp.2328-2331, Singapore, 1991.

備考

1　論文題目は，用語が英語以外の外国語のときは日本語訳をつけて，外国語，日本語の順に列記すること。

2　参考論文は，論文題目，著者名，公刊の方法及び時期を順に明記すること。

3　参考論文は，博士論文の場合に記載すること。

様式７

論　文　内　容　要　旨

報告番号	[✓ 甲 工] 乙 工 工 修 第 ２６ 号	氏　名	何　　晨
学位論文題目	Non-Uniform Cellular Neural Network and its Applications		

内容要旨

セルラーニューラルネットワーク（ＣＮＮ）には連続時間的なものと，離散時間的なものがあり，本研究は主に後者について議論する．ＣＮＮは1988年にカリフォルニア大学バークレ校のL.O.Chua教授らによって提案され，現在，アメリカ，ヨーロッパを中心に盛んに研究が進められている．ＣＮＮは従来のニューラルネットワークと異なり，近傍のセルとのみ結合しているため集積回路としての実現が容易であり，画像処理用ＣＮＮとして注目されている．

第一章では，ニューラルネットワークに関する研究の動向，および，人間の目と同様な処理機能を持つ連続時間ＣＮＮに関する研究の動向と，この論文で議論している離散時間ＣＮＮの背景について簡単に述べている．

第二章では，離散時間的な非均一ＣＮＮとして，二相同期信号の回路モデルを提案し，その安定性等について議論してある．このモデルは各セルについて二相同期信号１個で実現できるため，VLSIの実現が容易であると云う特徴がある．まず，モデルの動作原理から状態電圧，出力電圧の動作領域を明かにした．このことは物理的に実現可能なＣＮＮを設計するために重要である．つぎに，安定性を議論するためにエネルギー関数からリアプノフ関数を定義し，その関数の時間単調減少の条件を利用して，大域的な安定性を持つ離散時間ＣＮＮの設計方法を明らかにした．

第三章では，非線形システムにおける平衡点の求解法について議論している．連想記憶に用いられるＣＮＮは多くの平衡点をもち，

入力信号によってどの平衡点に到達するかが決定せられる．ロバストな連想記憶用ＣＮＮを設計するためには，このような平衡点を調べることが必要である．ここでは，解曲線追跡法に基づいた複数解の求解アルゴリズムを提案している．このアルゴリズムは急激な解曲線の変化を効率よく追跡できるように，エルミート予測子と BDF 積分公式に基づいている．また，大規模系に適用できるようにニュートン・ラフソン法の代わりにブラウンの反復法を採用している．このようなアルゴリズを採用することによりロバストなＣＮＮの設計が可能となる．

第四章では，離散時間ＣＮＮによる連想記憶について述べている．連想記憶は人間の脳の基本的な機能であり，ニューラルネットワーク応用研究の一つとして古くから盛んに研究されている．本章では，離散的なＣＮＮを用いた外積学習アルゴリズムと中点写像アルゴリズムの２種類の記憶方式を提案し，その性質を解明している．まず，前者は，入力パターンに対して，エネルギー関数の値が最少になるようにニューロン間の接続を表す重み行列を設定しようと云うものであり，これはHebbの理論に基づいている．また，上のような手法で学習されたパターンを連想記憶できる条件について議論した．中点写像アルゴリズムは重み行列の設定方法に対して，いま考えている中心セルからの近傍を定義し，近傍に存在するセルの状態をベクトル表示する．これを全てパターンについて実行し，このようにして決定された行列によって写像されるセルのパターンが，元の中心セルと同一のパターンを持つように重み行列を設定しようというもので，数学的には一般化逆行列の理論に基づいている．このような学習方法の特徴は入力された画像が全て連想されると云うことである．本章では，さらに，このことを応用例によって実証した．

第五章では，画像処理への応用として，輪郭抽出，雑音除去，視覚パターンの認識に対する離散的なＣＮＮについて述べている．多くの結果から処理時間は従来のものと比較して極端に短縮されることが分かった．また，不均一離散時間ＣＮＮによって，一つ画面中に多数の異なる視覚パターンを同時に認識できることも分かった。

第六章では，不均一離散的なＣＮＮの特徴と今後の問題点について述べている．

様式９

論 文 審 査 の 結 果 の 要 旨

報 告 番 号	[✓ 甲 工] 乙 工 第 ２６ 号 工 修	氏 名	何 晨
審 査 委 員	主 査 牛田 明夫 教授 副 査 為貞 建臣 教授 副 査 木内 陽介 教授		

学位論文題目

Non-Uniform Cellular Neural Network and its Applications

審査結果の要旨

セルラーニュラルネットワーク(CNN)に関する研究は1988年にカリフォルニア大学の L.O. Chua らによって提案され，アメリカ，ヨーロッパを中心に盛んに研究されている．CNNにはアナログ型と離散的なものがあり，彼の研究はセル間が不均一に結合した二相同期信号を利用した離散時間CNN（DTCNN）についてである．本論文では，DTCNNの安定性対する十分条件を導出し，２種類の連想メモリーの設計手法を提案した．一つはHebbの理論に基づいた外積学習アルゴリズムを利用するものであり，他は写像されるセルのパターンが元の中心セルと同一のパターンを持つように重み行列を設定しようという中点写像アルゴリズムをもちいるものである．これに関する論文は電子情報通信学会の英文誌，中国，ハワイで開催された国際会議に報告されている．

一般に，ニューラルネットワークは多くの平衡点を持っており，その性質を解明するためには，演算効率のよい平衡点の求解法が必要である．本論文では予測子ー修正子法を用いた一求解法を提案している．これに関する論文は電子情報通信学会の英文誌に報告されている．

さらに，DTCNNの画像処理への応用として，輪郭抽出，雑音除去，視覚パターンの認識用CNNについて述べている．

彼の提案したDTCNNはとくに演算処理時間と安定性の点で優れており，VLSIによる実現が期待されている．

以上のように本研究は，学会，国際会議でも評価されており，本論文は博士（工学）の学位授与に値するものと判定する．

Non-Uniform Cellular Neural Network and its Applications

September 1994

CHEN HE

②

Non-Uniform Cellular Neural Network and its Applications

Dissertation Submitted
in Candidacy for
the Ph. D. Degree

by

Chen HE

Doctor Course for System Engineering
Graduate School, Engineering
Tokushima University, Japan

September 1994

Contents

List of Figures

List of Tables

Acknowledgements

This present thesis is a collection of studies carried out under the direction and guidance of Professor Akio Ushida of Tokushima University during 1991-1994. Therefore, I would like to take this opportunity to thank all those who helped me and provided me to accomplish my work.

Four years ago, I came to Tokushima University as a foreign visiting scholar. Since October of 1991, I have been working for Ph. D. degree in electronics engineering. I would like to express my great gratitude to Professor Akio Ushida for supplying me with this research and study chance, and also for his guidance, helpful advice and continuous encouragement during this period.

I wish to gratefully acknowledge my indebtedness to Professor Kiyoshi Kohno, Director of Faculty of Engineering, and His wife Mrs. Nayoko Kuhno who is my first Japanese Language Lecturer after I came to Japan. As very nice teachers, seniors and friends, they give me and my wife with many helps and very warm encouragements. I never forget their important supports and great helps forever.

I sincerely thank Professor Akio Sakamoto, Dr. Takashi Shimamoto and Dr. Yoshifumi Nishio for their helpful advice and kind encouragement throughout the work.

I wish to express my appreciation to Professor Takeomi Tamesada and Professor Yoshisuke Kinouchi of Tokushima University, the member of my thesis committee, for their comments and suggestions, and particularly for their careful and critical reading of this dissertation.

I am very much grateful to Mr. Yoshihiro Yamagami of Tokushima University for his very useful helps and warm encouragements. From him I learned many computer techniques during these four years. I believe he is one of the most excellent computer technical expert who I know until now.

I have enjoyed an interesting research life with my colleagues of Ushida and Sakamoto Laboratory and I am deeply indebted to them. I am very grateful to Mr. Shinzao Liu, Mr. Zhen Hong, Mr. Youichi Tanji, Mr. Yoshinobu Setou, Mr. Motoshi Miyamoto, Mr. Takashi Ikoma, Mr. Kouji Nakai, Mr. Kazuhisa Hirata and other peoples for their friendly helps during the period of this research. I also wish to thank all members in Ushida and Sakamoto Laboratory, Department of Electrical and Electronic Engineering, Tokushima University, for a warm and interesting research environment which I have enjoyed in Ushida and Sakamoto Laboratory.

I would like to express my deep appreciation to my wife Ling-ge Jiang for her affectionate continuous encouragement and great help.

At last, but no the least, I would like to express my gratitude to my father, my mather, my wife's father, my wife's mather and my son who helped and encouraged me greatly, and gave all kinds of assistances to obtain my educational aims under the most comfortable atmosphere and to make this study a great success.

Department Electrical and Electronic Engineering
Faculty of Engineering
TOKUSHIMA University

Chen He

Chen HE

Chapter 1

General Introduction

1.1 Background

1.1.1 The neuron

The human brain is one of the most complicated things that we have studied in detail, and in the same time, is poorly understood on the whole.
The neuron is the basic unit of the brain, it is shown in Fig.1.1.

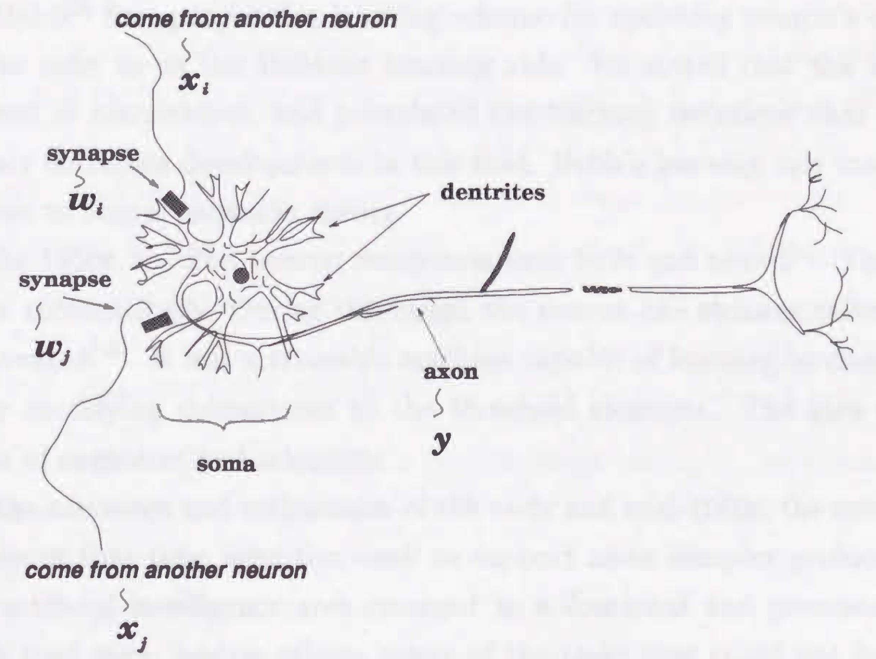


Figure 1.1: The structure of the neuron

The soma is the body of the neuron. Attached to the soma are long, irregularly sharped filaments, called dendrites. The dendrites act as connections through which all the inputs to the neuron arrive. Another type of nerve process attached to the soma is called an axon. This is electrically active and serves as the output channel of the neuron. The axon terminates in a specialized contact called a synapse that couples the axon with the dendrites of another cells.
The dendrites can perform addition on the inputs. The axon is a non-linear device, producing a monotone increasing output voltage when the resting potential within the soma varies over a certain critical threshold. The contact strength between the dendrites and other neuron's synapse is different from another.

1.1.2 Development of artificial neural networks

The year 1943 is often considered as the initial year in the development of artificial neural networks[1]. McCulloch and Pitts[2] outline the first formal model of an elementary computing neuron. The model included all necessary logic computing element. Although the implementation of this model was not technologically feasible in that era, their model laid the groundwork for future developments.

Donald Hebb[3] first proposed a learning scheme for updating neuron's connections that we now refer to as the Hebbian learning rule. He stated that the information can be stored in connections, and postulated the learning technique that had a profound impact on future developments in this field. Hebb's learning rule made primary contributions to neural networks theory.

During the 1950s, the first neuron computers were built and tested[4]. They adapted connections automatically. During this stage, the neuron-like element called a perception was invented[5]. It was a trainable machine capable of learning to classify certain patterns by modifying connections to the threshold elements. The idea caught the imagination of engineers and scientists.

Despite the successes and enthusiasm of the early and mid-1960s, the existing learning theorems in that time were too weak to support more complex problems. Meanwhile, the artificial intelligence area emerged as a dominant and promising research field, which took over, among others, many of the tasks that could not be solved by neural networks of that time. the research activity in the neural network field had sharply decreased.

During the period from 1965 to 1984, further pioneering work was accomplished by a handful of researchers. The study of learning in in networks of threshold elements and of the mathematical theory of neural networks was pursued by S.Amari[6,7]. Also in Japan, K.Fukushima developed a class of neural network architectures called as neocognitrons[8]. The neocognitron is a model for visual pattern recognition and is concerned with biological plausibility. The network emulates the retinal images and processes them using two- dimensional layers of neurons.

Associative memory research has been pursued by, among others, T. Kohonen[9-11] and J.A.Anderson[12]. Unsupervised learning networks were developed for feature mapping into regular arrays of neurons[10]. S.Grossberg and G.Carpenter have introduced a number of neural architectures and theorems and developed the theory of adaptive resonance networks[13,14].

During the period from 1982 until 1986, several seminal publications were published that significantly furthered the potential of neural networks. The era of renaissance started with J.J.Hopfield[15, 16] introducing a recurrent neural network architecture for associative memories. His papers formulated computational properties of a fully connected network of units.

Another revitalization of the field came from the publication in 1986 of two volumes on parallel distributed processing, edited by J.McClelland and D.Rumelhart[17]. The new learning rule and other concepts introduced in this work have removed one of the most essential network training barriers that grounded the mainstream efforts of the mid-1960s.

Beginning in 1986-87, many new neural networks research programs were initiated. Among of them, the researchs of cellular neural network theorems and applications are very activity and developed in surprising speed.

1.1.3 Cellular neural networks

Cellular neural network(CNN) is a nonlinear dynamical analog processing array having a 2-, or 3-dimensional grid architecture[18, 19]. There are only finit local connections from each processing cell to their adjacent elements, so that it is very suitable for the tasks where signal values are placed on a regular 2-D or 3-D geometric grid and the direct interactions between the signals are limited within a local neighborhood[20]. Differing from general neural networks, CNN cells capture the geometric, nonlinear, and/or delay-type properties in the interaction weights. Also differing from Hopfield network, due to their local connectivity, CNN can be easily realized with VLSI technique. Meanwhile, the range of dynamics and the connection complexity are independent of the total number of processing cells, the implementation is reliable and robust.

Since 1988, just in a short period, it has given rise to wide interests in the theoretical researches for various generalizations and their applications in the areas like as image processing, pattern recognition, robot vision, motion detection and others. T.Roska and L.O.Chua presented a structure with nonlinear and delay-type templates[21], H.Harrer and J.A.Nossek extended the continuous model to discrete-time architecture[22]. At the same time, many other researchers also make significant contributions to the CNN paradigm, which have been documented in some proceedings[23, 24] and special issues[25, 26].

1.2 Purpose of this study

In Chapter 2, first, we will show the cell model of the continuous-time CNN, and some typical types of 2-D array structures briefly. After introducing a two phases synchronous-updating signal into a continuous-time CNN, we obtain a synchronous-updating CNN, we call it as SCNN. By extracting the values of state variations v_i and output variations y_i at the updating moments $t = kT,\ k = 0, 1, 2.\cdots$, we derive a discrete-time CNN which topology and output function are distinct from the DTCNN presented by Harrer and Nossek. With the dynamic route method, the dynamical properties of out DTCNN are analyzed. Moreover, the generalized energy functions for our SCNN and DTCNN are presented respectively. After then, two convergent theorems for our DTCNN are described. Since these convergent theorems are suitable for generalized non-uniform DTCNN, they provide the potential to apply our DTCNN more widely, for examples, to multi-types of visual patterns recognition, associative memories and others.

In Chapter 3, we present a modified BDF curve tracing method. The results shows this algorithm could be used efficiently to trace those solution curve with some sharp turning points. Specially, we want to point out that the Brown method is a kind of the Gauss-Seidel algorithm to be used for nonlinear algebraic functions. It is known that the convergence ratio is second order near to the solution. Furthermore, a number of the function evaluations is $(N^2 + 3N)/2$ when the function consists of N functions. Observe that that the Newton method takes N^2 evaluations of the partial derivatives and N evaluations of functions. Thus, the Brown method is efficiently applied to trace solution curve, such that the approximate solution is obtained by Hermite polynomial.

The algorithm presented here can be useful in the analysis of neural networks, e.g. during the design of templates for cellular neural networks. It can be applied to large networks provided that the extreme sparity and the structure of the coefficients are exploited. The method can be applied for some types of neurons with smooth non-linear output functions or piecewise linear output functions. In general, there does not seem to be much hope for an efficient way to find all equilibrium points in a given neural network unless appropriate guidelines are followed during the synthesis process.

In Chapter 4, first, we describe the outer product learning approach to set up the weights with suitable values which is related to the object patterns information, it is called as storing object patterns into a cellular associative memory. Meanwhile, some analyses about the stationary property of the cellular associative memory with outer product learning rule are taken. A condition is presented which ensure the stored patterns as the stable states of a cellular associative memory. After then, a middle-mapping learning algorithm for cellular associative memory is presented, which makes full use of the properties of the cellular neural network so that every stored prototype can be guaranteed as an equilibrium point of our memory. At the same time, it has ability of iterative learning. This kind of computation is typical of a learning process: once the synaptic matrix has been computed from a given set of prototype vectors, the addition of one extra item of knowledge does not require that the whole computation is performed again. One just has to carry out one iteration, starting from the previous matrix, so that the computational efficiency can be improved. Besides, its implementation with circuits is more feasible because the weight matrix is not symmetric.

Since the synchronous updating rule is used in both of them, their associative speeds are very fast compared to the Hopfield associative memory.

In Chapter 5, first, we apply our DTCNN to the feature extraction and noise remove for the image processing. Some real image are chosen as our processing object and then, input to DTCNN as both input signals and initial states. After a few times iterative operations, desired results are obtained. Although the same function can also be carried by continuous-time CNN, time consuming differential operations are taken during the procedure and more iterative operations are required, Contrasting it, our DTCNN realized by software simulation can do them only with 5% or 10%

computing cost, so it is faster and efficienter than continuous-time CNN in this case. After then, we illustrate the potential of DTCNN for the visual pattern recognition. From a prototype composed by over two types of elements, we can detect desired visual patterns successfully. When there exist obvious differences between these two types of elements, it is easily recognized by human vision system. But for some similar composed elements, it is said to be very difficult and time consuming for human vision system. For our DTCNN, after suitable template is designed, it is easily and quickly to pick out our desired patterns from a prototype in both cases. This technique can be applied for robot vision. Finally, based on our convergent analysis result in Chapter 2, we design space-varying non-uniform DTCNN for multiple visual patterns recognition. In a non-uniform DTCNN, two or more templates are used for the cells lying in different region of 2-D processing array. Two examples are given to show the ability of non-uniform DTCNN to detect multiple visual patterns from a prototype at the same time, which have distinct geometrical character so they can not be picked out by unique template at once. It extented the application region of our DTCNN more over. Since the weight matrix A and B contributed by two or more distinct templates are not symmetrical matrixes, or, $A_{ij} \neq A_{ji}$ and $B_{ij} \neq B_{ji}$ generally, the stability analysis of non-symmetric continuous-time CNN is still open problem and dose not been solved, the similar application by continuous-time CNN has not been reported until now.

References

[1] J.M.Zurada, *Introduction to: Artificial neural systems*, West Publishing Co., 1992.

[2] W.S.McCulloch and W.H.Pitts, "A logical calculus of the ideas imminent in nervous activity", *Bull. Math. Biophys.*, vol.5, pp.115-133, 1943.

[3] D.O.Hebb, *The organization of behavior, a neuropsychological theory*, New York: John Wiley, 1949.

[4] M.Minsky, "Neural nets and the brain", *Doctoral Dissertation*, Princeton University, NJ, 1954.

[5] F.Rosenblatt, "The perception: a probabilistic model for information storage and organization in the brain", *Psych. Rev.*, vol.65, pp.386-408, 1958.

[6] S.Amari, "Learning patterns and pattern sequences by self-organizing nets of threshold elements", *IEEE Trans. Computers*, vol:C-21, pp.1197-1206, 1972.

[7] S.Amari, "Neural theory of association and concept formation", *Biol. Cybern.*, vol.26, pp.175-185, 1977.

[8] K.Fukushima and S.Miyaka, "Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", *Biol. Cybern.*, vol:36(4), pp.193-202, 1980.

[9] T. Kohonen, *Associative memory: a system-theoretical approach*, Berlin: Spring-Verlag, 1977.

[10] T. Kohonen, "A simple paradigm for the self-organized formation of structured feature maps", in *Competition and cooperation in neural nets.*, ed. S.Amari, M.Arbib. vol.45, Berlin: Spring-Verlag, 1982.

[11] T. Kohonen, *Self-organization and associative memory.* Berlin: Spring-Verlag, 1984.

[12] J.A.Anderson, J.W.Silverstein, S.A.Rite and R.S.Jones, "Distinctive features, categorical perception, and probability learning: some applications of neural model", *Psych. Rev.*, vol.84, pp.413-451, 1977.

[13] S.Grossberg, "Classical and instrumental learning by neural networks", in *Progress in Theoretical Biology*, vol.3, New-York: Academic Press, pp.51-141, 1977.

[14] S.Grossberg, *Studies of mind and brain: neural principles of learning perception, development, cognition and motor control*, Boston: Reidel Press, 1982.

[15] J.J.Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proc. Natl. Sci.*, vol.79, pp.2254-2258, 1982.

[16] J.J.Hopfield, "Neurons with graded response have collective computational properties like those of two state neurons", *Proc. Natl. Acad. Sci.*, vol.81, pp.3088-3092, 1984.

[17] T.L.McClelland and D.E.Rumelhart, *Parallel distributed processing*, Cambridge: MIT Press and the PDP Research Group, 1986.

[18] L.O.Chua and L.Yang, "Cellular neural network: theory", *IEEE Trans. Circ. Syst.*, vol.CAS-35, pp.1257-1272, 1988.

[19] L.O.Chua and L.Yang, "Cellular neural network: application", *IEEE Trans. Circ. Syst.*, vol.CAS-35, pp.1273-1290, 1988.

[20] L.O.Chua and T.Roska, "The CNN paradigm", *IEEE Trans. Circ. Syst.-I*, vol.CAS-40, pp.147-156, 1993.

[21] T.Roska and L.O.Chua, "Cellular neural networks with nonlinear and delay-type template elements", in *Proc. IEEE Int. Workshop on CNN and Their Applications*, pp.12-25, 1990.

[22] H.Harrer and J.A.Nossek, "Discrete-time cellular neural networks: Architecture, applications and realization", *Int. J. of Circuit Theory and Applications*, vol.20, pp.453-467, 1992.

[23] *Proc. IEEE Int. Workshop on CNN and Their Applications (CNNA-90)*, Budapest, Oct. 1990.

[24] *Proc. 2nd IEEE Int. Workshop on CNN and Their Applications (CNNA-92)* Munich, Sept. 1992.

[25] *Inter. J. Cir. Theor. Appl.*, Special Issue on Cellular Neural Networks, vol.20, 1992.

[26] *IEEE Trans. Circ. Syst.*, Special Issue on Cellular Neural Networks, vol.40, 1993.

Chapter 2

Discrete-time Cellular Neural Network

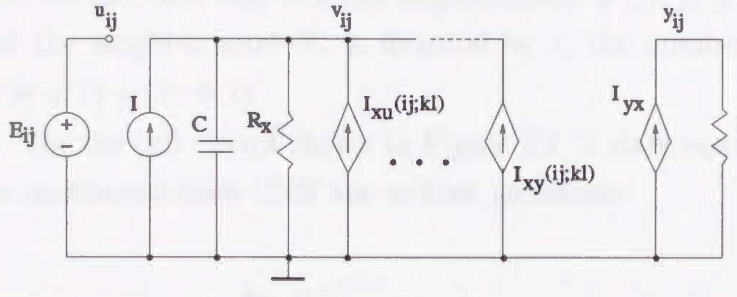
2.1 Introduction

In this chapter, first, a continuous-time cellular neural network(CNN) presented by Chua and Yang[1], and the network grid structure are introduced, their basic properties of continuous-time CNN are briefly described here. After then, as our study, a synchronous-updating clock signal is introduced into an original continuous-time CNN, by sampling the output values at a series of updating moments, we obtain another type of discrete-time CNN which circuit topology is different from that by Harrer and Nossek[2]. After then, some detail analyses about the dynamical property and stability of out DTCNN are performed. The results show that, if the parameters in the templates are designed carefully so that the convergent sufficient conditions are met, the generalized energy function is monotone decreasing and the stability of DTCNN can be guaranteed.

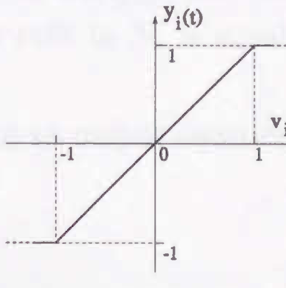
Cellular neural network(CNN) is a locally connected, nonlinear dynamical analog processing array having 2-, or 3-dimensional grid architecture. One processing element, called as a cell, with piecewise linear output function template is shown on Figure 2.1.

In general, all cells are arranged on a 2-D geometrical regular grid(one layer), but this layer can be duplicated to form 3-D multilayer CNN if it is required. Some typical 2-D regular grids are shown in Figure 2.2.

For simplicity, in this study, we just consider the case in which a 2-D rectangular regular grid with M rows and N columns, as Figure 2.2(a), is used. In this grid, each square represents a CNN cell. The $c(i, j)$ denotes a cell lying in ith row and jth column.

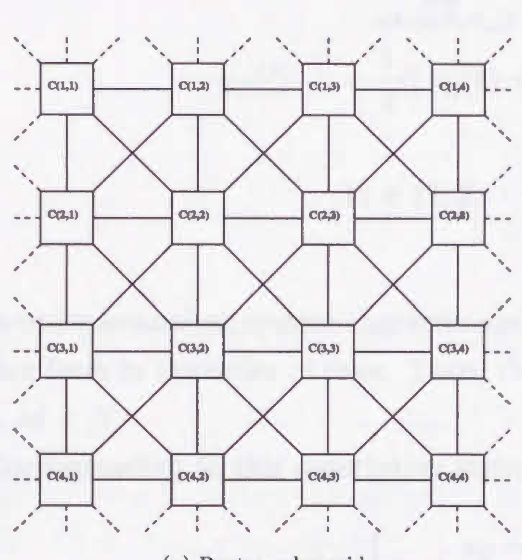


(a) A cell circuit

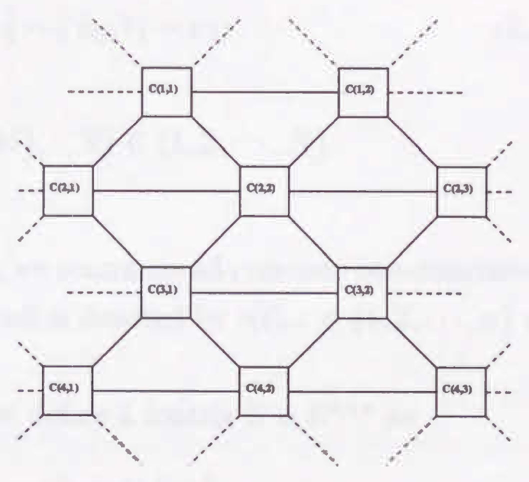


(b) Piecewise linear function

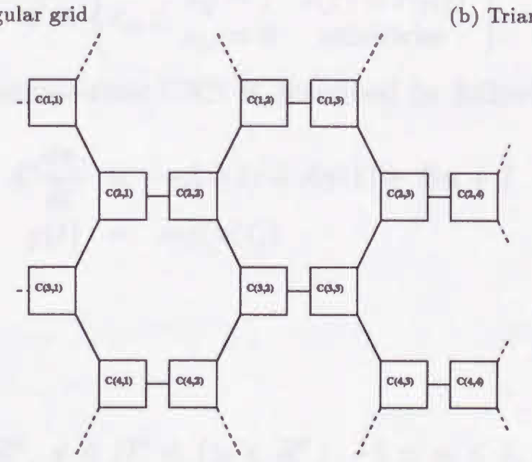
Figure 2.1: A continuous-time CNN cell



(a) Rectangular grid



(b) Triangular grid



(c) Hexagonal grid

Figure 2.2: Some typical 2-D regular grids

Every cell just only connects directly with near cells, which constitute a neighborhood N_r around that cell, and the neighborhood of $c(i,j)$ is denoted by $N_r(i,j)$. The radius of the neighborhood N_r is denoted by r, the number of the cells in N_r is equal to $(2r+1)\times(2r+1)$.

For the cell circuit shown in Figure 2.1, a state equation and an output equation of a continuous-time CNN are written as follows:

$$\begin{aligned} C\frac{dv_{ij}(t)}{dt} &= -\frac{1}{R_x}v_{ij}(t) + \sum_{c(k,l)\in N_r(i,j)} A(i,j;k,l)y_{kl}(t) \\ &+ \sum_{c(k,l)\in N_r(i,j)} B(i,j;k,l)u_{kl}(t) + I \qquad (2.1a) \\ y_{ij}(t) &= \frac{1}{2}(\mid v_{ij}(t)+1\mid - \mid v_{ij}(t)-1\mid) \qquad (2.1b) \end{aligned}$$

$$\forall i \in \{1,2,\cdots,M\},\quad \forall j \in \{1,2,\cdots,N\}$$

In order to analyze system character easily, we rearrange all cells into one-dimensional vector form in the order of rows. Then, the cell is denoted by $c(i), i \in \{1,2,\cdots,n\}$ and $n = M\times N$.

Corresponding to this description style, we define a matrix $S \in R^{n\times n}$ as

$$S \equiv \left\{ s_{ij} : \begin{array}{ll} s_{ij}=1 & c(j)\in N_r(i) \\ s_{ij}=0 & \text{otherwise} \end{array} \right\} \qquad (2.2)$$

In this way, the continuous-time CNN is described by following equations.

$$\begin{aligned} C\frac{d\mathbf{v}}{dt} &= -\Lambda\mathbf{v}(t) + A\mathbf{y}(t) + B\mathbf{u} + I \qquad (2.3a) \\ \mathbf{y}(t) &= sat(\mathbf{v}(t)) \qquad (2.3b) \end{aligned}$$

where

$$\mathbf{v}\in R^n,\ \mathbf{u}\in R^n,\ \mathbf{y}\in D^n \equiv \{y_i \in R^n:\ -1\le y_i\le 1,\ i=1,\cdots,n\};$$

$$\Lambda \in R^{n\times n} \equiv diag[\lambda\ \cdots\ \lambda] \text{ with } \lambda = \frac{1}{R_x} > 0;$$

$$A \in R^{n\times n} \equiv \{A_{ij};\ 1\le i\le n,\ 1\le j\le n\};$$

$$B \in R^{n \times n} \equiv \{B_{ij};\ 1 \leq i \leq n,\ 1 \leq j \leq n\}$$

Here, both A and B are sparse matrixes. Their elements satisfy the following conditions:

$$A_{ij} = A_{ij} \cdot s_{ij} \tag{2.3c}$$

$$B_{ij} = B_{ij} \cdot s_{ij} \tag{2.3d}$$

It means that, when $s_{ij} = 0$, A_{ij} and B_{ij} are equal to zero, but in other case, they are equal to arbitrary real number decided by the particular purpose of CNN.

$$I \in R^n \equiv \{\, I_1\ I_2\ \cdots\ I_n \,\}$$

$$sat(\mathbf{v}(t)) = [\, sat(v_1(t))\ sat(v_2(t))\ \cdots\ sat(v_n(t))\,]^T$$

$$sat(v_i) = \begin{cases} 1 & v_i > 1 \\ v_i & -1 \leq v_i \leq 1 \\ -1 & v_i < -1 \end{cases} \tag{2.3e}$$

When the next two conditions are met,

$$|v_i(0)| \leq 1, \quad |u_i| \leq 1 \tag{2.4}$$

the range of dynamics is bounded by a single number M which can be calculated in terms of the cloning templates:

$$M = \max\{|v_i|\} = \max\{1 + R_x|I| + R_x \sum_{c(j)\in N_r(i)} (|A_{ij}| + |B_{ij}|)\} \tag{2.5}$$

Moreover, if the following condition is satisfied,

$$A_{ii} > \frac{1}{R_x} \tag{2.6}$$

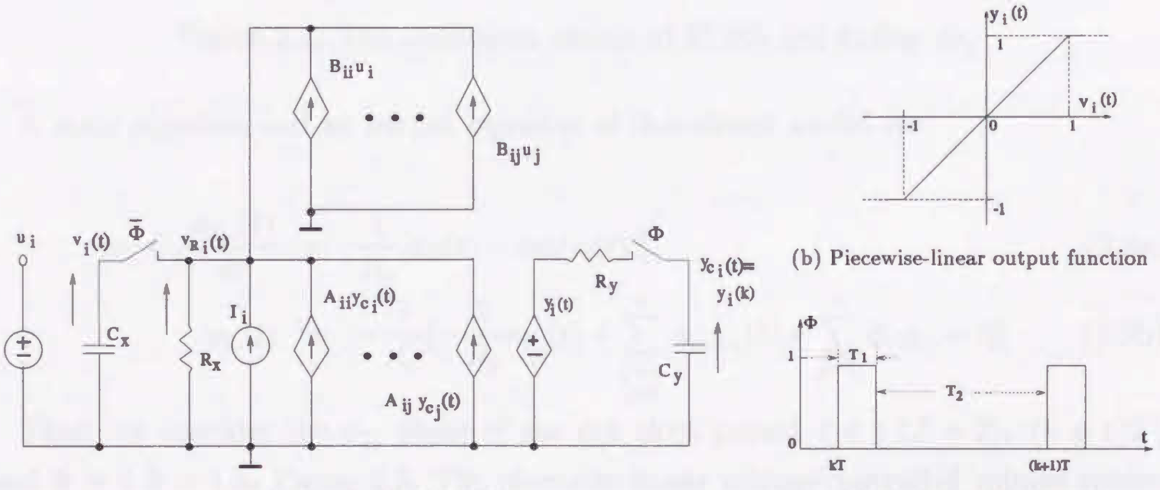
for symmetric continuous-time CNN, its stability can be proved. Then, the convergent results can be derived as follows:

$$\lim_{t\to\infty} \mid v_i(t) \mid > 1 \tag{2.7}$$

$$\lim_{t\to\infty} y_i(t) = \pm 1 \tag{2.8}$$

2.2 Discrete-time cellular neural network

In this section, we build up a model for our discrete-time CNN. First, a state updating signal is introduced into a cellular neural network, so that a synchronous-updating cellular neural network(SCNN) is obtained. which means that, at the kth updating time $t = kT$, the states of all cells are altered simultaneously. Here, T describes the updating period in our SCNN, According to this rule, a cell of SCNN is shown as Figure 2.3(a).



(a) A cell $c(i)$ of SCNN

(b) Piecewise-linear output function

(c) Clock signal for SCNN

Figure 2.3: A synchronous-updating CNN cell

Φ and $\bar{\Phi}$ in Figure 2.3(a) are a clock signal and its inverse, they control two updating switchs respectively. Φ is shown as Figure 2.3(b). C_x and C_y are two sample-hold capacitors.

During ϕ_{T_1} phase of the kth clock period, $t \in (\,kT,\ kT + T_1\,], k = 0, 1, 2, \cdots, \Phi = 1$, $\hat{\Phi} = 0$, the terminal voltage $v_i(t)$ in C_x is kept as its initial value $v_i(t) = v_i(k),\ k = kT$. The voltage-controlled voltage source $y_i(t) = \text{sat}(v_i(t))$ is also a constant during this phase, denoted by $y_i(k)$. The capacitor C_y is charged by the voltage source $y_i(k)$ through the resistance R_y. Since the value of R_y is a very small and in the order of the internal resistance of the voltage source $y_i(t)$, this charge is finished quickly in very short time, we can write $y_{c_i}(t) = y_i(k)$ after the transient response, about $2.3R_yC_y$, is completed. The voltage $v_{R_i}(t)$ is determined by the resistance R_x, the current source

I_i and the voltage-controlled current sources $A_{ij}y_{c_j}(t)$ and $B_{ij}u_j$, $j \in N_r(i)$. The equivalent circuit is illustrated in Figure 2.4, it is a one-order nonlinear dynamic circuit.

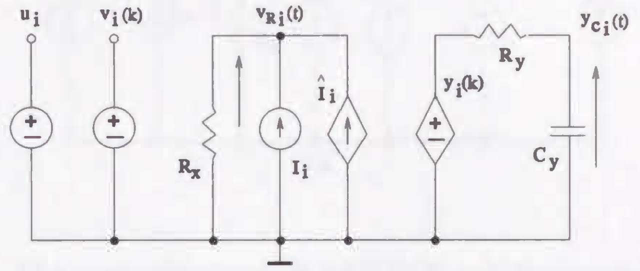


Figure 2.4: The equivalent circuit of SCNN cell during ϕ_{T_1}

A state equation and an output equation of this circuit model are

$$C_y \frac{dy_{c_i}(t)}{dt} = \frac{1}{R_y}[y_{c_i}(t) - \mathrm{sat}(v_i(t))] \tag{2.9a}$$

$$y_{c_i}(t) = -\frac{1}{A_{ii}}[-\frac{1}{R_x}v_{R_i}(t) + \sum_{\substack{j=1\\j\neq i}}^{n} A_{ij}y_{c_j}(t) + \sum_{j=1}^{n} B_{ij}u_j + I_i] \tag{2.9b}$$

Then, we consider the ϕ_{T_2} phase of the kth clock period, $t \in (kT + T_1, (k+1)T]$ and $\Phi = 0$ $\hat{\Phi} = 1$ in Figure 2.3. The piecewise linear voltage-controlled voltage source $y_i(t)$ is varying with the voltage $v_i(t)$, but since $\Phi = 0$, it has no feedback effection to $v_i(t)$ during this phase. The one-order dynamic circuit is consists of C_x, R_x, I_i and the voltage-controlled current sources $A_{ij}y_{c_j}(t)$ and $B_{ij}u_j$, here, $y_{c_j}(t)$ is the terminal voltage in C_y of the jth cell. Since the terminal voltage $y_{c_i}(t) = y_i(k)$, $i = 1, 2, \cdots, n$ is held as a constant here, this dynamic circuit is equivalent to a linear RC dynamic circuit, the initial value of the terminal voltage in C_x is determined by $v_i(t)$ in previous clock updating moment $t = kT$, i.e., $v_i(k)$. Obviously, after $2.3R_xC_x$, the transient response is settled to zero, the circuit must convergent to its steady state. The equivalent circuit is shown in Figure 2.5.

Corresponding to this circuit model, we can derive a state equation and an output equation as follows:

$$C_x \frac{dv_i(t)}{dt} = -\frac{1}{R_x} v_i(t) + \sum_{j=1}^{n} A_{ij}\, y_j(k) + \sum_{j=1}^{n} B_{ij}\, u_j + I_i \tag{2.10a}$$

$$y_i(t) = \mathrm{sat}(v_i(t)) \tag{2.10b}$$

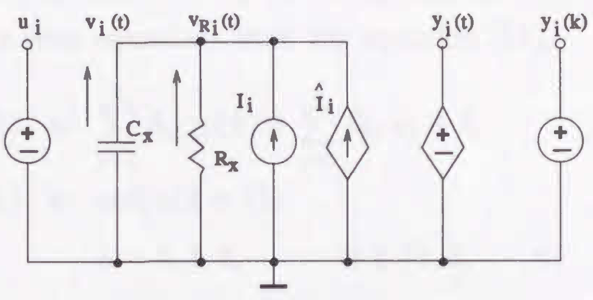


Figure 2.5: The equivalent circuit of SCNN cell during ϕ_{T_2}

In contrast to continuous-time CNNs, $y_j(k)$ in the state equation (2.10a) is a sampled state of output variant at the kth updating time $t = kT$ and is held in the capacitor C_y. The feedback strength $\sum_{j=1}^{n} A_{ij} y_j(k)$ from the neighbor cells to a cell $c(i)$ remains a constant value for $t \in (kT, (k+1)T]$, but the variants $v_i(t)$ and $y_i(t)$ in the equations are varying continuously with time t.

The equations (9) and (10) describe the state and output equations of SCNN in ϕ_{T_1} and ϕ_{T_2} phases respectively. Combining them together, we get a set of equations to describe SCNN in a whole clock period.

When $t \in (\,kT,\ kT + T_1\,]$

$$
\begin{aligned}
C_y \frac{dy_{c_i}(t)}{dt} &= \frac{1}{R_y}[y_{c_i}(t) - \mathrm{sat}(v_i(t))] && (2.11a) \\
y_{c_i}(t) &= -\frac{1}{A_{ii}}[-\frac{1}{R_x} v_{R_i}(t) + \sum_{\substack{j=1 \\ j \neq i}}^{n} A_{ij} y_{c_j}(t) + \sum_{j=1}^{n} B_{ij} u_j + I_i] && (2.11b)
\end{aligned}
$$

When $t \in (\,kT + T_1,\ (k+1)T\,]$

$$
\begin{aligned}
C_x \frac{dv_i(t)}{dt} &= -\frac{1}{R_x}\, v_i(t) + \sum_{j=1}^{n} A_{ij}\, y_j(k) + \sum_{j=1}^{n} B_{ij}\, u_j + I_i && (2.11c) \\
y_i(t) &= \mathrm{sat}(v_i(t)) && (2.11d) \\
& \qquad k = 0, 1, 2, \cdots, \quad \forall i \in \{1, 2, \cdots, n\}
\end{aligned}
$$

In order to derive a DTCNN, extracting the state variable v_i and the output variable y_i at a series of updating moments $t = kT$, $k = 0, 1, 2. \cdots$, and assume the updating interval is long enough, i.e. $T \gg 2.3 R_x C_x$, then, after the transient response has decayed to zero, $dv_i(t)/dt = 0$ is kept at every updating moment. In this case, we can obtain the following discrete-time equations from the equation (11).

$$\begin{aligned} \frac{1}{R_x} v_i(k+1) &= \sum_{j=1}^{n} A_{ij}\, y_j(k) + \sum_{j=1}^{n} B_{ij}\, u_j + I_i && \text{(2.12a)} \\ y_i(k+1) &= \text{sat}(v_i(k+1)) && \text{(2.12b)} \\ & k = 0, 1, 2, \cdots, \quad \forall i \in \{1, 2, \cdots, n\} \end{aligned}$$

They describe the state and the output of our discrete-time CNN. Here, we derive a discrete-time CNN which topology is distinct from the DTCNN presented by Harrer and Nossek, but has more tighter corresponding relation with continuous-time CNN. Moreover, we can write them as vector equations as follows.

$$\begin{aligned} \frac{1}{R_x} \mathbf{v}(k+1) &= A\mathbf{y}(k) + B\mathbf{u} + I && \text{(2.13a)} \\ \mathbf{y}(k+1) &= \text{sat}(\mathbf{v}(k+1)) && \text{(2.13b)} \end{aligned}$$

where $k = 0, 1, 2, \cdots,\ \mathbf{v},\ \mathbf{y},\ \mathbf{u},\ I \in R^n,\ \ A,\ B \in R^{n\times n}$

2.3 Dynamical range of our DTCNN

In order to implement physical DTCNN, we need to investigate its dynamic range. In an continuous-time CNN, the topology of the network is unvaried, its dynamic range has been proved for the initial state $|v_i(0)| < 1$. But in our network, the topology is time-variant, the initial value of v_i in each updating period is obtained from the steady-state value of the last updating period, so that its initial value may be changed within its dynamical range. To get the dynamic range of synchronous-updating CNN, let us consider the equivalent circuit shown as Figure 2.6.

In Figure 2.6, $\hat{I}_i = \sum_{j=1}^{n} A_{ij}\, y_j(k) + \sum_{j=1}^{n} B_{ij}\, u_j$. While in ϕ_{T_2} of the kth clock period, $\Phi = 0, \bar{\Phi} = 1$, $v_i = v_{R_i}$, assuming $T_2 \gg 2.3 R_x C_x$, when $t = (k+1)T$, the circuit converges to its steady state. Analyzing the equivalent circuit in steady state, we can get the maximum value of $v_i(t)$ in steady state as follows.

$$\begin{aligned} v_i(t) &= R_x\,(\hat{I}_i + I_i) \\ &= R_x\,[\sum_{j=1}^{n} A_{ij}\, y_j(k) + \sum_{j=1}^{n} B_{ij}\, u_j + I_i\,] \end{aligned}$$

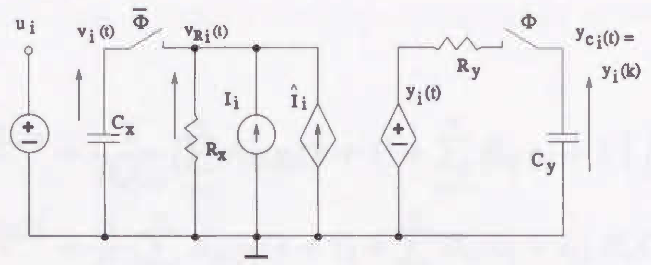


Figure 2.6: The equivalent cell circuit of SCNN

$$\begin{aligned} &\leq R_x\,[\sum_{j=1}^{n} \mid A_{ij} \mid \mid y_j(k) \mid + \sum_{j=1}^{n} \mid B_{ij} \mid \mid u_j \mid + \mid I_i \mid] \\ &\leq R_x\,[\sum_{j=1}^{n} \mid A_{ij} \mid + \sum_{j=1}^{n} \mid B_{ij} \mid \mid u_j \mid + \mid I_i \mid] \end{aligned} \tag{2.14}$$

here, we define

$$v_{max} = R_x \{ \max[\sum_{j=1}^{n} \mid A_{ij} \mid + \sum_{j=1}^{n} \mid B_{ij} \mid \mid u_j \mid] + \mid I_i \mid \} \tag{2.15}$$

Next, in ϕ_{T_1} phase of the next clock period, $t \in (\,(k+1)T, (k+1)T + T_1\,]$, $\Phi = 1$, $\bar{\Phi} = 0$, $v_i(t)$ is held as $v_i(k+1)$, but from (2.11b), we have

$$v_{R_i}(t) = R_x\,[\sum_{j=1}^{n} A_{ij} y_j(t) + \sum_{j=1}^{n} B_{ij} u_j + I_i\,]$$

Obviously,

$$v_{R_i}(t) \leq v_{max} \qquad t \in (\,(k+1)T, (k+1)T + T_1\,]$$

The value of the voltage source $y_i(t)$ depends on $v_i(t)$, so that during this phase it is also a constant. We denote it as $y_i(k+1)$. By charging to C_y, it is stored in C_y. In the following ϕ_{T_2} phase, $t \in (\,(k+1)T + T_1, (k+2)T\,]$, $\Phi = 0$ and $\bar{\Phi} = 1$ again, the dynamical character of this cell is described as

$$C_x \frac{dv_i(t)}{dt} = -\frac{1}{R_x} v_i(t) + \sum_{j=1}^{n} A_{ij}\, y_j(k+1) + \sum_{j=1}^{n} B_{ij}\, u_j + I_i$$

where $t \in (\,(k+1)T + T_1, (k+2)T\,]$.

In order to analyze the transient response, we solve this equation, the initial voltage in C_x is $v_i(k+1)$ obtained previously, thus we get

$$\begin{aligned}
& v_i(t) \\
= \; & v_i(k+1)\, e^{-\frac{t-(k+1)T-T_1}{R_xC_x}} + \frac{1}{R_xC_x}[\sum_{j=1}^{n} A_{ij}\, y_j(k+1) + \sum_{j=1}^{n} B_{ij}\, u_j + I_i\,] \int_{(k+1)T+T_1}^{t} e^{-\frac{t-\tau}{R_xC_x}} d\tau \\
= \; & v_i(k+1)\, e^{-\frac{t-(k+1)T-T_1}{R_xC_x}} + \frac{1}{C_x}[\sum_{j=1}^{n} A_{ij}\, y_j(k+1) + \sum_{j=1}^{n} B_{ij}\, u_j + I_i\,]\, R_xC_x\, [\, 1 - e^{\frac{(k+1)T+T_1-t}{R_xC_x}}\,] \\
= \; & v_i(k+1)\, e^{-\frac{t-(k+1)T-T_1}{R_xC_x}} + R_x\, [\sum_{j=1}^{n} A_{ij}\, y_j(k+1) + \sum_{j=1}^{n} B_{ij}\, u_j + I_i\,]\, (\, 1 - e^{\frac{(k+1)T+T_1-t}{R_xC_x}}\,)
\end{aligned}$$

where $t \in (\,(k+1)T + T_1, (k+2)T\,]$.

Then, we obtain the maximum value of $v_i(t)$ during whole ϕ_{T_2} phase.

$$\begin{aligned}
|\, v_i(t)\,| \;\leq\; & |\, v_i((k+1)T)\,|\, e^{-\frac{t-(k+1)T-T_1}{R_xC_x}} + R_x\, |\sum_{j=1}^{n} A_{ij}\, y_j(k+1) \\
& + \sum_{j=1}^{n} B_{ij}\, u_j + I_i\,|\, (\, 1 - e^{\frac{(k+1)T+T_1-t}{R_xC_x}}\,) \\
\leq\; & |\, v_{max}\,|
\end{aligned}$$

In this way, the biggest dynamical range of our DTCNN is illustrated as

$$|\, v_i(t)\,| \leq |\, v_{max}\,| \quad \text{for all } t$$

and

$$v_{max} = R_x\, \{\, |\, I_i\,| + \max[\sum_{j=1}^{n} |\, A_{ij}\,| + \sum_{j=1}^{n} |\, B_{ij}\,|\, |\, u_j\,|\,]\,\} \tag{2.16}$$

Here, the maximum value of $v_i(t)$ is less than that of the continuous-time CNN, in addition of that, the required initial condition $|\, v_i(0)\,| \leq 1$ is also eliminated.

2.4 DTCNN with binary output

From previous analysis, we proved the dynamical range of state variables in SCNN. Since a DTCNN is derived from a SCNN by extracting the states and outputs at a series of discrete time $t = 0, T, 2T, \cdots$, in general, the values of the state variables in DTCNN is a set of arbitrary numbers which amplitudes are less than $|v_{max}|$. Then, the output $y_i(k+1)$ in DTCNN is a variable from -1 to +1, its value is determined by the state variable $v_i(k+1)$ with (2.12b). But for some applications, the binary output is required. In this section, we give two theorems to describe the sufficient condition and necessary condition respectively to guarantee a set of binary outputs in our DTCNN.

Theorem 2.1 *If the following condition is satisfied, the output of DTCNN must be equal to +1 or -1.*

$$\mid A_{ii} \mid \geq \frac{1}{R_x} + \sum_{\substack{j=1 \\ j \neq i}}^{n} |A_{ij}| + \mid \sum_{j=1}^{n} B_{ij} u_j \mid + |I_i| \tag{2.17}$$

Proof: From (2.12a), we have

$$\frac{1}{R_x} v_i(k+1) = \sum_{j=1}^{n} A_{ij}\, y_j(k) + \sum_{j=1}^{n} B_{ij}\, u_j + I_i$$

When the output amplitude $|y_i(k+1)|$ is equal to 1, the amplitude of state variable must be greater than 1 or equal to 1, $|v_i(k+1)| \geq 1$. so that we have

$$\mid \sum_{j=1}^{n} A_{ij}\, y_j(k) + \sum_{j=1}^{n} B_{ij}\, u_j + I_i \mid \geq \frac{1}{R_x} \tag{2.18}$$

Since

$$\begin{aligned}
&\mid \sum_{j=1}^{n} A_{ij}\, y_j(k) + \sum_{j=1}^{n} B_{ij}\, u_j + I_i \mid \\
&\geq \mid \sum_{j=1}^{n} A_{ij}\, y_j(k) \mid - \mid \sum_{j=1}^{n} B_{ij}\, u_j \mid - |I_i| \\
&\geq |A_{ii} y_i(k)| - \sum_{\substack{j=1 \\ j \neq i}}^{n} |A_{ij} y_j(k)| - \mid \sum_{j=1}^{n} B_{ij}\, u_j \mid - |I_i| \\
&\geq |A_{ii} y_i(k)| - \sum_{\substack{j=1 \\ j \neq i}}^{n} |A_{ij}| - \mid \sum_{j=1}^{n} B_{ij}\, u_j \mid - |I_i|
\end{aligned} \tag{2.19}$$

From (2.18) and (2.19) we can find that when the equation (2.17) is met, $|v_i(k+1)| \geq 1$, then, $|y_i(k+1)| = 1$, the output is a binary value.

□

In the next theorem, we give the necessary condition for a binary output in DTCNN.

Theorem 2.2 *If the output $y_i(k+1)$ of DTCNN is a binary value, the following relation must be satisfied.*

$$|A_{ii}| \geq \frac{1}{R_x} - \sum_{\substack{j=1 \\ j \neq i}}^{n} |A_{ij}| - \mid \sum_{j=1}^{n} B_{ij}\, u_j \mid - |I_i| \tag{2.20}$$

Proof: When the output $y_i(k+1)$ of DTCNN is a binary value, we must have $|v_i(k+1)| \geq 1$ so that

$$|\sum_{j=1}^{n} A_{ij} y_j(k) + \sum_{j=1}^{n} B_{ij} u_j + I_i| \geq \frac{1}{R_x} \tag{2.21}$$

Since

$$\begin{aligned} &|\sum_{j=1}^{n} A_{ij} y_j(k) + \sum_{j=1}^{n} B_{ij} u_j + I_i| \\ &\leq |A_{ii}| + \sum_{\substack{j=1 \\ j \neq i}}^{n} |A_{ij}| + |\sum_{j=1}^{n} B_{ij}\, u_j| + |I_i| \end{aligned} \tag{2.22}$$

Considering the equation (2.21), it can be found when $|v_i(k+1)| \geq 1$, the equation (2.20) must be satisfied.

□

From above two theorems, it is known when the templates for DTCNN are designed to meet the equation (2.17), DTCNN can be used to realize the mapping from R^n to B^n so that the equation (2.13) can be written as

$$\frac{1}{R_x}\mathbf{v}(k+1) = A\mathbf{y}(k) + B\mathbf{u} + I \tag{2.23a}$$

$$\mathbf{y}(k+1) = \mathrm{sat}(\mathbf{v}(k+1)) \tag{2.23b}$$

where $k = 0, 1, 2, \cdots,\ \ \mathbf{v},\ \mathbf{u},\ I \in R^n,\ \ \mathbf{y} \in B^n,\ \ A,\ B \in R^{n \times n}$.

2.5 Stability analysis of DTCNN

In other chapters, we will give the applications of our DTCNN to image processing, include associative memory and visual pattern recognition. In these applications, first, a probe image with multiple gray level is inputted into DTCNN as its initial state at $t = 0$. Then, by some times of updating, final stable state is obtained, which means that the subsequent set of state are the same totally, no state change is risen by a clock signal. This final stable state is corresponding to an equilibrium point distributed in DTCNN's dynamical space. The output in the final stable state is an object image. In general, a pixel in the object image is a multiple gray value, but if the matrix A

and the matrix B are designed to meet the sufficient condition (2.17), the pixel in the object image is only binary value $+1$ and -1.

It is known that one of the most effective technique for analyzing the convergence properties of dynamic nonlinear circuits is Lyapunov's method. This method is also used by N.Fruehauf, L.O.Chua and E.Lueder for convergence analysis of reciprocal DTCNN with continuous nonlinearities[4], but their result is just suitable for reciprocal DTCNN, not for general case.

In this section, first, Lyapunov energy functions are defined to SCNN and DTCNN respectively. Then, the convergence condition is analyzed for SCNN. Since our DTCNN is obtained by extracting a series of updating moments from SCNN, all convergence analyzing results are easily extended to DTCNN. Our basic object is concentrated on general case, i.e., nonuniform and nonreciprocal DTCNN, which covers a reciprocal DTCNN's convergence condition by N.Fruehauf, L.O.Chua and E.Lueder just as a special case.

First, we define a generalized energy function for SCNN as follows:

$$\begin{aligned} E(t) &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} y_i(t) y_j(k) - \sum_{i=1}^{n}\sum_{j=1}^{n} B_{ij}\, y_i(t)\, u_j \\ &\quad + \frac{1}{2R_x}\sum_{i=1}^{n} y_i^2(t) - \sum_{i=1}^{n} I_i\, y_i(t) \end{aligned} \tag{2.24}$$

where $t \in (\,kT,\ (k+1)T\,],\ \ k = 0, 1, 2, \cdots$.

$y_j(k)$ in the equation (2.24) is a constant, $y_i(t)$ is a variable during ϕ_{T_2}, but in ϕ_{T_1} phase, $y_i(t)$ is kept as a constant so that $E(t)$ is invariant in this phase. At the updating moments of $t = (k+1)T,\ k = 0, 1, 2, \cdots$, the value of $y_j(k+1)$ is substituted into $y_j(k)$, so that $E(t)$ may jump at those moments.

Meanwhile, a generalized energy function for our DTCNN is defined as follows:

$$\begin{aligned} E(k+1) = &- \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\, y_i(k+1)\, y_j(k) - \sum_{i=1}^{n}\sum_{j=1}^{n} B_{ij}\, y_i(k+1)\, u_j \\ &+ \frac{1}{2R_x}\sum_{i=1}^{n} y_i^2(k+1) - \sum_{i=1}^{n} I_i\, y_i(k+1) \end{aligned} \tag{2.25}$$

Since the energy function denoted in (2.24) is consists of four sums of finite items, obviously, it is bounded, and then, we will prove that if the convergent conditions are met, this energy function is a uncontinuous monotone decreasing function, so that the differential of $E(t)$ to time T is equal to zero when T is tending toward infinite.

$$\lim_{t \to \infty} \frac{dE(t)}{dt} = 0 \tag{2.26}$$

Differing from the continuous-time CNN, here, the $E(t)$ is an uncontinuous monotone decreasing function so that this equation has two meanings, first, it denotes that $E(t)$ is kept as a constant during ϕ_{T_1} phase, but it is monotonely decreasing within ϕ_{T_2} phase. Second, around an updating instantaneous, the value of $E(t)$ may be suddenly changed, but its monotone decreasing property is still remained, so that the instantaneous value of $E(t)$ under a updating must less than or equal to the previous value before this updating. This situation is illustrated in Figure 2.7. During every period, $E(t)$ is continuous decreasing, but at some updating moments, for examples, at $t = (k-1)T$ and $t = (k+1)T$, it is reduced uncontinuously.

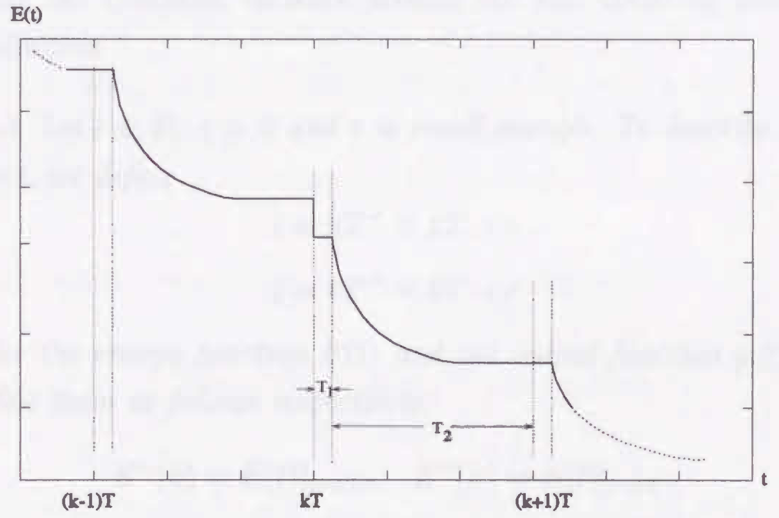


Figure 2.7: Uncontinuous monotone decreasing $E(t)$ curve

At the same time, from (2.24), we can find

$$\frac{dE(t)}{dt} = -\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\, y_j(k)\frac{dy_i(t)}{dt} - \sum_{i=1}^{n}\sum_{j=1}^{n} B_{ij}\, u_j \,\frac{dy_i(t)}{dt} + \frac{1}{R_x}\sum_{i=1}^{n} y_i(t)\,\frac{dy_i(t)}{dt} - \sum_{i=1}^{n} I_i \,\frac{dy_i(t)}{dt} \tag{2.27}$$

where $t \in (\,kT,\ (k+1)T\,]$

so that if the convergence conditions are met, the differential of $y_i(t)$ to time T is also equal to zero when T is tending toward infinite.

$$\lim_{t\to\infty} \frac{dy_i(t)}{dt} = 0 \tag{2.28}$$

It means not only during every phases ϕ_{T_2} in early stage, after the transient response, the stable state and output can be obtained, but also after some times of updating, the state and output are always stabilized in whole updating period.

Next, we provide the monotone decreasing conditions for DTCNN in two theorems. In the first theorem, a sufficient monotone decreasing condition of $E(t)$ is proved for general SCNN. Then, it is extended to DTCNN in a corollary. Another sufficient monotone decreasing condition for $E(t)$ is proved on the worst-case in the second theorem. This condition is stricter than the condition in the first theorem, but it is convenient to be used.

For analyzing the transient variance around the kth updating moment, first, we introduce a definition

Definition 2.1 *Let $\varepsilon \in R$, $\varepsilon > 0$ and ε is small enough. To describe the time round the kth moment, we define*

$$t = kT^- = kT - \varepsilon$$

$$t = kT^+ = kT + \varepsilon$$

Similarly, for the energy function $E(t)$ and the output function $y_i(t)$ at those moments, we define them as follows respectively.

$$E^-(k) = E(t)|_{t=kT^-} \quad E^+(k) = E(t)|_{t=kT^+}$$

$$y_i^-(k) = y_i(t)|_{t=kT^-} \quad y_i^+(k) = y_i(t)|_{t=kT^+}$$

Next, we give a theorem about monotone decreasing property of the energy function of our SCNN with reciprocal or nonreciprocal weight matrix.

Theorem 2.3 *The generalized energy function of a synchronous-updating CNN is monotone decreasing if the relations (2.29) and (2.30) are satisfied.*

(1) when $y_p(k) = -1$ and $y_p(k+1) = 1$, then

$$A_{pp} \geq -\sum_{\substack{i=1 \\ i\neq p}}^{n} A_{ip}\, y_i^+(k+1) \tag{2.29}$$

(2) when $y_p(k) = 1$ and $y_p(k+1) = -1$, then

$$A_{pp} \geq \sum_{\substack{i=1 \\ i \neq p}}^{n} A_{ip}\, y_i^{+}(k+1) \tag{2.30}$$

$$\forall p \in \{1, 2, \cdots, n\}$$

If "$\geq$" in above equations are changed to "$>$", the energy function is strict monotone decreasing, so that oscillation with limited cycle length does not exist in our DTCNN.

Proof: Differing from the proof in original continuous CNNs, our proof is divided into two steps, first, it must be proved that $E(t)$ is monotone decreasing within a updating period, i.e. $t \in (kT, (k+1)T]$ However, because of $\bar{\Phi} = 0$ in ϕ_{T_1}, $v_i(t)$ and $y_i(t) = f[v_i(t)]$ is a constant, $E(t)$ is also retained as a fixed value, in fact, it just require to prove $E(t)$ monotone decreasing in ϕ_{T_1} phase. Second, at the moment of $t = (k+1)T$, the network is updated by the clock signal, it substitutes $y_j(k+1)$ instead of $y_j(k)$ in energy function, we must prove that the value of $E(t)$ will be reduced or remained after undergoing one time of updating. It means that the monotone decreasing of $E(t)$ is kept throughout a series of updating periods until the network converges to its final stable points.

(1). First, during ϕ_{T_2}, the energy function can be written as:

$$E(t) = -\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\, y_i(t)\, y_j(k) + \hat{E}(t) \tag{2.31}$$

where

$$\hat{E}(t) = \frac{1}{2R_x}\sum_{i=1}^{n} y_i^2(t) - \sum_{i=1}^{n}\sum_{j=1}^{n} B_{ij} y_i(t)\, u_j - \sum_{i=1}^{n} I_i\, y_i(t)$$

Then, the differentiation of $E(t)$ with respect to time t is derived from above equations.

$$\begin{aligned}
\frac{dE(t)}{dt} &= \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\frac{dy_i(t)}{dv_i(t)}\cdot\frac{dv_i(t)}{dt}\, y_j(k) + \frac{1}{R_x}\sum_{i=1}^{n}\frac{dy_i(t)}{dv_i(t)}\cdot\frac{dv_i(t)}{dt}\, y_i(t) \\
&\quad -\sum_{i=1}^{n}\sum_{j=1}^{n} B_{ij}\frac{dy_i(t)}{dv_i(t)}\cdot\frac{dv_i(t)}{dt}\cdot u_j - \sum_{i=1}^{n} I_i\frac{dy_i(t)}{dv_i(t)}\cdot\frac{dv_i(t)}{dt} \\
&= -\sum_{i=1}^{n}\frac{dy_i(t)}{dv_i(t)}\cdot\frac{dv_i(t)}{dt}\Big[\sum_{j=1}^{n} A_{ij}\, y_j(k) - \frac{1}{R_x} y_i(t) + \sum_{j=1}^{n} B_{ij}\, u_j + I_i\Big] \\
&= -\sum_{\substack{i=1 \\ |v_i|<1}}^{n}\frac{dv_i(t)}{dt}\Big[\sum_{j=1}^{n} A_{ij}\, y_j(k) - \frac{1}{R_x} v_i(t) + \sum_{j=1}^{n} B_{ij}\, u_j + I_i\Big]
\end{aligned}$$

According to the cell circuit equation (2.10a), it can be derived as follows:

$$\frac{dE(t)}{dt} = - \sum_{\substack{i=1 \\ |v_i|<1}}^{n} C_x \, [\frac{dv_i(t)}{dt}]^2 \; \leq \; 0 \tag{2.32}$$

This conclusion is the same as in the continuous-time CNN[1], but the required constraint condition of $A_{ij} = A(\zeta, \xi; m, n)$ is eliminated here, so that the conclusion (2.32) is suitable to both reciprocal and nonreciprocal synchronous-updating CNN.

(2). As mentioned above, at the updating moment of $t = (k+1)T$, $y_j(k)$ in (2.13) is replaced by $y_{c_j}(k+1)$ so that the energy function may be not continuous at that point, a step change of the value of $E(t)$ may exist at $t = (k+1)T$. It is obvious that if the relation $E^+((k+1)T) \leq E^-((k+1)T)$ is satisfied, then, the monotone decreasing of $E(t)$ through any two sequent updating periods can be guaranteed. In order to do so, we compare $y_j(k+1)$ with $y_j(k)$ on three cases.

i). $\forall p \in \{1, 2, \cdots, n\}$

$$y_p(k+1) = y_p(k)$$

then

$$E^+(k+1) = E^-(k+1)$$

ii). $\exists p \in \{1, 2, \cdots, n\}$

$$y_p(k) = -1, \text{ but } \; y_p(k+1) = 1$$

It means the voltage on the terminal of the capacitor C_y is inverted from -1 to $+1$ in ϕ_{T_1} phase after $(k+1)$th clock signal. We can write $E^-(k+1)$ and $E^+(k+1)$ respectively as follows:

$$E^-(k+1) = [-\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\, y_i(t)\, y_j(k) + \hat{E}(t)\,]|_{t=(k+1)T^-} \tag{2.33}$$

where $\hat{E}(t)$ is the same as that in (2.31).

Since $y_i(t)$ is variable continuously at $t = (k+1)T$, based on the circuit character analyzed previously, it is known that if ε is chosen small enough, $y_i^+(k+1) = y_i^-(k+1)$ and $\hat{E}^+(k+1) = \hat{E}^-(k+1)$. Then, we get

$E^+(k+1)$

$$\begin{aligned}
&= [-\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\, y_i(t)\, y_j(k+1) + \hat{E}(t)]|_{t=(k+1)T^+} \\
&= -\{ [\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\, y_i(t)\, y_j(k) + 2\sum_{i=1}^{n} A_{ip}\, y_i(t)\, y_p(k+1)] + \hat{E}(t) \}|_{t=(k+1)T^+} \\
&= E^-(k+1) - 2\sum_{i=1}^{n} A_{ip}\, y_i(t)\, y_p(k+1)|_{t=(k+1)T^+} \qquad (2.34)
\end{aligned}$$

As it is assumed

$$y_p(k+1) = 1$$

we derive

$$E^+(k+1) = E^-(k+1) - 2\sum_{i=1}^{n} A_{ip}\, y_i(t)|_{t=(k+1)T^+} \qquad (2.35)$$

According to (2.29) and $y_p^+(k+1) = y_p(k+1) = 1$, we have

$$A_{pp} y_p^+(k+1) \geq -\sum_{\substack{i=1\\ i\neq p}}^{n} A_{ip} \cdot y_i(t)|_{t=(k+1)T^+}$$

and

$$\sum_{i=1}^{n} A_{ip} y_i(t)|_{t=(k+1)T^+} \geq 0$$

Substituting it into (2.35), it can be found that, in case of (2.29) being met, we have $E^+(k+1) \leq E^-(k+1)$.

If the strict monotone decreasing condition is satisfied, we have

$$A_{pp} y_p^+(k+1) > -\sum_{\substack{i=1\\ i\neq p}}^{n} A_{ip} \cdot y_i(t)|_{t=(k+1)T^+}$$

and

$$\sum_{i=1}^{n} A_{ip} y_i(t)|_{t=(k+1)T^+} > 0$$

then, $E^+(k+1) < E^-(k+1)$.

iii). Next, it is assumed that

$$y_p(k) = 1, \text{ but } y_p(k+1) = -1$$

Similarly, we can write the energy function at the moment of $t = (k+1)T^+$ as

$$E^+(k+1) = E^-(k+1) + 2\sum_{i=1}^{n} A_{ip}\, y_i(t)|_{t=(k+1)T^+} \tag{2.36}$$

If the following relation is met,

$$2\sum_{i=1}^{n} A_{pi}\, y_i(t)|_{t=(k+1)T^+} \leq 0$$

then, we can get $E^+(k+1) \leq E^-(k+1)$.

Based on (2.30) and $y_p^+(k+1) = y_p(k+1) = -1$, we have

$$A_{pp} y_p^+(k+1) \leq -\sum_{\substack{i=1\\ i\neq p}}^{n} A_{ip}\, y_i(t)|_{t=(k+1)T^+}$$

and

$$\sum_{i=1}^{n} A_{ip}\, y_i(t)|_{t=(k+1)T^+} \leq 0$$

Substituting it into (2.36), it is proved that if the condition (2.30) is satisfied, we have

$$E^+(k+1) \leq E^-(k+1)$$

Similarly, when the strict monotone decreasing condition is met, we have $E^+(k+1) < E^-(k+1)$.

$\square$

Since our discrete-time CNN described with (2.12) is derived with just extracting the values of SCNN at a series of discrete time, $t = kT,\ k = 0, 1, 2, \cdots$, it is believable that all properties of SCNN are kept in our DTCNN, the conclusion of Theorem 2.3 is also suitable to our DTCNN. A corollary can be easily derived.

Corollary 2.1 *If the conditions (2.29) and (2.30) are satisfied, the energy function of discrete-time CNN defined by (2.25) is also monotone decreasing.*

Theorem 2.3 gives a proof for monotone decreasing property of an energy function of SCNN. In the next theorem, we want to give another convergent criterion for nonreciprocal SCNNs and DTCNNs, which is used easily. In order to do so, first, we present a definition about diagonal-column eigendominant.

Definition 2.2 *The weight matrix A is said as diagonal-column eigendominant if the ith diagonal element A_{ii} is greater than or equal to the sum of absolute values of other elements in the ith column. i.e.,*

$$A_{ii} \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} | A_{ji} | \tag{2.37}$$

$$\forall i \in \{1, 2, \cdots, n\}$$

For strict diagonal-column eigendominant, the sign "$\geq$" in above equation is replaced by "$>$".

For a SCNN, the ith diagonal element A_{ii} in its weight matrix A is corresponding to the self-feedback coefficient of a cell $c(i)$, but other elements in the ith column are corresponding to the feedback coefficients from near cells within the neighbor to cell $c(i)$. Therefore, the physical meaning of the diagonal-column eigendominant is that a self-feedback of cell $c(i)$ is stronger than feedbacks from other near cells.

Theorem 2.4 *A SCNN generalized energy function defined as (2.24) is monotone decreasing if it is diagonal-column eigendominant. Thus, this SCNN must be convergent.*

Proof: Within a period $t \in (kT, (k+1)T)$, $k = 0, 1, 2, \cdots$, the proof is the same as that in Theorem 1. Here, we just need to provide a proof of the case $y_i^+((k+1)T) \neq y_i^-((k+1)T)$.

i). Let us assume

$$y_p(k) = -1, \qquad y_p(k+1) = +1$$

then, the equation (2.35) is written again as follows:

$$E^+(k+1) = E^-(k+1) - 2\sum_{i=1}^{n} A_{ip}\, y_i(t)|_{t=(k+1)T+} \tag{2.35}$$

On the base of (2.37), we can obtain

$$\begin{aligned} A_{pp} & \\ \geq & \sum_{\substack{i=1 \\ i\neq p}}^{n} | A_{ip} | = \sum_{\substack{i=1 \\ i\neq p}}^{n} | A_{ip}\, y_i(t) \,||_{t=(k+1)T^+} \\ \geq & \;| \sum_{\substack{i=1 \\ i\neq p}}^{n} A_{ip}\, y_i(t) \,||_{t=(k+1)T^+} \geq - \sum_{\substack{i=1 \\ i\neq p}}^{n} A_{ip}\, y_i(t)|_{t=(k+1)T^+} \end{aligned}$$

thus

$$\sum_{i=1}^{n} A_{ip}\, y_i(t)|_{t=(k+1)T^+} \geq 0 \tag{2.38}$$

Substituting (2.38) into (2.35), we have

$$E^+(k+1) \leq E^-(k+1)$$

ii). Next, we assume

$$y_p(k) = +1, \text{ but } y_p(k+1) = -1$$

then, the equation (2.36) is rewritten as follows:

$$E^+(k+1) = E^-(k+1) + 2\sum_{i=1}^{n} A_{ip}\, y_i(t)|_{t=(k+1)T^+} \tag{2.36}$$

According to (2.37), we have

$$\begin{aligned} A_{pp} & \geq | \sum_{\substack{i=1 \\ i\neq p}}^{n} A_{ip}\, y_i(t) \,|_{t=(k+1)T^+} \\ & \geq \sum_{\substack{i=1 \\ i\neq p}}^{n} A_{ip}\, y_i(t)|_{t=(k+1)T^+} \end{aligned}$$

Since $y_p(t)|_{t=(k+1)T^+} = -1$, then

$$\sum_{i=1}^{n} A_{ip}\, y_p(t)|_{t=(k+1)T^+} \leq 0 \tag{2.39}$$

Substituting (2.39) into (2.36), we get

$$E^{+}(k+1) \leq E^{-}(k+1)$$

In the case of the strict diagonal-column eigendominant, the relation $E^{+}(k+1) < E^{-}(k+1)$ is available.

□

The theorem has been proved. Similarly with Corollary 2.1, here, we can derive following corollary.

Corollary 2.2 *If a nonreciprocal discrete-time CNN satisfies the condition of the diagonal-column eigendominant defined as (2.37), its energy function is monotone decreasing and the network is convergent.*

Theorem 2.4 is suitable to general case including nonreciprocal and reciprocal SCNN. Specially, if the network is reciprocal, i.e., $A_{ij} = A_{ji}$, we can obtain Corollary 2.3 as follows:

Corollary 2.3 *The generalized energy function of a SCNN with reciprocal weight coefficients is monotone decreasing if the next relation is satisfied.*

$$A_{ii} \ \geq \ \sum_{\substack{j=1 \\ j\neq i}}^{n} | A_{ij} | \qquad \forall i \in \{1, 2, \cdots, n\} \tag{2.40}$$

In this case, similar with the column-diagonal eigendominant, the condition (2.40) can be also called as row-diagonal eigendominant. It means that within a weight matrix A, the ith diagonal element A_{ii} is greater than or equal to the sum of absolute values of other elements in the ith row. In fact, the condition (2.40) is similar with that presented by N.Fruehauf, L.O.Chua and E.Lueder in [4]. Here, we can find that this condition is a special result of Theorem 2.4 for the reciprocal SCNN and DTCNN just only.

2.6 Conclusion

In this chapter, first, we showed the cell model of the continuous-time CNN, and some typical types of 2-D array structures briefly. After introducing a two phases synchronous-updating signal into a continuous-time CNN, we obtained a synchronous-updating CNN, we called it as SCNN. By sampling the values of state variations v_i and output variations y_i at the updating moments $t = kT,\ k = 0, 1, 2.\cdots$, we derived a discrete-time CNN which topology and output function are distinct from the DTCNN presented by Harrer and Nossek. In general, the output of this DTCNN is a variable value during $(-1,\ +1)$, so that it can be used to image processing in which the output is a multiple grey level image. in order to guarantee the output as a binary value to meet some special applications, a sufficient condition and a necessary condition are presented here, which provide the design requirement for the matrix A and the matrix B. Moreover, in order to analyze convergence condition of this DTCNN, the generalized energy functions for our SCNN and DTCNN are defined respectively. Here, we don't directly compare the value of the energy function of DTCNN at two sequent of updating moments, which method is used by N.Fruehauf, L.O.Chua and E.Lueder [4] for reciprocal DTCNN with the same output function. We analyze the energy function of SCNN during a clock period and around a updating moment, because the energy function is not continuous at those moments, which impact must be considered carefully. Two theorems about the convergence condition of nonreciprocal and nonuniform SCNN are described first. Meanwhile, since the energy function of DTCNN is sampled and discreted from that of SCNN, two convergence conditions are also available to nonreciprocal and nonuniform DTCNN. The result covers the reciprocal DTCNN as a special case, and provide the potential to apply our DTCNN more widely, for examples, associative memories, multiple visual pattern recognitions and others.

References

[1] L.O.Chua and L.Yang, "Cellular neural network: theory", *IEEE Trans. Circ. S yst.*, vol.CAS-35, pp.1257-1272, 1988.

[2] H.Harrer and J.T.Nossek, "Discrete-time cellular neural networks", *Int. J. Circ. Theor. Appl.*, vol.20, pp.453-467, 1992.

[3] C. He and A.Ushida, "Convergence analysis of synchronous-updating CNN and related DTCNN", *Proc. of 1993 Int. Symp. on Nonlinear Theor. and its Appl.*, pp.29-34, Hawaii, 1993.

[4] N.Fruehauf, L.O.Chua and E.Lueder, "Convergence of reciprocal time-discrete cellular neural networks with continuous nonlinearities", *Proc. 1992 IEEE Int. Workshop on CNNA*, pp.106-111, 1992.

Chapter 3

A Modified Tracing Curve Algorithm for CNN

3.1 Introduction

Owing to the piecewise linear character of the non-linearities, cellular neural networks depend crucially their nonlinear dynamics. Proper operation often requires the existence of multiple equilibrium points or DC operating points. Therefore, it is important to have an efficient analysis method for obtaining a global picture of the dynamic behavior, the equilibrium pattern and the basins of attraction in a given network. It is a problem to find equilibrium points in CNN described by the equation (2.3), it amounts to solving a set of piecewise linear equations

$$-\Lambda \mathbf{v}(t) + A\text{sat}(\mathbf{v}(t)) + B\mathbf{u} + I = 0 \tag{3.1}$$

The Newton-Raphson iteration method is a general tool to be used for nonlinear algebraic equations, but Newton method converges only in those cases where an initial guess is a pretty exact approximation to the actual solution. In order to widen convergence region of iteration methods, the continuation method is applied to solve homotopy equation. This method is designed to solve a system of n nonlinear algebraic equations with $n+1$ variables.

$$F(\mathbf{x}) = 0, \quad F:\ D \subset R^{n+1} \to R, \quad \mathbf{x} \in R^{n+1} \tag{3.2}$$

Some algorithm[1 - 4] have been proposed to solve equation (3.2). The predictor-corrector tracing curve algorithm in Ref. [1] may be one of the most effective algorithms between them, where the implicit backward-differentiation formula was used, so called the BDF algorithm. In this section, first, we present an modified BDF algorithm. Its main feature is bellow:

1. At the jth step of curve tracing algorithm, a guess value $\mathbf{x}^p(s_j)$ is predicted, where the kth order Hermite polynomial[5] is used whose coefficients are determined by $(k+1)/2$ known functional values and their differentiations. The Hermitor formula is used to extrapolate and produce the predictor $\mathbf{x}^p(s_j)$.
2. After the $\mathbf{x}^p(s_j)$ is obtained, the Brown iteration method[8] is used to solve augmented equations. With this method, the total numbers of subfunction evaluations are only half of that in classical Newton-Raphson method, so that the evaluation times of nonlinear functions are reduced efficiently.
3. We could get the curvature and the norm after $\mathbf{x}^p(s_j)$ is obtained. Then the next step size could be determined based on the norm. At the neighborhood of the sharp turning points on the solution curve, the step could be reduced in time so that some iterations are avoided which may result in failure. At the smooth part of the solution curve, we can get more large step size. In this way, our algorithm get more better stability and efficiency.

3.2 Predictor algorithm

At first, we introduce parameter s and describe variables $\mathbf{x}$ in equation (3.2) as function $x_i(s)$, $i = 1, 2, ..., n+1$. If $x_i(s)$ is continuously differentiable, we can get

$$\begin{aligned} F(\mathbf{x}) &= 0 \\ (ds)^2 &= (dx_1)^2 + (dx_2)^2 + \cdots + (dx_{n+1})^2 \end{aligned}$$

where s is an arc-length of the solution curve when it stretches from a starting point along a direction. It could be described as:

$$G(\mathbf{x}) = 0, \quad G:\ D \subset R^{n+1} \to R^{n+1}, \mathbf{x} \in R^{n+1} \tag{3.3}$$

where s is an implicit parameter. We can trace the solution curve from $s = 0$ along the positive direction and get half of the solution curve. Another half of the solution curve can be obtained when it be traced along negative direction. Of course, if it is a close curve, we can trace the total curve along either position and direction. Every tracing step consists of two stages which are called as predictor and corrector respectively.

With the algorithm suggested in Ref. [1], when the curve is traced from $s = s_{j-1}$ to $s = s_j$, the kth order Lagrange polynomial is used to predict an initial point for the corrector iteration. Obviously, the predictor is given more precisely, the iteration in the corrector step converge quickly. Thus, we propose a technique for finding the precise predictor based on Hermite extrapolation.

According to remainder theory of Lagrange polynomial, once $k+1$ points $s_{j-1}, s_{j-2}, \cdots, s_{j-k-1} \subset (a,b)$ are selected to formulate a Lagrange polynomial $L_k(s)$ which approximates $x_i(s)$ in range (a,b), the cutoff error $R_{L_k}(s) = x_i(s) - L_k(s)$ is

$$R_{L_k}(s) = \frac{x_i^{(k+1)}(\xi)}{(k+1)!} W_{k+1}(s) \tag{3.4}$$

where

$$\xi \in (a, b)$$
$$W_{k+1}(s) = (s - s_{j-1})((s - s_{j-2}) \cdots (s - s_{j-k-1})$$

We can set

$$\sup_{a \le s \le b} |x_i^{(k+1)}(s)| = M_{k+1}$$

Then, the upper bounded the remainder of the kth order Langrange polynomial is

$$\sup_{a \le s \le b} |R_{L_k}(s)| = \frac{M_{k+1}}{(k+1)!} |W_{k+1}(s)| \tag{3.5}$$

This value relates to $|W_{k+1}(s)|$ when M_{k+1} has been determined. Because we use this Langrange polynomial to extrapolate $x_i(s_i)$ at $s = s_j$, $|W_{k+1}(s)|$ generally is larger. In order to reduce the error of the predictor, $(k+1)/2$ known function values $x_i(s_{j-1}), x_i(s_{j-2}), \cdots, x_i(s_{j-\frac{k+1}{2}})$ and their differentials $\dot{x}_i(s_{j-1}), \dot{x}_i(s_{j-2}), \cdots, \dot{x}_i(s_{j-\frac{k+1}{2}})$ are selected, where $s_{j-1}, s_{j-2}, \cdots, s_{j-\frac{k+1}{2}}$ locate in side of the range (a, b) near the point s_j, and formulate a kth order Hermite polynomial $H_k(s)$ to approximate $x_i(s)$. Then, the remainder is given by

$$R_{h_k}(s) = x_i(s) - H_k(s) = \frac{x_i^{(k+1)}(\xi)}{(k+1)!} W^2_{(k+1)/2}(s) \tag{3.6}$$

where

$$\begin{aligned}
&\xi \in (s_{j-(k+1)/2},\ s_{j-1}) \\
&W_{(k+1)/2}(s) \ = \ (s - s_{j-1})(s - s_{j-2}) \cdots (s - s_{j-(k+1)/2})
\end{aligned}$$

Because

$$(s_{j-(k+1)/2},\ s_{j-1}) \in (s_{j-k-1}, s_{j-1})$$

there exists

$$\sup_{s_{j-(k+1)/2} \le s \le s_{j-1}} |x_i^{(k+1)}(s)| \le \sup_{s_{j-k-1} \le s \le s_{j-1}} |x_i^{(k+1)}(s)| \le M_{k+1}$$

moreover, because

$$(s - s_{j-1}) \ < \ (s - s_{j-2}) \ < \cdots < \ (s - s_{j-k}) \ < \ (s - s_{j-k-1})$$

so that

$$W^2_{(k+1)/2}(s) \ < \ W_{k+1}(s)$$

and

$$|R_{h_k}(s)| \le |R_{L_k}(s)|$$

Thus, we can derive

Theorem 3.1 *If we select $(k+1)/2$ known function values $x_i(s_{j-1}), x_i(s_{j-2}), \cdots, x_i(s_{j-\frac{k+1}{2}})$ and their differentials $\dot{x}_i(s_{j-1}), \dot{x}_i(s_{j-2}), \cdots, \dot{x}_i(s_{j-\frac{k+1}{2}})$ to formulate a kth order Hermite polynomial $H_k(s)$ for approximating $x_i(s)$, and where $s_{j-1}, s_{j-2}, \cdots, s_{j-\frac{k+1}{2}}$ locate in side of the range (a, b) near the point s_j, it can be proved*

$$|R_{h_k}(s)| \le |R_{L_k}(s)| \tag{3.7}$$

Although this conclusion comes froms interpolations, it can be applied to the extrapolations for small $\triangle s$. So, we can get conclusion that the precision of kth order Hermite polynomial which is used to extrapolate for getting the predictor $x_i^p(s_i)$ is higher than the precision of the kth order Lagrange polynomial.

The coefficients of the predictor formula for $x_i^p(s_j)$ could be obtained from the equation bellow:

$$\begin{bmatrix} 0 & 1 & 2s_{j-1} & \cdots & ks_{j-1}^{k-1} \\ 0 & 1 & 2s_{j-2} & \cdots & ks_{j-2}^{k-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 2s_{j-(k+1)/2} & \cdots & ks_{j-(k+1)/2}^{k-1} \\ 1 & s_{j-1} & s_{j-1}^2 & \cdots & s_{j-1}^k \\ 1 & s_{j-2} & s_{j-2}^2 & \cdots & s_{j-2}^k \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & s_{j-(k+1)/2} & s_{j-(k+1)/2}^2 & \cdots & s_{j-(k+1)/2}^k \end{bmatrix} \begin{bmatrix} a_0^p \\ a_1^p \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ a_k^p \end{bmatrix} = \begin{bmatrix} \dot{x}_i^{(j-1)} \\ \dot{x}_i^{(j-2)} \\ \vdots \\ \dot{x}_i^{(j-(k+1)/2} \\ x_i^{(j-1)} \\ x_i^{(j-2)} \\ \vdots \\ x_i^{(j-(k+1)/2} \end{bmatrix} \tag{3.8}$$

We can express it as the matrix:

$$\mathbf{s}_k^p \mathbf{a}_k^p = (\dot{\mathbf{x}}_k^p \mathbf{x}_k^p)^T \tag{3.9}$$

thus, we have

$$\begin{aligned} x_i^p(s) &= (1\ s\ s^2\ \cdots s^k) \cdot \mathbf{a}_k^p \\ &= ((1\ s\ s^2\ \cdots s^k)(\mathbf{s}_k^p)^{-1}(b\dot{f}x_p^p\dot{\mathbf{x}}_p^p)^T \end{aligned}$$

It can be rearranged as

$$\begin{aligned} x_k^p(s) &= \sum_{l=j-(k+1)/2}^{j-1} h_l(s)[(s_l - s)(2a_l^p x_i^{(l)} - \dot{x}_i^{(l)}) + x_i^{(l)}] \\ &= \sum_{l=j-(k+1)/2}^{j-1} \{h_l(s) \cdot [1 + 2a_l^p(s_l - s)] \times x_i^{(l)} + h_l(s)(s - s_l)\dot{x}_i^{(l)}\} \\ &= \sum_{l=j-(k+1)/2}^{j-1} [p1_l(s)x_i^{(l)} + p2_l(s)\dot{x}_i^{(l)}] \end{aligned} \tag{3.10}$$

where

$$h_l(s) = \prod_{\substack{i=j-(k+1)/2 \\ i \neq j}} [(s - s_l)/(s_l - s_i)]^2$$

$$a_l = \sum_{\substack{i=j-(k+1)/2 \\ i \neq j}} \frac{1}{s_l - s_i}$$

3.3 Corrector algorithm

We use $\mathbf{x}^p(s_j)$ as an initial point to iterate and solve equations bellow:

$$\begin{aligned} F(\mathbf{x}(s_j)) &= 0 && (3.11a)\\ \sum_{i=1}^{n+1} \dot{x}_i(s_j) &= 1 && (3.11b)\end{aligned}$$

If $\dot{x}_i(s_j)$ can not be obtain by differentiating $F(\mathbf{x}(s_j))$ respect to s, we can construct the Hermite interpolation polynomial

$$x_i^c(s) = a_0^c + a_1^c + a_2^c s^2 + \cdots + a_k^c s^k$$

and make differentiation by s. In this case, we have the following equation.

$$\begin{bmatrix} 1 & s_j & s_j^2 & \cdots & s_j^k \\ 1 & s_{j-1} & s_{j-1}^2 & \cdots & s_{j-1}^k \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & s_{j-(k+1)/2} & s_{j-(k+1)/2}^2 & \cdots & s_{j-(k+1)/2}^k \\ 0 & 1 & 2s_{j-1} & \cdots & ks_{j-1}^k \\ 0 & 1 & 2s_{j-2} & \cdots & ks_{j-2}^k \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 2s_{j-(k+1)/2+1} & \cdots & ks_{j-(k+1)/2+1}^k \end{bmatrix} \begin{bmatrix} a_0^c \\ a_1^c \\ \vdots \\ \vdots \\ a_k^c \end{bmatrix} = \begin{bmatrix} x_i^{(j)} \\ x_i^{(j-1)} \\ \vdots \\ x_i^{(j-(k+1)/2)} \\ \dot{x}_i^{(j-1)} \\ \dot{x}_i^{(j-2)} \\ \vdots \\ \dot{x}_i^{(j-(k+1)/2+1)} \end{bmatrix} \qquad (3.12)$$

This equation can be written as follows

$$\mathbf{s}_k^c \mathbf{a}_k^c = (\dot{\mathbf{x}}_k^c \mathbf{x}_k^c)^T \qquad (3.13)$$

We have

$$\begin{aligned} x_i^c s &= (1\ s\ s^2 \cdots s^k) \cdot \mathbf{a}_k^c \\ &= (1\ s\ s^2 \cdots s^k)(\mathbf{s}_k^c)^{-1}(\dot{\mathbf{x}}_k^c \mathbf{x}_k^c)^T \\ &= c1_j(s)x_i^{(j)} + c1_{j-1}(s)x_i^{(j-1)} + \cdots + c1_{j-(k+1)/2}(s)x_i^{(j-(k+1)/2)} + c2_{j-1}(s)\dot{x}_i^{(j-1)} \\ &\quad + c2_{j-2}(s)\dot{x}_i^{(j-2)} + \cdots + c2_{j-(k+1)/2+1}(s)\dot{x}_i^{(j-(k+1)/2+1)} \end{aligned} \qquad (3.14)$$

In this case, the corrector formula is given by

$$F(\mathbf{x}(s_j)) = 0 \tag{3.15a}$$
$$\sum_{i=1}^{n+1}[\dot{c}1_j(s_j)\cdot x_i^{(j)} + Q_i]^2 - 1 = 0 \tag{3.15b}$$

where

$$Q_i = \sum_{l=j-1}^{j-(k+1)/2} \dot{c}1_l x_i^{(l)} + \sum_{l=j-1}^{j-(k+1)/2} \dot{c}2_l \dot{x}_i^{(l)}$$

Brown method[8] is applied to solve this set of equation as (11) or (15). First, every subfunction is expressed as taylor series, then, it is linearized and the variables is eliminated one by one until the linearized system of equations is transformed into a triangular system of equation. For per iteration step of Brown method, in every iteration step of (3.2) which consists of $n+1$ functions, $(n+1)(n+4)/2$ numbers of subfunctions should be evaluated, but with Newton-Raphson method, $(n+2)(n+1)$ umbers of subfunctions evaluations is needed. When subfunction calculations are more complex, Brown method can reduce cost of calculation. Moreover, the evaluation times of subfunction $g_1(\mathbf{x})$ is $(n+2)$ in per iteration, the evaluation $g_2(\mathbf{x})$ is $(n+1)$ which is less than that of $g_1(\mathbf{x})$, the evaluation times of another functions are reduced gradually in proper order. So, we can rearrange the linear equation in simultaneous equations (3.3) as the first equation, but nonlinear equations are rearranged as the latter, the nonlinear equation which calculation is most complex is rearranged as the last. In this way, actual calculation for nonlinear functions is reduced once more. This method is suitable especially for analysis of a nonlinear network in which only a few elements are nonlinear but most of elements are linear.

3.4 Choice of the step sizes

In curve tracing algorithm, one of the crucial problems is the choice of the step sizes. In Ref. [1], the next step size is predicted based on previous step sizes, if the predicted step size is not suitable so that the difference of the first iteration corrector $\mathbf{x}^{c(1)}$ and the predictor $\mathbf{x}^p$, $|\mathbf{x}^{c(1)} - \mathbf{x}^p|$ is over the upper limit δ_{max}, then, this step size should be reduced and the first corrector should be carried again. In our algorithm, the curvature of the solution curve is used as a parameter to control next step size. In the neighborhood of a sharp turning point on a solution curve, its curvature must get more larger, then we can choose a small step size to prevent the iteration failure. In smooth parts of the solution curve, their curvature should get less, so we can obtain more larger step size. In this way, we can obtain a robust and efficient curve tracing algorithm which gives more satisfactory results n practical numerical calculation.

After the jth curve tracing step, we define

$$d\dot{x}_i(s_j)/ds = \dot{x}_i(s_{j-1})/(s_j - s_{j-1}) \tag{3.16}$$

and

$$\sigma^{(i)} = \sum_{i=1}^{n+1} |d\dot{x}_i(s_j)/ds| \tag{3.17}$$

$\sigma^{(i)}$ describes the sum of the curvature variations of the solution curves $x_i(s)$ $i = 1, 2, .., n+1$ at the neighborhood of $s = s_j$. Considering the stability of the calculation, we further introduce averages of the curvatures and step sizes as follows.

$$AV^{(j)} = (\sigma^{(j)} + \sigma^{(j-1)} + .. + \sigma^{(j-(k+1)/2)})\frac{2}{k+1} \tag{3.18a}$$

$$\bar{h}_j = (h_j + h_{j-1} + \cdots + h_{j-(k+1)/2})\frac{2}{k+1} \tag{3.18b}$$

Based on $\sigma^{(j)}, AV^{(j)}$ and $\bar{h}_j$, we determine the h_{j+1}

$$h_{j+1} = \begin{cases} \beta_1 \bar{h}_j & \sigma^{(j)} < (1-\alpha_1)AV^{(i)} \\ \bar{h}_j & (1-\alpha_1)AV^{(i)} < \sigma^{(j)} < (1+\alpha_2)AV^{(i)} \\ \beta_2 \bar{h}_j & (1+\alpha_2)AV^{(i)} < \sigma^{(j)} \end{cases} \tag{3.18c}$$

The value of the a_i and β_i are relating to the solved equations, they can be modulated in calculating procedure based on the calculated results.

3.5 Computational Algorithm

At the first, we set the parameters as follows.

k: the order of BDF, because $(k+1)/2$ points should be used in Hermite polynomial, the number of $k+1$ must be an even.

H_{max}: maximal length of curve tracing step.

H_{min}: minimal length of curve tracing step.

δ_{max}: maximal difference between a predictor $\mathbf{x}^p(s_j)$ and the first corrector $\mathbf{x}^{c(1)}(s_j)$ as

$$\delta_{max} > |\mathbf{x}^{c(1)}(s_j) - \mathbf{x}^p(s_j)|$$

α_1, α_2, β_1, β_2: a group of parameter to be used to control next curve tracing step size.

ε_1, η: when $|G(\mathbf{x})| < \varepsilon_1$ or $|\mathbf{x}^{c(k+1)} - \mathbf{x}^{c(k)}|, 10^{-\eta}$, the iteration at that point is successful.

N_{max}: maximal number of corrector times.

Calculation procedure:

[step 1] Input initial parameters: $\mathbf{x}^0$, s_0 and H^0 so that

$$G(\mathbf{x}^0) = 0, \qquad H_{min} < H_0 < H_{max}$$

[step 2] Set $j = 1$.

[step 3] When $j = 1$, $x_i^p = x_i^0$, $i = 1, 2, \cdots, n-1, n$, $x_{n+1}^p = x_{n+1}^0 + H_0$, go to step 6.

[step 4] When $j \neq 1$, the coefficients of kth order Hermite polynomial, $pl_i(s)$ and $p2_i(s)$, $i = 1, 2, \cdots, j-(k+1)/2$ are produced according to the formula (3.10).

[step 5] Find

$$x_i^p(s) = \sum_{l=j-(k+1)/2}^{j-1} [pl_l(s)x_i^{(l)} + p2_l(s)\dot{x}_i^{(l)}]$$
$$i = 1, 2, \cdots, n+1$$

[step 6] If $\dot{\mathbf{x}}$ can be derived directely, go to the next step; otherwise, find $cl_l(s)$ and $c2_l(s)$, $l = j-1,\ j-2, \cdots,\ j = (k+1)/2$, according to the formula (3.13).

[step 7] take $\mathbf{x}^p(s_j)$ as an initial point, Brown method is applied to solve $G(\mathbf{x}) = 0$ once so that $\mathbf{x}^{c(1)}(s_j)$ is obtained.

[step 8] Find $DB = |\mathbf{x}^{c(1)}(s_j) - \mathbf{x}^p(s_j)|$, if $DB < \delta_{max}$, go to step 10, else, go to next step.

[step 9]

$$\begin{aligned} s'_j &= (s_j - s_{j-1}) \cdot \frac{DB}{\delta_{max}} + s_{j-1} \\ x_i^p(s'_j) &= (x_i^p(s_j) - x_i(s_{j-1})) \cdot \frac{DB}{\delta_{max}} + x_i(s_{j-1}) \\ & i = 1, 2, \cdots, n+1 \end{aligned}$$

set $s_j = s_j$, go to step 6.

[step 10] Use last corrector result $\mathbf{x}^{(k-1)}(s_j)$, $k = 2, 3, \cdots$ and coefficients $\dot{c}1_l(s)$, $\dot{c}2_l(s)$, to set up the equation:

$$\begin{aligned} & g_{n+1}(\mathbf{x}^{(k-1)}(s_j)) \\ & = \sum_{i=1}^{n+1} (\dot{c}1_j(s_j) \cdot x_i^{(k-1)}(s_j) + Q_i)^2 - 1 \\ & = 0 \end{aligned}$$

[step 11] Carry the kth iteration for the equation $G(\mathbf{x}) = 0$.

[step 12] If $|G(\mathbf{x}^{(k)}(s_j))| < \varepsilon_1$ or $|\mathbf{x}^{(k-1)}(s_j) - \mathbf{x}^{(k)}(s_j)| < 10^{\eta}$, go to next step; else, go to step 10.

[step 13] Find $\sigma^{(j)}$

$$\sigma^{(j)} = \sum_{i=1}^{n+1} |\frac{d\dot{x}_i}{ds}|$$

[step 14] Find $AV^{(j)}$ and $\hat{h}_j$,

$$\begin{aligned} AV^{(j)} &= \frac{2}{k+1} \sum_{l=j}^{j-(k+1)/2} \sigma^{(l)} \\ \hat{h}_j &= \frac{2}{k+1} \sum_{l=j}^{j-(k+1)/2} (s_l - s_{l-1}) \end{aligned}$$

[step 15] Determine h_{j+1} according to the formula (18).

[step 16] If a homotopy equation is solved, when $x_{n+1} = \lambda = 1$, one solution is found, if all solutions are found, the curve tracing procedure terminates; else, go to step 3 to carry on.

3.6 Illustration examples

Three examples are demonstrated here. With the first example to derive multiple solutions for a set of equations, we show the effective of this algorithm. Then, we analyze the DC operating points for a 6-orders Hopfield network and obtain its multiple solutions successfully. Finally, a example is shown to apply our modified curve tracing algorithm to CNN, some results are gotten.

Example 1:

In this example, a system of 10 equations should be solved.

$$g_k(\mathbf{x}) \;=\; x_k - e^{\cos(k \sum_{i=1}^{10} x_i)} \;=\; 0 \quad k \;=\; 1,\; 2,..,10$$

After introducing an additional parameter, we can set up the homotopy equations:

$$g_k(\mathbf{x}, \lambda) \;=\; x_k - \lambda e^{\cos(k \sum_{i=1}^{10} x_i)} \;=\; 0 \quad k \;=\; 1,\; 2,..,10$$

The solution curve is shown in Figure 3.1.

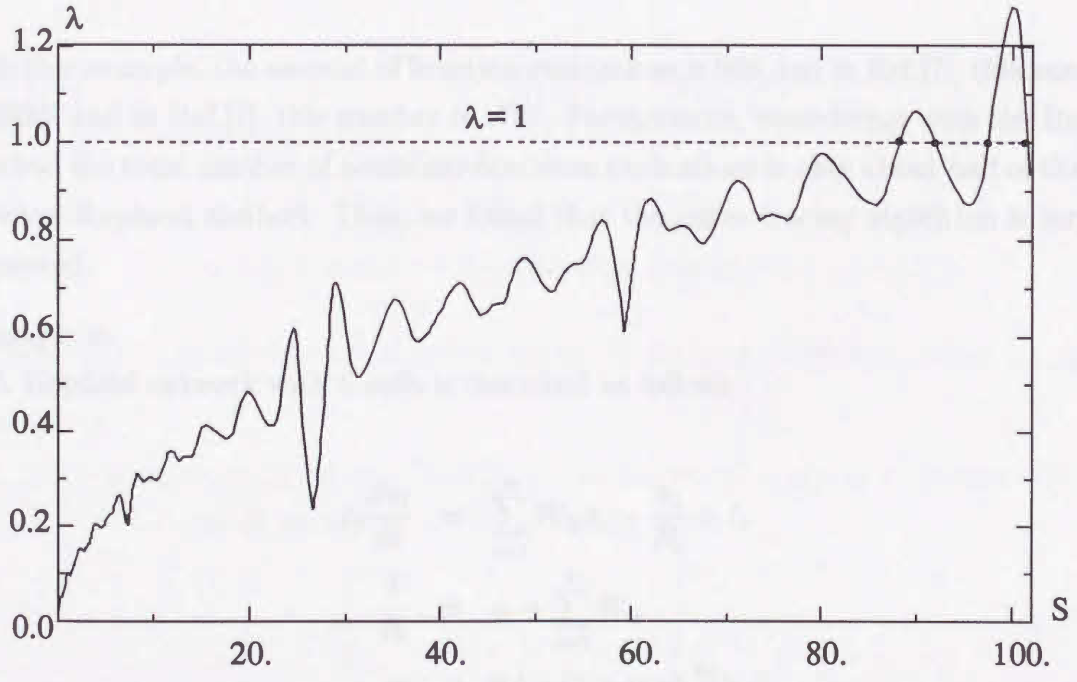


Figure 3.1: Solution curve with our algorithm

Table 3.1 describes some points near the most sharp turning point on the solution curve. The minimum observed step size is 0.00904 which is at $s = 58.43885$. The maximum step size is 0.44303 at $s = 61.84822$.

No. of step	s	step size	No. of iteration	No. of modulated step size	λ	$\sum_{i=1}^{11} \lvert d\dot{x}_i/ds \rvert$
329	56.7222	1	0	.42454	.8196	1.252
331	57.2869	1	0	.26069	.7620	1.665
333	58.0191	1	0	.40250	.6596	1.355
334	58.3170	1	0	.29788	.6170	.9956
335	58.3610	1	1	.04406	.6117	.4824
336	58.4389	10	1	.07783	.6152	61.05
337	58.4479	2	0	.00904	.6167	.9227
339	58.5453	2	0	.06810	.6340	1.382
342	58.7531	3	0	.09353	.6690	1.258
346	59.1476	1	0	.10032	.7324	1.545
351	59.6430	1	0	.13363	.8030	1.600
360	61.8482	1	0	.44303	.8635	1.010

Table 3.1: Some calculated data for Example 1

In this example, the amount of function evaluations is 966, but in Ref.[7], this number is 5936, and in Ref.[1], this number is 1787. Furthermore, considering with the Brown method the total number of nonlinear functions evaluations is only about half of that in Newton-Raphson method. Thus, we found that the curve tracing algorithm is largely improved.

Example 2:

A Hopfield network with 6 cells is described as follows

$$\begin{aligned} c_i\frac{du_i}{dt} &= \sum_{i=1}^{n} W_{ij}x_i - \frac{u_i}{R_i} + I_i \\ \frac{1}{R_i} &= g_i + \sum_{i=1}^{n} W_{ij} \\ x_i &= 0.5 * (1 + \tanh\frac{u_i}{a}) \end{aligned}$$

where, the parameters are selected as

$$W = \begin{bmatrix} 0 & 1 & -2 & -2 & -2 & -2 \\ 1 & 0 & -2 & -2 & -2 & -2 \\ -2 & -2 & 0 & -2 & -2 & -2 \\ -2 & -2 & -2 & 0 & -2 & -2 \\ -2 & -2 & -2 & -2 & 0 & 1 \\ -2 & -2 & -2 & -2 & 1 & 0 \end{bmatrix} \quad I = \begin{bmatrix} 3.5 \\ 3.5 \\ 5. \\ 5. \\ 3.5 \\ 3.5 \end{bmatrix} \quad g = \begin{bmatrix} 8 \\ 8 \\ 11 \\ 11 \\ 8 \\ 8 \end{bmatrix}$$

Assume $a = 0.1$, beginning from a initial point $(\mathbf{u}, \lambda) = (0.1, 0.1, -0.1, 0.1, -0.1, -0.1, 0.0)$, the solution curve is traced in 322 step. In this case, 9 solutions are found together. It is shown in Figure 3.2.

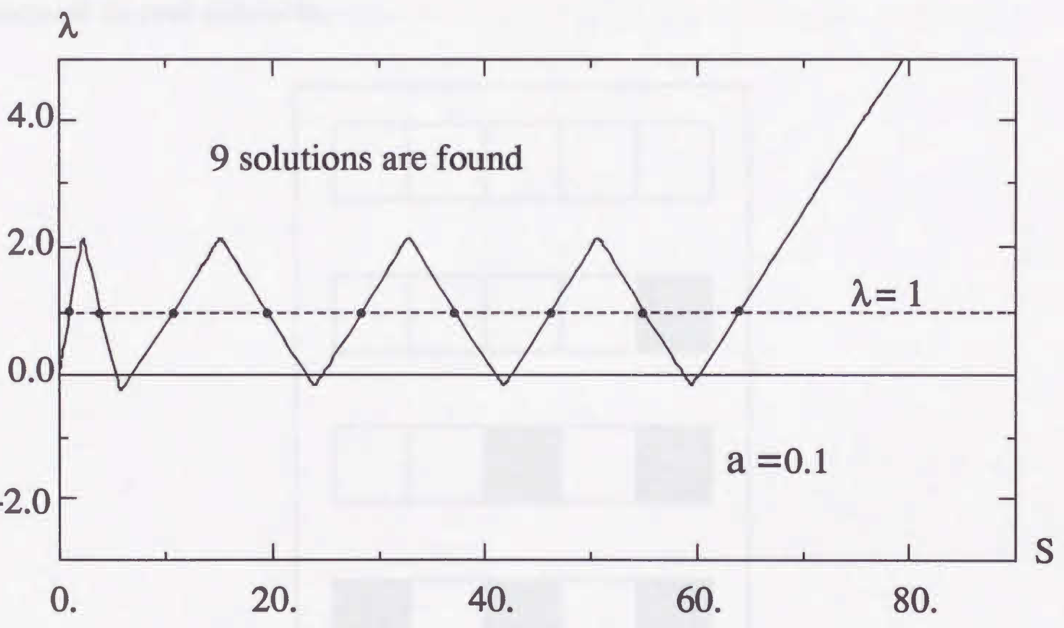


Figure 3.2: Solution curve for 6-cells Hopfield net. in a=0.1

Beginning from different initial points, others equilibrium points can also be searched.

Example 3:

In this example, we apply the algorithm to the connected component detector CNN[9] with n=5. The system equations are described as follows

$$\begin{aligned} C\frac{d\mathbf{v}}{dt} &= -\Lambda\mathbf{v}(t) + A\mathbf{y}(t) + B\mathbf{u} + I \\ \mathbf{y}(t) &= sat(\mathbf{v}(t)) \end{aligned}$$

The matrix A, B and I are composed by the templates

$$T_A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix} \quad T_B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad T_I = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

The external input $\mathbf{u}$ is euqal to 0. This CNN detects the number of connected black components on a white background(or vice versa). The algorithm is applied to obtain total 20 solutions from different start point tracing the solution curves. Between them, only four solutions shown in Figure 3.3 are stable and correspond to real convergent points in CNN. The remainder are corresponding to unstable equilibrium points, since there exists an component at least which absolute value is less than 1, they are non-measurable in real networks.

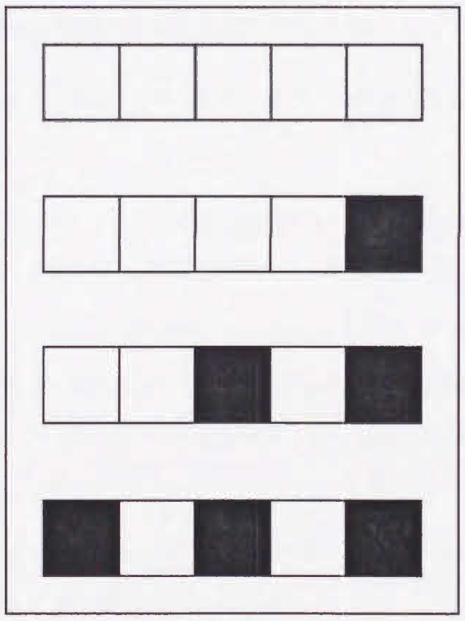


Figure 3.3: Four convergent states for CCD with CNN

3.7 Conclusion

In this chapter, we present a modified BDF curve tracing method. The result shows this algorithm could be used efficiently to trace those solution curve with some sharp turning points. Specially, we want to point out that the Brown method is a kind of the Gauss-Seidel algorithm to be used for nonlinear algebraic functions. It is known that the convergence ratio is second order near to the solution. Furthermore, a number of the function evaluations is $(N^2 + 3N)/2$ when the function consists of N functions. Observe that the Newton method takes N^2 evaluations of the partial derivatives and N evaluations of functions. Thus, the Brown method is efficiently applied to trace solution curve, such that the approximate solution is obtained by Hermite polynomial.

The algorithm presented here can be useful in the analysis of neural networks, e.g. during the design of templates for cellular neural networks. It can be applied to large networks provided that the extreme sparity and the structure of the coefficients are exploited. The method can be applied for some types of neurons with smooth non-linear output functions or piecewise linear output functions. In general, there does not seem to be much hope for an efficient way to find all equilibrium points in a given neural network unless appropriate guidelines are followed during the synthesis process.

References

[1] A.Ushida and L.O.Chua, "Tracing solution curves of nonlinear equations with sharp turning points", *Inter. J. Cir. Theor. and Appl.*, vol.12, pp.1-12, 1984.

[2] K.Yamamura, H.Kubo and K.Horiuchi, "A globally convergent algorithm based on fixed-point homotopy for the solution of nonlinear resistive circuits", *Proc. IEICE*, vol.J70-A, no.10, pp.1430-1438, 1987.

[3] T.Takase, S.Oishi, H.Io and K.Yamamura, "Simplicial fixed points algorithms for finding several solutions of nonlinear circuits", *Proc. IEICE*, vol.J66-A, no.11, pp.112-1129, 1983.

[4] L.V.Kolve, "A quasi-Newton algorithm for following solution curves", *Inter. J. Cir. Theor. and Appl.*, vol.15, pp.181-188, 1987.

[5] K.S.Chao and P.Saeks, "Contribution method in circuit analysis", *Proc. IEEE*, vol.65, pp.1187-1194, 1977.

[6] L.O.Chua and A.Ushida, "A switching-parameter algorithm for finding multiple solutions of nonlinear resistive networks", *Inter. J. Cir. Theor. and Appl.*, vol.4, pp.215-237, 1976.

[7] K.Georg, "On tracing an implicit defined curve by quasi-Newton steps and calculating bifurcation by local perturbations", *SIAM J. Stat. Comp.*, vol.2, pp.35-50, 1981.

[8] K.Brown, *Numerical solution of system of nonlinear algebraic equations*, New York: Academic press, pp.281-348, 1973.

[9] T.Matsumoto, L.O.Chua and H.Suzuki, "CNN cloning template: connected component detector", *IEEE Trans. Circ. Syst,*, vol.CAS-37, pp.633-635, 1990.

Chapter 4

Associative Memory with DTCNN

4.1 Introduction

The artificial realization for the associative memory is one of the most important problems on the neural network applications. In several books[1, 2] and papers[3-8], the ability of neural networks to implement associative memories has been discussed. First, the information of several prototypes are stored into a neural network and then, a signal is inputted to the network where some of information from a prototype is lossed because of the distortions and noises during the signal transmission and processing. Then, all or most of original information can be recovered with the associative memory. The researches on the associative memory can be directly applied to pattern recognitions and classifications.

Depending on recalling approaches of stored information, associative memories can be classified into two groups. The approach in the first group can be performed in the feedforward mode, where a signal is only from input toward output. The typical example is a linear associative memory presented by Kohenen[1]. The second group of associative memories performs the recalling computation with feedback operation, these networks are called recurrent networks. A typical example of recurrent associative memory is the Hopfield network.

Hopfield presented continuous time and discrete time systems that are capable of implementing associative memory in 1982 and 1984[5, 6], respectively. Recently, several other investigators addressed the analysis of various types of continuous time and discrete time neural networks[7,8, 9].

For a two-dimensional cellular associative memory with $N \times M$ cells, one neural cell has a output from -1 tp $+1$, which corresponds to one bit in the two-dimensional prototype with multi-gray degree, so that the total state of this cellular associative memory describes a $N \times M$ bits two-dimensional prototype. Since the associative recursion algorithm is carried out with the synchronize refreshing approach, the state of the ith cell at time t_m can be transferred to the input of the jth cell through the connection weight A_{ij}, which affects the next state.

In order to realize associative memory, first, we must store the information of some prototypes into the networks through a learning process. During a pattern storing stage, the weights A_{kl} $(k = 1, 2, \cdots, n; l = 1, 2, \cdots, n)$ are gradually set to suitable values with a learning rule so that the weight matrix A includes the information of all prototypes. Then, in a stage of the prototype recalling, a probe pattern which just remains a part of information of an original prototype pattern is input to the cellular associative memory as the initial state. Furthermore, due to the monotone-decreasing property of the energy function, if the stationary condition is satisfied for the stored prototypes, after some recursion iterations, we can find out the complete stored prototype which has minimal Hamming distance to the input pattern within all stored prototypes. In this way, it is enable to realize the associative memory with a cellular neural network.

There some types of learning rules for associative memories[9]. Differing from the supervised learning mode and the unsupervised learning mode, most of artificial neural memories are trained under a batch learning mode. It means that the complete design information is available a priori, so that the network is first designed by recording desired equilibria points, after then, the weights of such network remain fixed during the associative memory process. Experimental data on biological systems have led Hebb's learning mechanism[20] whereby the synaptic coupling between two neurons is enhanced if both neurons are active at the same time. Based on this idea, the outer product method for computing the coupling coefficients has been proposed by Cooper et al[21]. Outer product learning rule is used for cellular associative memories by S.Tan et al[16] such that the application area of CNN is extended to associative memories, which is one of the most important function of the brains. But associative memories with the outer product method have some fatal weaknesses to limit their applications. The associative memories designed by the outer product method can not guarantee every stored prototypes as equilibrium points in general, so that the stored information can not always be recovered. In the section 4.2, first, we describe the outer product learning approach to set up suitable values for the weights of associative memory with discrete-time cellular neural network, these values are related to the object patterns information. This procedure is called as storing object patterns. After then, we focus on the stability analysis and present two theorems to ensure stored prototypes by the outer product learning algorithm as equilibrium points of cellular associative memories. Some application examples are also given in this section. In another aspect, experience and theoretical analysis show that Hopfield network with the outer product method can effectively store only up to $0.15n$ arbitrary vectors as equilibrium points, where n denotes the order of the network[18]. Moreover, it may be possible that the memory capacity and/or ability of an artificial associative memory decreases as the number of interconnection decreasing. We want to find another learning method suitable for the cellular associative memory networks. In the section 4.3, we present a middle-mapping learning algorithm for the cellular a ssociative memory. Its basic principle is similar to the project learning rule presented by L.Personnaz et al[19] where the projection learning rule was considered to be used for the system presented by McCulloch and Pitts[23], which operates in a synchronous mode. Since in the interconnection weight matrix A obtained with the projection learning rule, the diagonal element A_{ii} is equal and/or approximate to 1, never equal to zero[19], so it is difficult to apply this method directly to the Hopfield network. But in our memory, this kind of problems does not exist. It can guarantee to store a given vector as an equilibrium point so that every stored prototype is retrievable.

4.2 Outer product learning algorithm

Outer product learning rule is used for discrete-time cellular associative memories by S.Tan et al[16] in 1990, but the convergent and stability conditions are not analyzed in their paper. Moreover, in general, outer product learning rule can not guarantee that networks always store the desired prototypes as equilibrium points of the network. How to solve this problem and, what is the stationary condition of a cellular associative memory, the analyses are also not given. In this section, first, we describe the outer product learning approach to set up suitable values for the weights of associative memory with discrete-time cellular neural network, these values are related to the object patterns information. This procedure is called as storing object patterns. After then, we focus on the stability analysis and present two theorems to ensure stored prototypes by the outer product learning algorithm as equilibrium points of cellular associative memories. Some application examples are also given in this section.

4.2.1 Storing object patterns

In order to realize associative memory, first we must store all the object patterns information into the associative memory network. This storing process is also called as the network learning process. According to the Hebb theory[20], the weight matrix contains the information of the stored patterns, so we try to find out a technique to set up the weight matrix that will produce a stationary state of that network for each object pattern. Since the energy function of the cellular associative memory network is the monotone decreasing function, the stationary state of the network lies in a minimal point of the energy function. Through storing a pattern, we minimize the value of the energy function for the particular pattern so that it occupies a minimal point in the energy landscape. However, we also want to leave any previously stored patterns in their hollows at the same time, so that adding new patterns does not destroy any of the previous information.

Let us show this learning process in terms of the object patterns and the corresponding energy function. First, assume that there are S stored object patterns $\mathbf{o}^{(i)} \subset R^n, n = N \times M, i = 1, 2, \cdots, S$, which elements are either $+1$ or -1.

To simplify the analysis, we let $B = 0$ and $I = 0$ in the discrete-time cellular associative memory defined with (2.12), then, the state equation and the energy function can be described as follows:

$$\begin{aligned} \frac{1}{R_x} v_i(k+1) &= \sum_{j=1}^{n} A_{ij}\, y_j(k) && \text{(4.1a)} \\ y_i(k+1) &= \operatorname{sat}(v_i(k+1)) && \text{(4.1b)} \end{aligned}$$

Then, the energy function of discrete-time CNN in (2.25) can be described as follows:

$$\begin{aligned} E(k+1) &= -\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\, y_i(k+1)\, y_j(k) + \frac{1}{2R_x}\sum_{i=1}^{n} y_i^2(k+1) \\ &= E_1(k+1) + E_2(k+1) && \text{(4.2)} \end{aligned}$$

Because A_{ij} contains the patterns information mapped into the neighborhood $N_r(i)$ from all the object patterns, we can split E_1 into two parts. One represents the effects of all the patterns except the lth one and denotes it by A'_{ij}, and the second is the contribution made by the lth pattern alone, $l \in \{1, 2, \cdots, S\}$, shown as $A^{(l)}_{ij}$. Besides, we ignore the description for the iteration number, because we just consider final stationary state in which there exists $\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)}$. Thus, we can rewrite E_1 in two parts

$$\begin{aligned} E_1 &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} A'_{ij} y_i y_j - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} A^{(l)}_{ij} y_i y_j \\ &= E_{others} + E_l \end{aligned} \tag{4.3}$$

where E_l is the energy due to the pattern l, while E_{others} is due to the contributions from all the other patterns.

Storing lth pattern corresponds to making the energy function as small as possible for this pattern. The first term corresponds to the other patterns, so we can not change this term now. But we can reduce the contribution made by the second term E_l. In other words, to store pattern l, i.e. $\mathbf{y} = \mathbf{y}^{(l)}$, we want to minimize the contribution to the energy function from the lth energy term, and so make

$$E_l = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} A^{(l)}_{ij} y^{(l)}_i y^{(l)}_j \tag{4.4}$$

as small as possible.

Due to the minus sign in the equation above, this corresponds to making

$$\sum_{i=1}^{n}\sum_{j=1}^{n} A^{(l)}_{ij} y^{(l)}_i y^{(l)}_j$$

as large as possible.

Now, since the elements in the lth sample $\mathbf{y}^{(l)}$ are either -1 or +1, hence $y^{(l)2}_i$ is always positive. So if we make an energy term equal $y^{(l)2}_i y^{(l)2}_j$, it will always be positive and the sum will arrive to the largest value.

Thus, in this case, we have

$$\sum_{i=1}^{n}\sum_{j=1}^{n} A^{(l)}_{ij} y^{(l)}_i y^{(l)}_j = \sum_{i=1}^{n}\sum_{j=1}^{n} y^{(l)2}_i y^{(l)2}_j$$

It is equal to make the weight as follows:

$$A^{(l)}_{ij} = y^{(l)}_i y^{(l)}_j \tag{4.5}$$

Therefore, we obtain an important result: setting the values of the weight $A_{ij}^{(l)} = y_i^{(l)} y_j^{(l)}$ for every i and j will minimize the energy function for the pattern l. In order to calculate the weight values for all the patterns, we sum this equation over all the patterns as follows:

$$A_{ij} = \sum_{l=1}^{S} A_{ij}^{(l)} = \sum_{l=1}^{S} y_i^{(l)} y_j^{(l)} \tag{4.6}$$

Since the energy function shown in (4.2) can be rewritten as

$$\begin{aligned} E &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} y_i y_j + \frac{1}{2R_x}\sum_{i=1}^{n} y_i^2 \\ &= -\frac{1}{2}\sum_{i=1}^{n}\Big[\sum_{\substack{c(j)\in N_r(i)\\ c(j)\neq c(i)}} A_{ij} y_i y_j + (A_{ii} - \frac{1}{R_x}) y_i^2\Big] \end{aligned} \tag{4.7}$$

we make some adjusting for A_{ij} so as to minimize the energy function once again for the stored object patterns. In this way, the weight between $c(i)$ and its neighborhood cell $c(j)$ is given by

$$\begin{aligned} A_{ij} &= \sum_{\substack{l=1\\ c(j)\in N_r(i)\\ c(j)\neq c(i)}}^{S} y_i^{(l)} y_j^{(l)} \\ A_{ii} &= \frac{1}{R_x} + \sum_{l=1}^{S} y_i^{(l)2} \end{aligned}$$

Let us summary above description as follows:

Lemma 4.1 *Applying outer product learning rule to a discrete-time cellular associative memory, in the case of $B = 0$ and $I = 0$, we can derive the weight coefficients connecting $c(i)$ and its neighboring cell $c(l)$ as*

$$A_{ij} = \sum_{\substack{l=1\\ c(j)\in N_r(i)\\ c(j)\neq c(i)}}^{S} y_i^{(l)} y_j^{(l)} \tag{4.8}$$

$$A_{ii} = \frac{1}{R_x} + \sum_{l=1}^{S} y_i^{(l)2} \tag{4.9}$$

Based on Theorem 2.4, it is known that a DTCNN is convergent if it is diagonal-column eigendominant. Considering the connection coefficients described in Lemma 4.1, we can get a theorem as follows.

Theorem 4.1 *If the number of cells during the neighborhood $N_r(i)$ is equal to p, and the R_x is selected so that the condition*

$$\frac{1}{R_x} \geq (p-2)S \tag{4.10}$$

is met, the discrete-time cellular associative memory with outer product learning algorithm is convergent, its generalized energy function is monotone decreasing.

Proof: From Definition 2.2, the weight matrix A is said as diagonal-column eigendominant if the ith diagonal element A_{ii} is greater than or equal to the sum of absolute values of other elements in the ith column. i.e.

$$A_{ii} \geq \sum_{\substack{j=1 \\ j\neq i}}^{n} | A_{ji} | \tag{4.11}$$

$$\forall i \in \{1, 2, \cdots, n\}$$

Since the matrix A is a sparse matrix, besides the elements A_{ij} for $c(j) \in N_r(i)$, other elements A_{ij} must equal zero. Moreover, according to Lemma 4.1, we have

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} | A_{ji} | \leq (p-1)S$$

But

$$\sum_{l=1}^{S} y_i^{(l)\,2} = S$$

so that when the relation (4.10) is met, the weight matrix A has diagonal-column eigendominant property as denoted in (4.11), the energy function is monotone decreasing and the cellular neural network designed by outer product learning algorithm is convergent.

□

4.2.2 Stationary character analysis

There are some different ways to characterize the performance of an associative memory network, but finding the stationary condition of the stored patterns is one of the most important problems. Because, if and only if every stored object pattern corresponds to a fixed point in the associative recursive algorithm, then these object patterns will be retrieval. In this section, we discuss the stationary property of the cellular associative memory, and present a stationary condition.

First, we give a definition of the stationary state of the cellular memory network.

Definition 4.1 *The state of an associative memory network* $\mathbf{y}$ *will be called stationary point, if and only if, for the recursive operation from an initial state* $\mathbf{y}$, *the following state is still kept in* $\mathbf{y}$. *i.e.*

$$\mathbf{y}(k+1) = \mathbf{y}(k) \tag{4.12}$$

For a discrete-time cellular associative memory network with $B = 0$ *and* $I = 0$, *this condition can be denoted bellow*

$$y_i(k) g_i[\mathbf{y}(k)] > 1 \tag{4.13}$$
$$\forall i \in \{1, 2, \cdots, n\}$$

where, $g_i[\cdot]$ *is the transfer operator of the cell* $c(i)$ *obtained from the state equation (4.1).*

This stationary condition is suitable for a discrete-time cellular associative memory networks with $B = 0$ and $I = 0$. But at the same time, it could be found from the structure of cellular neural network, that $v_i = g_i[\mathbf{v_y}]$ has direct relation only with the output value y_j of the cells $c(j) \in N_r(i)$. It implies that, when each cell's equilibrium condition is judged individually, just a part of the stored object pattern is concerned. Thus, we define the local pattern as bellow:

Definition 4.2 *Assume* $N \times M$ *bits two-dimension pattern* $\mathbf{o}$ *are stored in a two-dimension cellular associative memory network constituted by* $N \times M$ *cell units. Let* $N \times M = n$ *and write* $\mathbf{o}$ *as a 1-dimensional vector, thus, we get* $\mathbf{o} = \{o_k, k = 1, 2, \cdots, n\}$. *In the neighborhood of the cell* $c(i)$, *the number cells covered in this neighborhood is p, a part of the object pattern is mapped and stored, we call this part of the object pattern* *as a sub-prototype $\mathbf{f}_i$, $\mathbf{f}_i \in R^p$. The context in $\mathbf{f}_i$ is a part of the object pattern $\mathbf{o}$. For different neighborhood, the contents of $\mathbf{f}_i$ are also different. The $\mathbf{f}_i$ corresponding with the $N_r(i)$ is*

$$\mathbf{f}_i = \{f_j = o_j;\ c(j) \in N_r(i)\}, \qquad \mathbf{f}_i \in R^p$$

Similarly, we can define a state sub-vector $\hat{\mathbf{v}}_i$ and a output sub-vector $\hat{\mathbf{y}}_i$ to correspond to the neighborhood $N_r(i)$ as

$$\begin{aligned} \hat{\mathbf{v}}_i &= \{\hat{v}_j = v_j;\ c(j) \in N_r(i)\}, \qquad \hat{\mathbf{v}}_i \in R^p \\ \hat{\mathbf{y}}_i &= \{\hat{y}_j = y_j;\ c(j) \in N_r(i)\}, \qquad \hat{\mathbf{y}}_i \in R^p \end{aligned}$$

From this definition, we prove a condition of the stationary state on the neighborhood $N_r(i)$ of the cellular associative memory network. Obviously, if every neighborhood of the cellular associative memory network satisfies this stationary condition, the network is stationary.

Theorem 4.2 *Suppose a cellular associative memory network has $M \times N = n$ cells. Assume the number of cells in a neighborhood is $p = (2r+1)^2$, every cell is directly connected with near cells in its neighborhood, and within a neighborhood $N_r(i)$, S sub-prototypes $\mathbf{f}_i^{(1)}, \mathbf{f}_i^{(2)}, \cdots, \mathbf{f}_i^{(S)}$ are stored, $\mathbf{f}_i^{(k)} \in R^{(2r+1)\times(2r+1)}$, and the Hamming distance between arbitrary two sub-prototypes $\mathbf{f}_i^{(k)}$ and $\mathbf{f}_i^{(l)}$ is $\delta(\mathbf{f}_i^{(k)} \cdot \mathbf{f}_i^{(l)})$. Then, if and only if the condition (4.14) is satisfied, the local object patterns $\mathbf{f}_i^{(l)}$ stored in $N_r(i)$ will correspond to the stationary states obtained by the associative memory recursive algorithm.*

$$p + \sum_{\substack{k=1 \\ k \neq l}}^{S} [p - 2\delta(\hat{\mathbf{y}}_i^{\mathsf{T}(k)}, \mathbf{f}_i^{(l)})]\hat{y}_i^{(k)} f_i^{(l)} \geq 0 \tag{4.14}$$

Proof: For $B = 0$ and $I = 0$ in (4.1a), we have

$$\sum_{c(j)\in N_r(i)} A_{ij}\hat{y}_j(k) - \frac{1}{R_x}\hat{v}_i(k+1) = 0 \tag{4.15}$$

$$\forall i \in \{1, 2, \cdots, n\}$$

First, for simplicity, we denote $\hat{v}_i(k)$ and $\hat{y}_i(k+1)$ as $\hat{v}_i$ and $\hat{y}_i$, respectively. Then while $\hat{\mathbf{y}}_i = \mathbf{f}_i$, from Lemma 4.1, the network state $\hat{v}_i = g_i[\mathbf{f}_i]$ is given by

$$\begin{aligned}
g_i[\mathbf{f}_i] &= R_x \sum_{c(j)\in N_r(i)} A_{ij} f_j \\
&= R_x[A_{ii} f_i + \sum_{\substack{c(j)\in N_r(i)\\ j\neq i}} A_{ij} f_j] \\
&= R_x[f_i(\sum_{l=1}^{S} \hat{y}_i^{2(l)} + \frac{1}{R_x}) + \sum_{l=1}^{S} \sum_{\substack{c(j)\in N_r(i)\\ j\neq i,j}} \hat{y}_i^{(l)} \hat{y}_j^{(l)} f_j] \\
&= f_i + R_x[\sum_{l=1}^{S} \hat{y}_i^{2(l)} f_i + \sum_{l=1}^{S} \sum_{\substack{c(j)\in N_r(i)\\ j\neq i}} \hat{y}_i^{(l)} \hat{y}_j^{(l)} f_j] \\
&= f_i + R_x[\sum_{l=1}^{S} \sum_{c(j)\in N_r(i)} \hat{y}_i^{(l)} \hat{y}_j^{(l)} f_j] \\
&= f_i + R_x \sum_{l=1}^{S} (\hat{\mathbf{y}}_i^{\top(l)} \cdot \mathbf{f}_i) \hat{y}_i^{(l)}
\end{aligned} \tag{4.16}$$

where, $\hat{\mathbf{y}}_i^{(l)}$ is the output corresponding with the lth local object pattern stored in the neighborhood N_r, and $^\top$ denotes the transpose.

According to (4.4), if $\mathbf{f}_i^{(l)}$ is a recoverable sub-prototype and corresponds to a minimum point of the energy function, it must have

$$\begin{aligned}
f_i^{(l)} \times g_i(\mathbf{f}_i^{(l)}) &= f_i^{(l)2} + R_x \sum_{k=1}^{S} (\hat{\mathbf{y}}_i^{\top(l)} \cdot \mathbf{f}_i) f_i^{(l)} \\
&\geq 1
\end{aligned} \tag{4.17}$$

Since the first term in right side above equation $f_i^{(l)2} = 1$, above relation is corresponding to

$$\sum_{k=1}^{S} (\hat{\mathbf{y}}_i^{\top(l)} \cdot \mathbf{f}_i) f_i^{(l)} \geq 0 \tag{4.18}$$

Because the number of cells in a neighborhood $N_r(i)$ is p, we have

$$\sum_{k=1}^{S} (\hat{\mathbf{y}}_i^{\top(l)} \cdot \mathbf{f}_i) f_i^{(l)} = p + \sum_{\substack{k=1\\ k\neq l}}^{S} (\hat{\mathbf{y}}_i^{\top(l)} \cdot \mathbf{f}_i) f_i^{(l)}$$

$$= \; p + \sum_{\substack{k=1\\k\neq l}}^{S} [p - 2\sigma(\hat{\mathbf{y}}_i^{\mathsf{T}(l)} \cdot \mathbf{f}_i)]\hat{y}_i^{(k)} f_i^{(l)} \tag{4.19}$$

so that $\mathbf{f}_i^{(l)}$ is a recoverable sub-prototype if and only if the relation (4.14) is satisfied.

□

Obviously, for eacn sub-prototype $\mathbf{f}_i^{(l)}$, the Hamming distance between it and other sub-prototypes stored in the same neighborhood, $\sigma(\hat{\mathbf{y}}_i^{\mathsf{T}(l)} \cdot \mathbf{f}_i)$ is diffrent. Beside it, the stored sub-prototype number S in a neighborhood is also different from each other. If and only if the relation (4.14) is satisfied for each sub-prototype stored in every neighborhood $N_r(i), i = 1, 2, \cdots, n$, the stored object patterns correspond to the stationary points of the network, and every of them will be recallable.

From above theorem, we can derive another sufficient condition as follows. First, we introduce a definition.

Definition 4.3 *For each sub-prototype $\mathbf{f}_i^{(l)}$ described in (4.14), we can define two collections $\mathbf{c}_1$ and $\mathbf{c}_2$.*

$$\mathbf{c}_1 \; = \; \{\mathbf{f}_i^{(k)};\; f_i^{(k)} f_i^{(l)} > 0\} \tag{4.20a}$$

$$\mathbf{c}_2 \; = \; \{\mathbf{f}_i^{(k)};\; f_i^{(k)} f_i^{(l)} < 0\} \tag{4.20b}$$

$$k \in \{1, 2, \cdots, S\}$$

Then, we can get a sufficient condition on the stationary property of the cellular associative memory as follows.

Theorem 4.3 *For sub-prototype $\mathbf{f}_i^{(l)}$, we assume that the number of elements in $\mathbf{c}_1$ is m and the number of elements in $\mathbf{c}_2$ is $s - m$. If the following relation is satisfied, $\mathbf{f}_i^{(l)}$ is corresponding to a stationary points of the network and it is recoverable after some times of associative operation.*

$$m\,[p - 2 \max_{\hat{\mathbf{y}}_i^{(k)} \in \mathbf{c}_1} \sigma(\hat{\mathbf{y}}_i^{\mathsf{T}(k)} \cdot \mathbf{f}_i^{(l)})] - (s-m)[p - 2 \min_{\hat{\mathbf{y}}_i^{(k)} \in \mathbf{c}_2} \sigma(\hat{\mathbf{y}}_i^{\mathsf{T}(k)} \cdot \mathbf{f}_i^{(l)})] \geq 0 \tag{4.21}$$

Proof: From (4.18), we know if and only if

$$\sum_{k=1}^{S}(\hat{\mathbf{y}}_i^{\mathsf{T}(l)} \cdot \mathbf{f}_i) f_i^{(l)} \geq 0$$

the $\mathbf{f}_i^{(l)}$ is recoverable. But

$$\begin{aligned}
&\sum_{k=1}^{S}(\hat{\mathbf{y}}_i^{\mathsf{T}(l)} \cdot \mathbf{f}_i) f_i^{(l)} \\
&= \sum_{\hat{\mathbf{y}}_i^{(k)} \in \mathbf{c}_1} [p - 2\sigma(\hat{\mathbf{y}}_i^{\mathsf{T}(k)} \cdot \mathbf{f}_i^{(l)})] - \sum_{\hat{\mathbf{y}}_i^{(k)} \in \mathbf{c}_2} [p - 2\sigma(\hat{\mathbf{y}}_i^{\mathsf{T}(k)} \cdot \mathbf{f}_i^{(l)})] \\
&> m\,[p - 2 \max_{\hat{\mathbf{y}}_i^{(k)} \in \mathbf{c}_1} \sigma(\hat{\mathbf{y}}_i^{\mathsf{T}(k)} \cdot \mathbf{f}_i^{(l)})] - (s-m)[p - 2 \max_{\hat{\mathbf{y}}_i^{(k)} \in \mathbf{c}_2} \sigma(\hat{\mathbf{y}}_i^{\mathsf{T}(k)} \cdot \mathbf{f}_i^{(l)})]
\end{aligned} \quad (4.22)$$

so that if (4.21) is satified, the $\mathbf{f}_i^{(l)}$ is corresponding to a stationary points of the network and it is recoverable after some times of associative operation.

□

4.2.3 Illustration example

In this section, a specific example of using a cellular neural network as the associative memory is given.

Example 1:

The network consists of 5×5 neural cells. Every cell has the neighborhood with $r = 1$ and the parameters as bellow:

$$R_x = 0.02\ k\Omega, \qquad B = 0\ (k\Omega)^{-1}, \quad and \quad I = 0\ mA$$

Since, for a cell lying in the boundary of the grid, there are not enough neighboring cells arounded it to construct a complete neighborhood, it may give bad influence on the associative operation of our memory network. To solve this problem, a ring of dummy cells are added to the border of the grid. The initial states of the dummy cells are all $y_i = -1$. After then, their states are changed as the same as the inner cells. The number of the dummy cells of a cellular associative memory grid with available $N \times M$ cells is $2(N+M+2)$. When N and M are large enough, $2(N+M+2)/(N \times M)$ becomes smaller, the dummy cells just take a very small part of total number of all cells. But they play important rule to the ability of the memory.

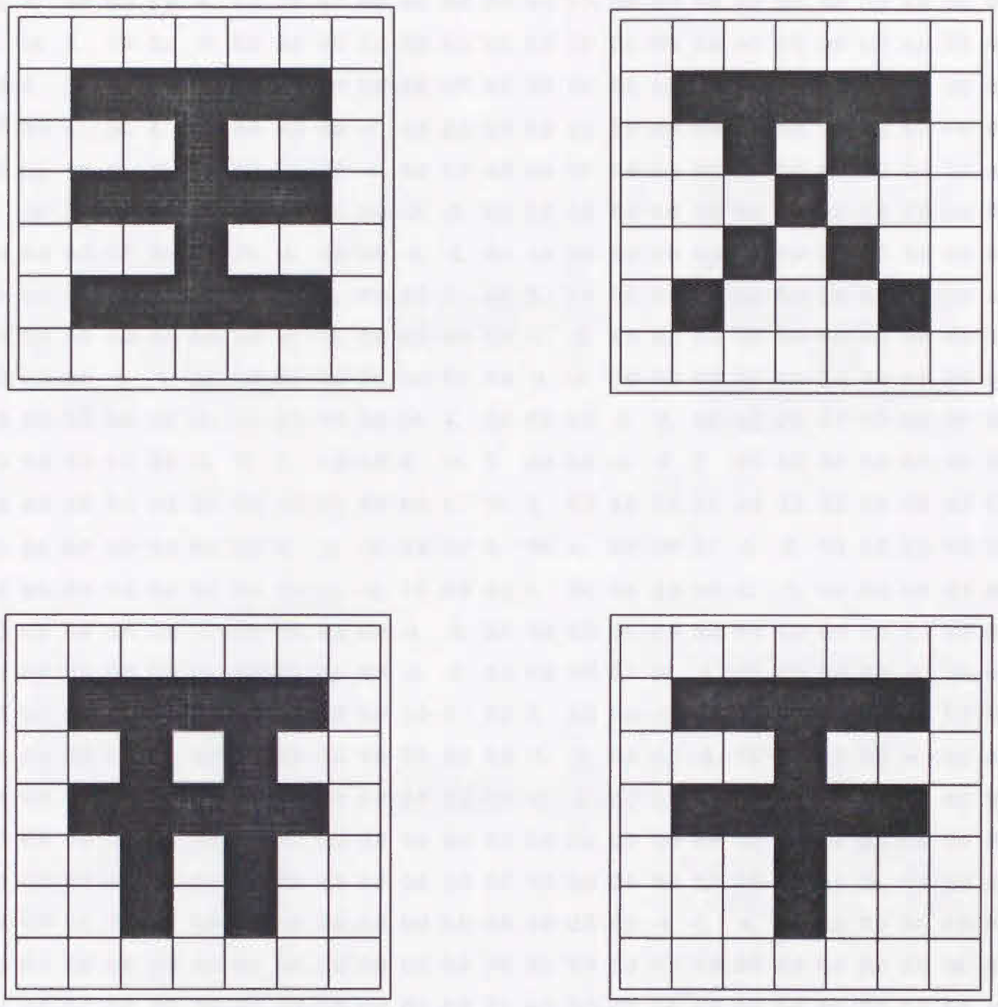


Figure 4.1: Stored sample patterns

First, the sample patterns shown in Fig.4.1 are stored into this cellular associative memory network.

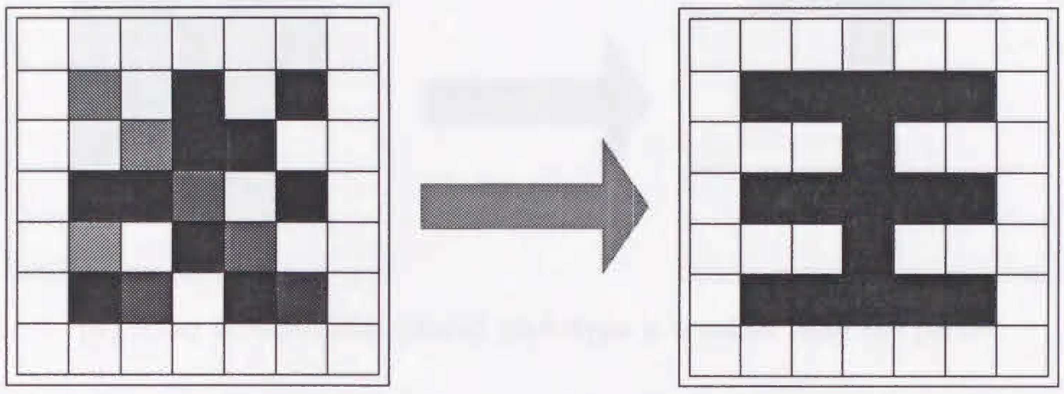
Obviously, here the numbers of stored prototypes is equl to 4 and $p = 9$, the convergent condition (4.10) mentioned in Theorem 4.1 is satisfied so that the network is convergent in this case.

Using (4.8) and (4.9), the weight A_{ij} between connected cells within a neighborhood of the network is determined from the stored patterns information. The numerical values of the elments in A weight matrix are shown as Figure 4.2.

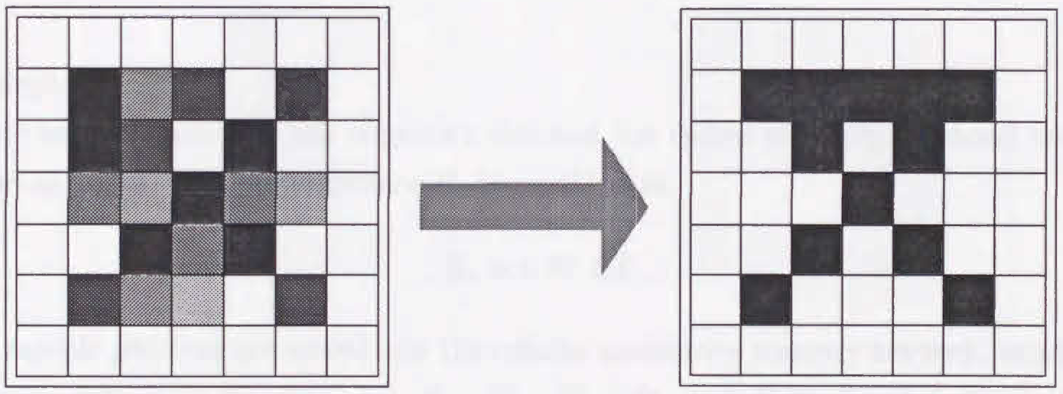
$$\begin{bmatrix}
54. & 4. & 0.0 & 0.0 & 0.0 & -4. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
4. & 54. & 4. & 0.0 & 0.0 & -4. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 4. & 54. & 4. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 4. & 54. & 4. & 0.0 & 0.0 & 0.0 & 0.0 & -4. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 4. & 54. & 0.0 & 0.0 & 0.0 & 0.0 & -4. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
-4. & -4. & 0.0 & 0.0 & 0.0 & 54. & 0.0 & 0.0 & 0.0 & 0.0 & -2. & -2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 54. & -4. & 0.0 & 0.0 & -2. & -2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -4. & 54. & -4. & 0.0 & 0.0 & 2. & 0.0 & 2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -4. & 54. & 0.0 & 0.0 & 0.0 & 0.0 & -2. & -2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & -4. & -4. & 0.0 & 0.0 & 0.0 & 0.0 & 54. & 0.0 & 0.0 & 0.0 & -2. & -2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -2. & -2. & 0.0 & 0.0 & 0.0 & 54. & 4. & 0.0 & 0.0 & 0.0 & -2. & -2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -2. & -2. & 2. & 0.0 & 0.0 & 4. & 54. & 2. & 0.0 & 0.0 & -2. & -2. & 2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 2. & 54. & 2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 2. & -2. & -2. & 0.0 & 0.0 & 2. & 54. & 4. & 0.0 & 0.0 & 2. & -2. & -2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -2. & -2. & 0.0 & 0.0 & 0.0 & 4. & 54. & 0.0 & 0.0 & 0.0 & -2. & -2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -2. & -2. & 0.0 & 0.0 & 0.0 & 54. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -2. & -2. & 0.0 & 0.0 & 0.0 & 0.0 & 54. & -4. & 0.0 & 0.0 & 0.0 & 0.0 & -4. & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 2. & 0.0 & 2. & 0.0 & 0.0 & -4. & 54. & -4. & 0.0 & 0.0 & 0.0 & 4. & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -2. & -2. & 0.0 & 0.0 & -4. & 54. & 0.0 & 0.0 & 0.0 & -4. & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -2. & -2. & 0.0 & 0.0 & 0.0 & 0.0 & 54. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 54. & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 54. & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -4. & 4. & -4. & 0.0 & 0.0 & 0.0 & 54. & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 54. & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 54.
\end{bmatrix}$$

Figure 4.2: The A matrix for Example 1 $(K\Omega)^{-1}$

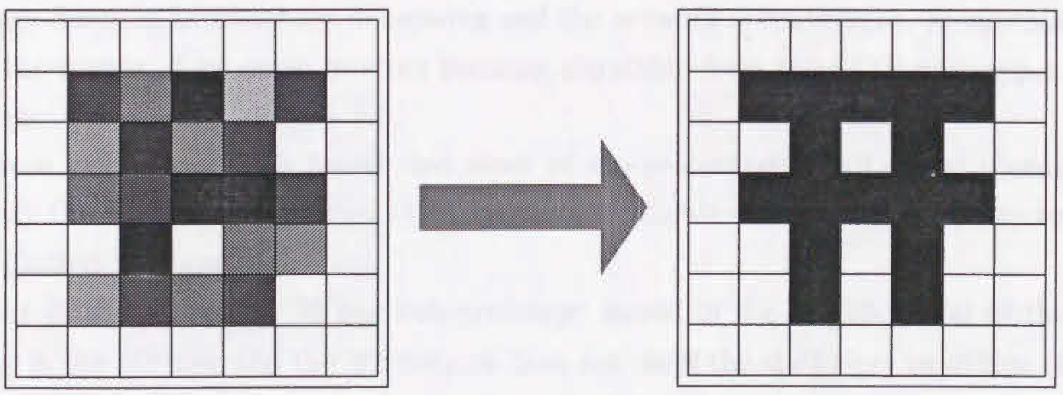
Moreover, based on the calculating for every neighborhood, the stationary condition (4.14) is also met for each stored sub-prototype so that they are recallable. Inputting four probe patterns shown as left side in Figure 4.3 which has distortion pixels from 48% to 68%, after some times associative memory recursion we can get the correct answer as shown in right side of Figure 4.3.



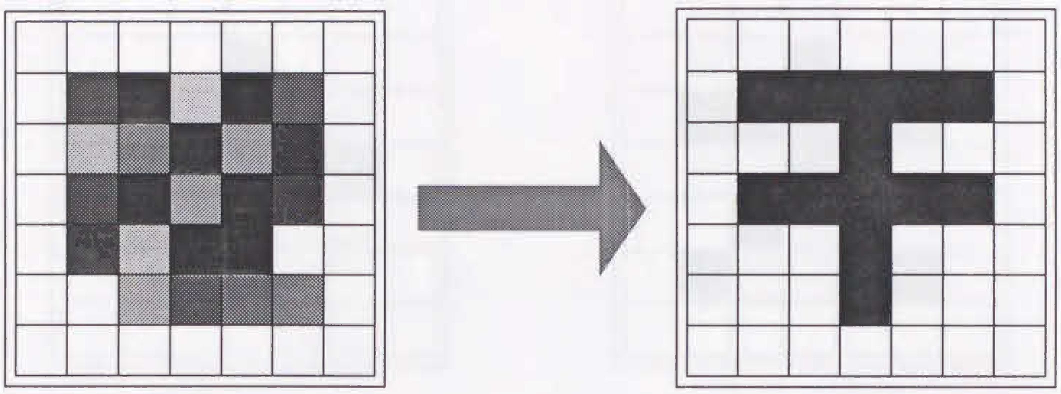
(a) After 31 iterations, stored prototype is recalled from the probe



(b) After 44 iterations, stored prototype is recalled from the probe



(c) After 35 iterations, stored prototype is recalled from the probe



(d) After 31 iterations, stored prototype is recalled from the probe

Figure 4.3: Associative memory with outer product learning rule

Example 2:

In the next example, the network's size and the radius of a neighborhood are the same as above. But the resistance R_x is modified as

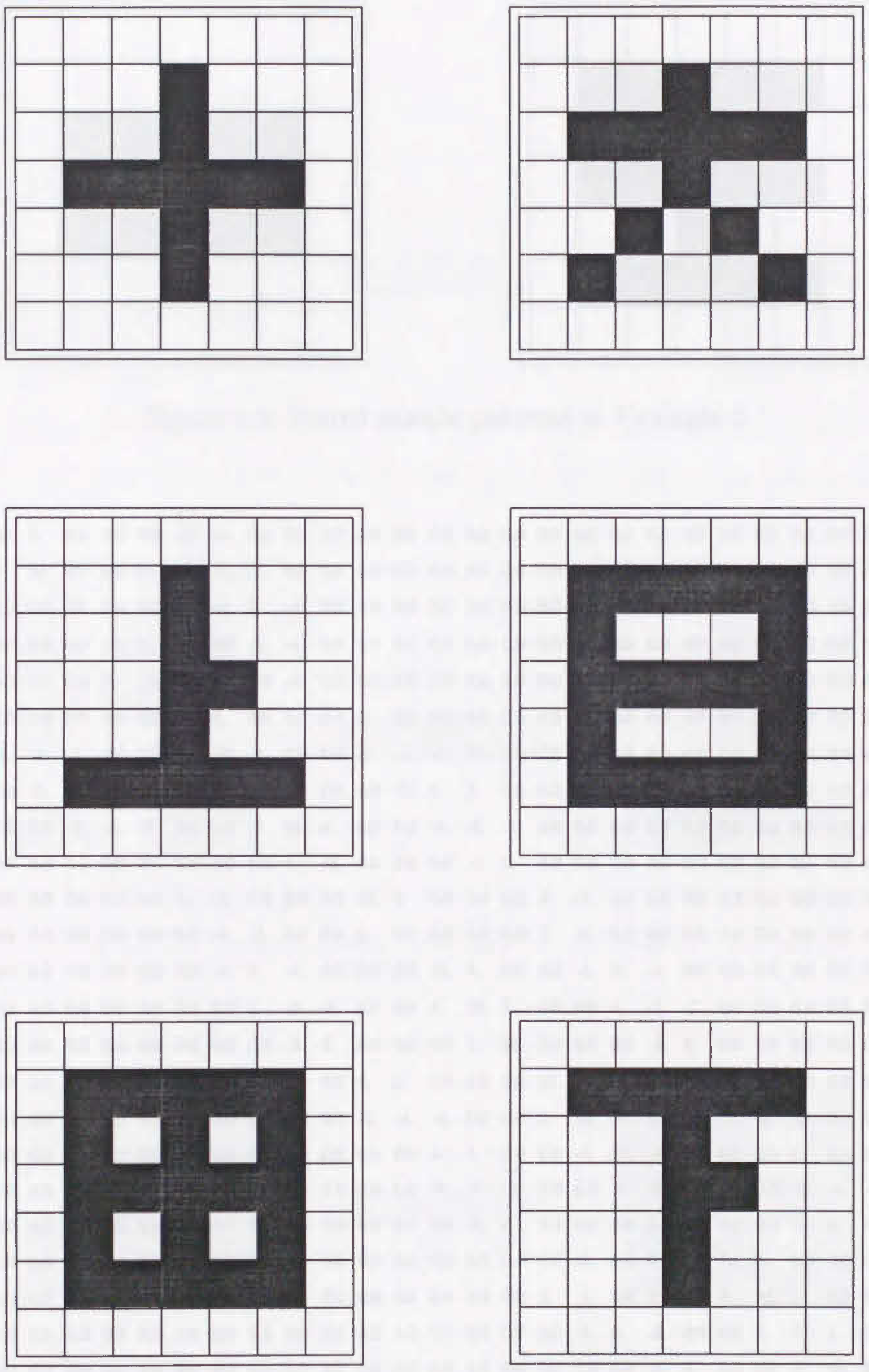
$$R_x = 0.01\ K\Omega$$

8 sample patterns are stored into the cellular associative memory network, which are Chinese characters 十、大、上、日、田、下、中、and 王 displayed in Figure 4.2.3.

In this example, the convergent condition (4.10) is also satisfied, the generalized energy function is monotone decreasing and the network is convergent. Meanwhile, we get the matrix A by outer product learning algorithm from stored all prototypes. the matrix A is shown in Figure 4.5.

From calculating, it is found that most of sub-prototypes in all stored prototypes satisfy the stationary condition (4.14) and are recallable sub-prototypes. Some results are plotted in Figure 4.6.

But in the prototype 大, one sub-prototype stored in the neighborhood of the cell lying in the 3*th* row and the 3*th* column does not meet the stationary condition (4.14) and it is not recallable in this case. A probe shown in the left side of Figure 4.7 is inputted into the network as a initial output state, after 20 times iteration operations, the result is obtained as the right of the same figure. The stored sub-prototype in $N_r(3,3)$ is destroyed but sub-prototypes stored in other cells are recalled successfully.



(to be continued)

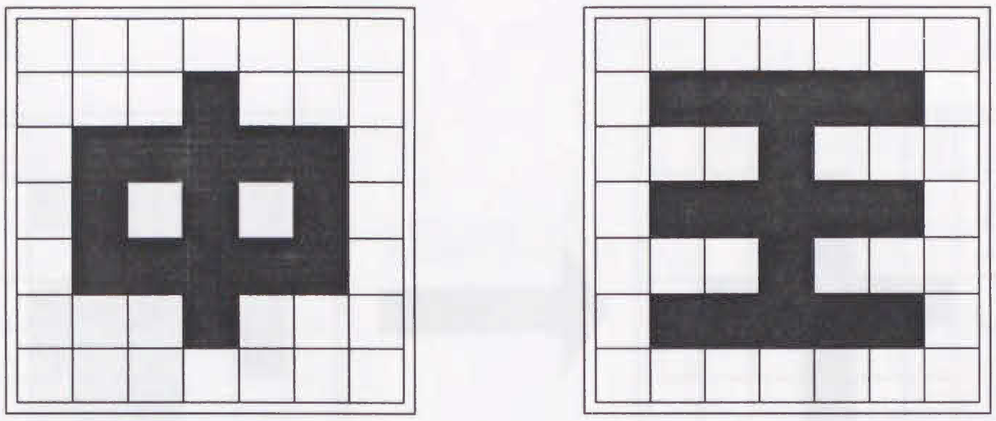
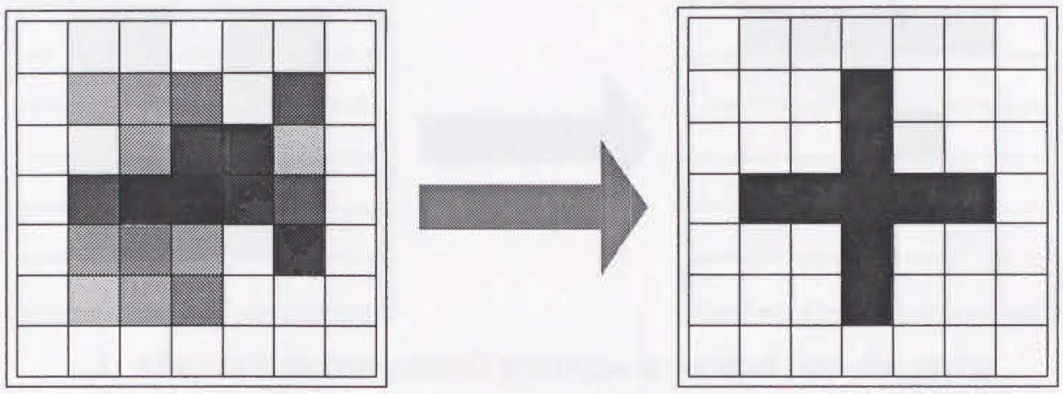


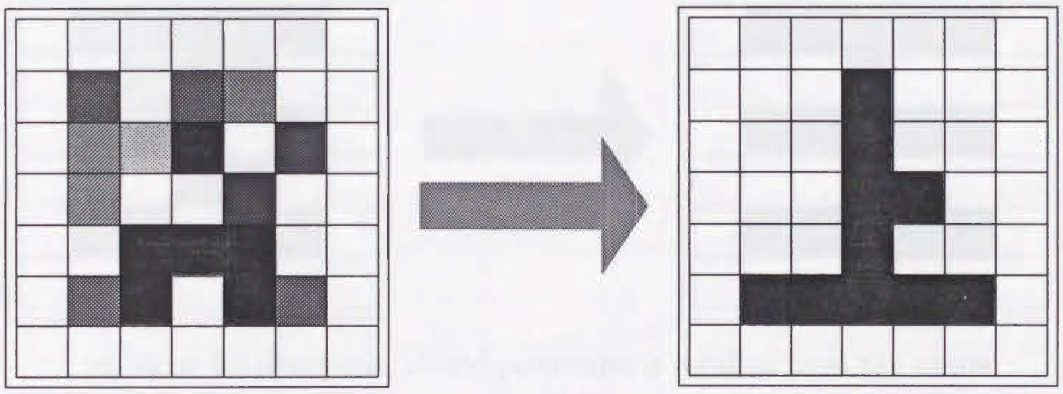
Figure 4.4: Stored sample patterns in Example 2

$$
\begin{bmatrix}
58. & 8. & 0.0 & 0.0 & 0.0 & 0.0 & -4. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
8. & 58. & 0.0 & 0.0 & 0.0 & 0.0 & -4. & -2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 58. & 0.0 & 0.0 & 0.0 & -4. & 6. & -4. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 58. & 8. & 0.0 & 0.0 & -2. & -4. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 8. & 58. & 0.0 & 0.0 & 0.0 & -4. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 58. & 4. & 0.0 & 0.0 & 0.0 & 2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
-4. & -4. & -4. & 0.0 & 0.0 & 4. & 58. & -2. & 0.0 & 0.0 & -2. & -4. & -4. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & -2. & 6. & -2. & 0.0 & 0.0 & -2. & 58. & -2. & 0.0 & 0.0 & -2. & 6. & 2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & -4. & -4. & -4. & 0.0 & 0.0 & -2. & 58. & 4. & 0.0 & 0.0 & -4. & -8. & -2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 4. & 58. & 0.0 & 0.0 & 0.0 & -4. & 2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 2. & -2. & 0.0 & 0.0 & 0.0 & 58. & 6. & 0.0 & 0.0 & 0.0 & 4. & -2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -4. & -2. & 0.0 & 0.0 & 6. & 58. & 0.0 & 0.0 & 0.0 & 2. & -4. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -4. & 6. & -4. & 0.0 & 0.0 & 0.0 & 58. & 4. & 0.0 & 0.0 & -4. & 4. & -4. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 2. & -8. & -4. & 0.0 & 0.0 & 4. & 58. & 2. & 0.0 & 0.0 & 4. & -8. & -2. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -2. & 2. & 0.0 & 0.0 & 0.0 & 2. & 58. & 0.0 & 0.0 & 0.0 & -2. & 4. & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 4. & 2. & 0.0 & 0.0 & 0.0 & 58. & 2. & 0.0 & 0.0 & 0.0 & 0.0 & 2. & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -2. & -4. & -4. & 0.0 & 0.0 & 2. & 58. & -4. & 0.0 & 0.0 & -2. & -4. & -6. & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 4. & 4. & 0.0 & 0.0 & -4. & 58. & -4. & 0.0 & 0.0 & 0.0 & 6. & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -4. & -8. & -2. & 0.0 & 0.0 & -4. & 58. & 2. & 0.0 & 0.0 & -6. & -4. & -2. \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -2. & 4. & 0.0 & 0.0 & 0.0 & 2. & 58. & 0.0 & 0.0 & 0.0 & 2. & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -2. & 0.0 & 0.0 & 0.0 & 58. & 6. & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 2. & -4. & 0.0 & 0.0 & 0.0 & 6. & 58. & 2. & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -6. & 6. & -6. & 0.0 & 0.0 & 2. & 58. & 2. & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -4. & 2. & 0.0 & 0.0 & 2. & 58. & 6. \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -2. & 0.0 & 0.0 & 0.0 & 0.0 & 6. & 58.
\end{bmatrix}
$$

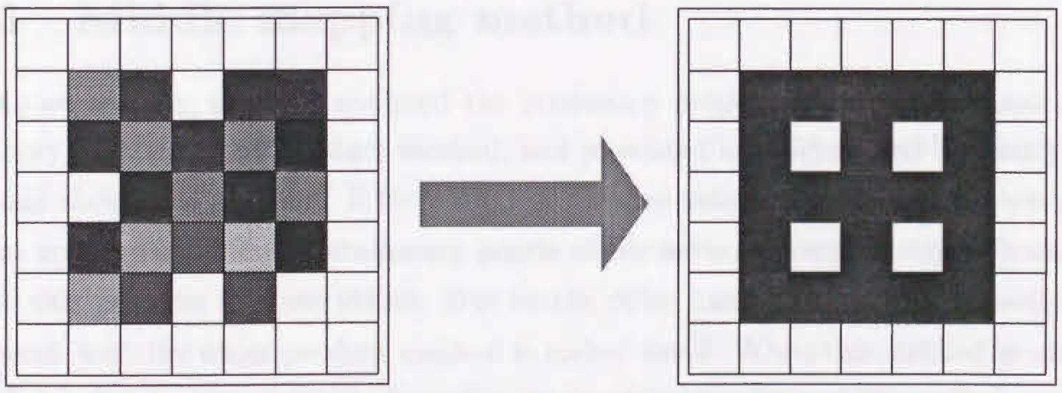
Figure 4.5: The A matrix for Example 2 $(K\Omega)^{-1}$



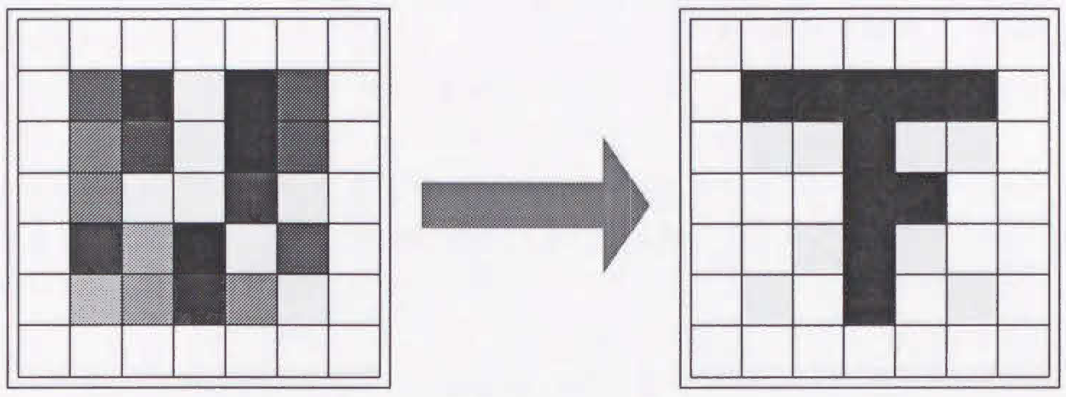
(a) After 9 iterations, stored prototype is recalled from the probe



(b) After 18 iterations, stored prototype is recalled from the probe



(c) After 19 iterations, stored prototype is recalled from the probe


(d) After 44 iterations, stored prototype is recalled from the probe

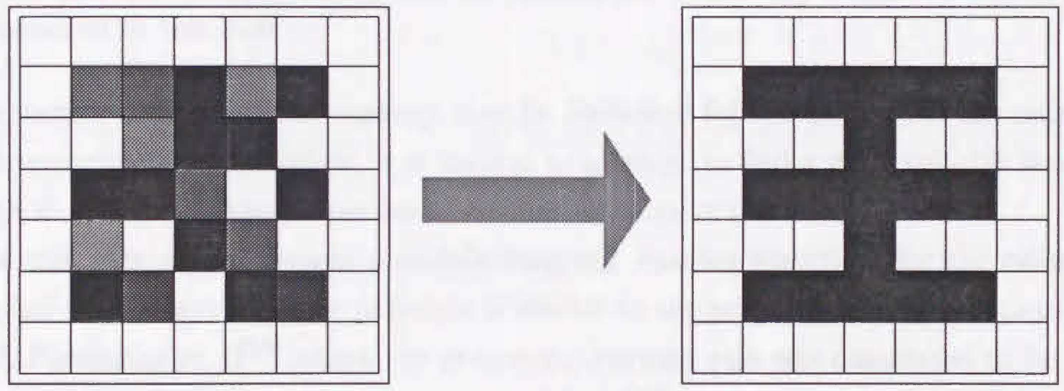
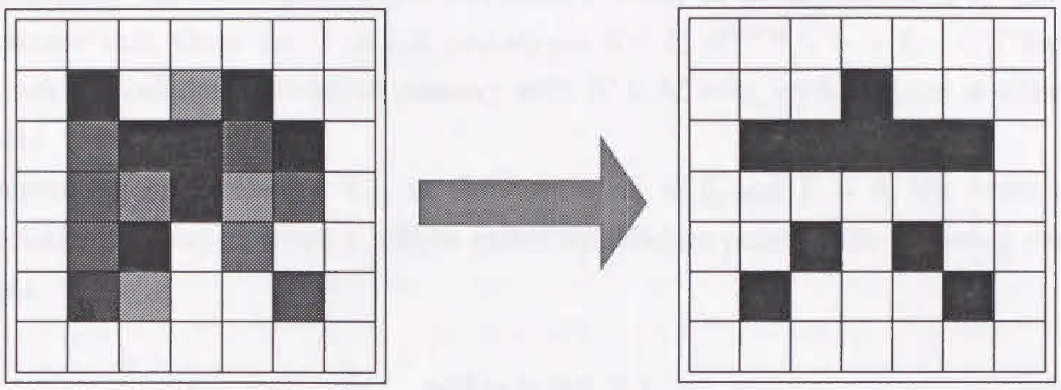

(e) After 22 iterations, stored prototype is recalled from the probe

Figure 4.6: Associative memory with outer product learning rule

4.3 Middle mapping method

In above section, we have analyzed the stationary property of a cellular associative memory with the outer product method, and presented a sufficient and necessary conditions about this problem. If the stored prototypes satisfy thie condition, then all of them are corresponding to stationary points of the networks respectively. We want to solve this problem to some extent. But on the other hand, the memory capacity of a network with the outer product method is rather small. When this method is used in a full interconnection network, from the experimental results and theoretical analyzes just $0.15n$ patterns can be effectively stored and recovered, where n is the number of the neuron units in the network[18]. When this method is applied for a cellular associa-



After 20 iterations, a part of prototype is recalled from the probe

Figure 4.7: One sub-prototype does not satisfy the stationary condition and can not be recurred in this case

tive memory, the available capacity may be decreased further because of the reducing of interconnection. Therefore, it is natural to attempt to find a more suitable learning method to improve the properties of cellular associative memories.

In thie section, we present a middle-mapping learning algorithm for the cellular a ssociative memory. Its basic principle is similar to the project learning rule presented by L.Personnaz et al[19] where the projection learning rule was considered to be used for the system presented by McCulloch and Pitts[23], which operates in a synchronous mode. Since in the interconnection weight matrix A obtained with the projection learning rule, the diagonal element A_{ii} is equal and/or approximate to 1, never equal to zero[19], so it is difficult to apply this method directly to the Hopfield network. But in our memory, this kind of problems does not exist. The main results of our memory are follows:

1. It can be guaranteed to store a given vector as an equilibrium point so that every stored prototype is retrievable.
2. It does not result in symmetric interconnection structure, i.e., $A_{ij} \neq A_{ji}$ in general, so that it is easy for the practical circuit implementations.

4.3.1 Middle-mapping learning algorithm

Now, let us show how the middle-mapping learning method is availably used in a cellular neural network to improve its properties so that it is possible to guarantee the stationary of the stored prototypes and make it easily to be implemented by circuits.

Assume that there are S stored prototypes $\mathbf{o}^{(i)} \subset R^{N\times M}, i = 1, 2, \cdots, S$ for a 2-dimensional cellular associative memory with $N \times M$ cells, each element is either $+1$ or -1.

Depending on Definition 4.1, in the case of $B = 0$ and $I = 0$, the state of an associative memory network $\mathbf{y}$ will be called equilibrium point, if the following relation is met.

$$y_i(k)g_i[\mathbf{y}(k)] \geq 1 \tag{4.23}$$
$$\forall i \in \{1, 2, \cdots, n\}$$

where, $g_i[\,\cdot\,]$ is the transfer operator of the cell $c(i)$ obtained from the state equation (4.1).

Depending on this description, the equilibrium condition about the kth sub-prototype $\mathbf{f}_i^{(k)}$ stored in neighborhood $N_r(i)$ could be provided by

$$f_i^{(k)} \cdot g_i[\mathbf{f}_i^{(k)}] \geq 1 \tag{4.24}$$
$$k = 1, 2, \cdots, S$$

For each cell $c(i)$ in DTCNN, there exists direct connections only with the cell $c(j)$ in its neighborhood $N_r(i)$, but for other cells outside the neighborhood $N_r(i)$, the element A_{ij} must be equal to 0 so that we can rewrite the equation (4.1) as follows,

$$\begin{aligned} \frac{1}{R_x} v_i(k+1) &= \sum_{c(j)\in N_c(i)} A_{ij}\, y_{c_j}(k) && (4.25a) \\ y_i(k+1) &= sat(v_i(k+1)) && (4.25b) \\ y_{c_i}(k) &= y_i(k) && (4.25c) \end{aligned}$$

Then, based on the definition 4.2 we get the output sub-vector $\hat{\mathbf{y}}_i$ to correspond to the neighborhood $N_r(i)$ as $\hat{\mathbf{y}}_i$. When $\hat{\mathbf{y}}_i^{(k)}(m) = \mathbf{f}_i^{(k)}$, next relation could be obtained from the equation mentioned above.

$$\begin{aligned} g_i[\mathbf{f}_i^{(k)}] &= g_i[\hat{\mathbf{y}}_i^{(k)}(m)] \\ &= v_i^{(k)}(m+1) \\ &= R_x \sum_{c(j)\in N_r(i)} A_{ij}\hat{y}_j^{(k)} \end{aligned} \tag{4.26}$$

Although A_{ij} is two-dimensional vector, in order to describe above relation as a form of the product of two vectors, we can rearrange it into 1-D vector as follows:

Definition 4.4 *For a cell $c(i)$, we extract all elements A_{ij} for every $c(j) \in N_r(i)$ from the ith row components in matrix A and obtain a vector $\hat{A}_i \in R^p$.*

$$\hat{A}_i = \{\hat{A}_{ij} = A_{ij}; c(j) \in N_r(i)\} \tag{4.27}$$

Then, the equation (4.26) can be equivalently written by

$$g_i[\mathbf{f}_i^{(k)}] = R_x(\hat{A}_i \cdot \mathbf{f}_i^{(k)}) \tag{4.28}$$

From this equation and the equation (4.24), we can obtain the next relation

$$f_i^{(k)} \cdot R_x(\mathbf{f}_i^{(k)} \cdot \hat{A}_i) \geq 1 \tag{4.29}$$

Obviously, if we assume

$$R_x(\mathbf{f}_i^{(k)} \cdot \hat{A}_i) = \lambda f_i^{(k)} \ \textit{and} \ \lambda \geq 1$$

then the above relation can be satisfied.

It means

$$(\mathbf{f}_i^{(k)} \cdot \hat{A}_i) = \frac{\lambda}{R_x} f_i^{(k)} \tag{4.30}$$

For simplicity, we let $\lambda = R_x$, then, we can find that, if the next relation exists

$$(\mathbf{f}_i^{(k)} \cdot \hat{A}_i) = f_i^{(k)} \tag{4.31}$$

the kth sub-prototype will meet the equilibrium condition (4.24).

We can extend (4.31) to all sub-prototypes $\mathbf{f}_i^{(k)}, k = 1, 2, \cdots, S$ stored in the neighborhood $N_r(i)$, such that they satisfy the equilibrium condition. Then a vector equation could be provided:

$$\mathrm{F} \cdot \hat{A}_i = \mathcal{F} \tag{4.32}$$

where

$$\mathrm{F} = [\mathbf{f}_i^{(1)}, \mathbf{f}_i^{(2)}, \cdots, \mathbf{f}_i^{(S)}]^T$$

$$\mathcal{F} = [f_i^{(1)}, f_i^{(2)}, \cdots, f_i^{(S)}]^T$$

The form of the equation (4.32) is similar to the formula of the projection learning rule[19], where the connection weight matrix is the orthogonal projection matrix into the subspace spanned by the prototype vector families. In our formula (4.32), $\hat{A}_{ij}$ is a $(2r+1)^2$-length vector which maps a prototype $\hat{\mathbf{y}}_i^{(k)}$ into its middle element $y_i^{(k)}$. Therefore, formula (4.32) could be termed as middle-mapping learning rule or pseudo-projection learning algorithm.

Let $p = (2r+1)^2$, then $\hat{A}_i \in R^p$, $\mathcal{Y} \in R^S$, $\mathrm{Y} \in R^{S \times p}$. Formula (4.32) is a $S \times p$ system of linear non-homogeneous equations. Based on the theory of linear algebra, if the rank of the coefficient matrix Y is equal to the rank of augmented matrix $(\mathrm{Y}|\mathcal{Y})$, i.e. $rank(\mathrm{Y}) = rank(\mathrm{Y}|\mathcal{Y})$, then, the equation (4.32) must have solutions. When $p > rank(\mathrm{Y}) = rank(\mathrm{Y}|\mathcal{Y})$, there are solutions more than one, the number of solutions depends on the value of $p - rank(\mathrm{Y})$.

Obviously, we can find that from formula (4.28), at the least, it has a solution $\hat{A}_i = (0, 0, \cdots, 0, 1, 0, \cdots, 0)$, i.e. the self-feedback coefficient of a cell is equal to 1, $\hat{A}_{ii} = 1$, while others are zero.

Note that in Hopfield network, the self-feedback coefficient of a cell is limited to equal zero, it causes that we can not directly apply the middle-mapping learning method to Hopfield associative memory, although this method has some advantages than the outer product method. In our cellular associative memory, the self-feedback coefficient of a cell is not equal to zero, the number is greater than $1/R_x$, so that if the value of R_x is selected big enough, the condition (6) can be satisfied easily.

On the basis of generalized matrix inverse, we can obtain

$$\hat{A}_i = \mathrm{F}^+ \cdot \mathcal{F} \tag{4.33}$$

where F^+ is the Moore-Penrose pseudoinverse[22] of F. If and only if $\mathrm{Y}\mathrm{Y}^+\mathcal{Y} = \mathcal{Y}$, then, the system of equations (4.32) has solutions. Even if there are no solution on (4.32), the $\hat{\mathbf{A}}_{ij}$ denoted in (4.33) is still a solution of the least squares problem $min\|\mathrm{Y}\hat{\mathbf{A}}_{ij} - \mathcal{Y}\|$

4.3.2 Iteration learning algorithm

In the preceding section, we present a middle mapping learning algorithm for cellular associative memory networks. It is well known that, in general case, the couping weight matrix shown as (4.33) can be directly calculated by pseudo-inverse. But on the other hand, the interconnection vector $\hat{A}_i$ can be obtained conveniently, without solving high-order matrix inversion, but using an iterative algorithm[22,24]. After some times iterations, the exact solutions of system (4.33) is yielded, the number of iterations is equal to the number of sub-prototype vectors. This kind of computation is typical of a learning process: once the synaptic matrix has been computed from a given set of prototype vectors, the addition of one extra item of knowledge does not require that the whole computation is performed again. One just has to carry out one iteration, starting from the previous matrix, so that the computational efficiency can be improved.

In this section, we describe an iterative learning algorithm which can be incorporated into our middle mapping algorithm to extend its computation ability. First, we give some definition and theorems.

Definition 4.5 *Let c_1, c_2, $\cdots$, c_m be column vectors in an n-dimensional space, and introduce a $n \times m$ matrix*

$$C_m = (c_1|c_2|\cdots|c_m)$$

where the jth column is c_j.

In terms of C_{m-1} and c_m, above matrix can be written as

$$C_m = (C_{m-1}|c_m) \qquad\qquad m = 2, 3, \cdots$$

Theorem 4.4 *If $C_{m+1} = (C_m|c_{m+1}) \in R^{n\times(m+1)}$ where $C_m \in R^{n\times m}$ is the submatrix of C_{m+1} consisting of the first m columns, and $c_{m+1} \in R^n$ is the $(m+1)$th column of C_{m+1}, then*

$$C_{m+1}^{+} = \begin{bmatrix} C_m^{+}(I - c_{m+1}k_{m+1}^T) \\ k_{m+1}^T \end{bmatrix} \tag{4.34}$$

where

$$k_{m+1}^T = \begin{cases} \frac{C_m^{+T} C_m^+ c_{m+1}}{1+\|C_m^+ c_{m+1}\|^2} & \text{if}(I - C_m C_m^+)c_{m+1} = 0 \\ \\ \frac{(I-C_m C_m^+)c_{m+1}}{\|(I-C_m C_m^+)c_{m+1}\|^2} & \text{otherwise} \end{cases} \tag{4.35}$$

Proof: The detail of the proof can be found in Ref.22. □

In that theorem, $(I - C_m C_m^+)c_{m+1} = 0$ if and only if c_{m+1} is in the space spanned by $c_1, c_2, \cdots, c_m$. Hence, k_{m+1} is defined by the first part of (4.35) if and only if c_{m+1} is not a linear combination of $c_1, c_2, \cdots, c_m$.

Theorem 4.5 *For the pseudo-inverse C_m^+ of C_m, we have*

$$(C_m^T)^+ = (C_m^+)^T$$

Proof: The proof can also be found in Ref.22.

After then, we define

$$\begin{aligned} \mathrm{F}_m &= [\mathbf{f}_i^{(1)}, \mathbf{f}_i^{(2)}, \cdots, \mathbf{f}_i^{(m)}]^T \\ \mathcal{F}_m &= [f_i^{(1)}, f_i^{(2)}, \cdots, f_i^{(m)}]^T \end{aligned}$$

and describe $\hat{A}_i$ derived from m sub-prototypes as $\hat{A}_i^{(m)}$. Here, $\mathbf{f}_i^{(k)}$ is a column vector, $\mathbf{f}_i^{(k)} \in R^p$, and $\mathrm{F}_m \in R^{m\times p}$, $\mathcal{F}_m \in R^m$ and $\hat{A}_i^{(m)} \in R^p$. In this way, the equation (4.33) can be expressed as

$$\hat{A}_i^{(m)} = \mathrm{F}_m^+ \mathcal{F}_m \tag{4.36}$$

It is used to store m sub-prototype vectors $\{\mathbf{f}_i^{(1)}, \mathbf{f}_i^{(2)}, \cdots, \mathbf{f}_i^{(m)}\}$ as stable equilibrium points of a cellular associative memory network. Now suppose we desire to store an additional sub-prototype $\mathbf{f}_i^{(m+1)}$ into the network. According to Theorem 4.4 and Theorem 4.5, we have

$$\begin{aligned} \hat{A}_i^{(m+1)} &= \mathrm{F}_{m+1}^+ \mathcal{F}_{m+1} \\ &= \begin{pmatrix} \mathrm{F}_m \\ \mathbf{f}_i^{(m+1)T} \end{pmatrix}^+ \begin{pmatrix} \mathcal{F}_m \\ f_i^{(m+1)} \end{pmatrix} \end{aligned}$$

$$
\begin{aligned}
&= \left[\begin{array}{c} (\mathrm{F}_m^{T\,+} - \mathrm{F}_m^{T\,+}\mathbf{f}_i^{(m+1)}k_{m+1}^T) \\ k_{m+1}^T \end{array} \right]^T \left(\begin{array}{c} \mathcal{F}_m \\ f_i^{(m+1)} \end{array} \right) \\
&= (\mathrm{F}_m^{+} - k_{m+1}\mathbf{f}_i^{(m+1)^T}\mathrm{F}_m^{+})\mathcal{F}_m + k_{m+1}f_i^{(m+1)} \\
&= \hat{A}_i^{(m)} + \triangle\hat{A}_i^{(m+1)} \qquad (4.37)
\end{aligned}
$$

where

$$\triangle\hat{A}_i^{(m+1)} = k_{m+1}(f_i^{(m+1)} - \mathbf{f}_i^{(m+1)^T}\mathrm{F}_m^{+}\mathcal{F}_m)$$

$$k_{m+1} = \begin{cases} \frac{\mathrm{F}_m^{+}\mathrm{F}_m^{T\,+}\mathbf{f}_i^{(m+1)}}{1+\|\mathrm{F}_m^{T\,+}\mathbf{f}_i^{(m+1)}\|^2} & \text{if } (I - \mathrm{F}_m^{+}\mathrm{F}_m^{T\,+})\mathbf{f}_i^{(m+1)} = 0 \\ \frac{(I-\mathrm{F}_m^{+}\mathrm{F}_m^{T\,+})\mathbf{f}_i^{(m+1)}}{\|(I-\mathrm{F}_m^{+}\mathrm{F}_m^{T\,+})\mathbf{f}_i^{(m+1)}\|^2} & \text{otherwise} \end{cases}$$

In this way, we can learn an additional sub-prototype $\mathbf{f}_i^{(m+1)}$ without affecting the sub-prototype vectors $\{\mathbf{f}_i^{(1)}, \mathbf{f}_i^{(2)}, \cdots, \mathbf{f}_i^{(m)}\}$ already learned by the network.

With the equation (4.37), an iterative learning algorithm can be performed to store all S sub-prototypes as the equilibrium points of the memory networks. It can be summarized as follows:

```
**Procedure :** Iterative Middle Mapping Learning Algorithm
**Begin**
  **for** i:=1 **to** M **do begin**
    **for** j:=1 **to** N **do begin**
      $\mathrm{Y}_1^+ := \hat{\mathbf{y}}_{ij}^{(1)^T} / (\hat{\mathbf{y}}_{ij}^{(1)^T}\hat{\mathbf{y}}_{ij}^{(1)})$;
      $\hat{\mathbf{A}}_{ij}^{(1)} := \mathrm{Y}_1^+\mathcal{Y}_1$;
      **for** m := 1 **to** S − 1 **do begin**
        **if** $(\mathrm{I} - \mathrm{Y}_m^+\mathrm{Y}_m^{T+})\mathbf{y}_{ij}^{(m+1)} = 0$
          **then** $\mathrm{k}_{m+1} := \mathrm{Y}_m^+\mathrm{Y}_m^{T+}\mathbf{y}_{ij}^{(m+1)}/(1 + \|\mathrm{Y}_m^{T+}\mathbf{y}_{ij}^{(m+1)}\|^2)$
          **else** $\mathrm{k}_{m+1} := (\mathrm{I} - \mathrm{Y}_m^+\mathrm{Y}_m^{T+})\mathbf{y}_{ij}^{(m+1)}/\|(\mathrm{I} - \mathrm{Y}_m^+\mathrm{Y}_m^{T+})\mathbf{y}_{ij}^{(m+1)}\|^2$ ;
        $\triangle\hat{\mathbf{A}}_{ij}^{(m+1)} := \mathrm{k}_{m+1}(\mathcal{y}_{ij}^{(m+1)} - \mathbf{y}_{ij}^{(m+1)^T}\mathrm{Y}_m^+\mathcal{Y}_m)$;
        $\hat{\mathbf{A}}_{ij}^{(m+1)} := \hat{\mathbf{A}}_{ij}^{(m)} + \triangle\hat{\mathbf{A}}_{ij}^{(m+1)}$
      **end**
    **end**
  **end**
**End**
```

4.3.3 Illustration example

To illustrate the associative memory ability of the cellular neural network with the middle-mapping learning algorithm, three examples are given bellow.

Example 1:

Let the network consist of a two-dimensional grid with 5×5 neural inner cells and add a ring of dummy cells. The radius of one neighborhood is $r = 1$, so, the number of cells in a neighborhood is equal to $(2r+1)^2 = 9$. The parameters used in our examples are selected as

$$R_x = 100\ k\Omega, \qquad B = 0\ (k\Omega)^{-1}, \quad and \quad I = 0\ mA$$

First, 4 prototypes shown as Figure 4.8 are stored into the cellular associative memory with our learning algorithm.

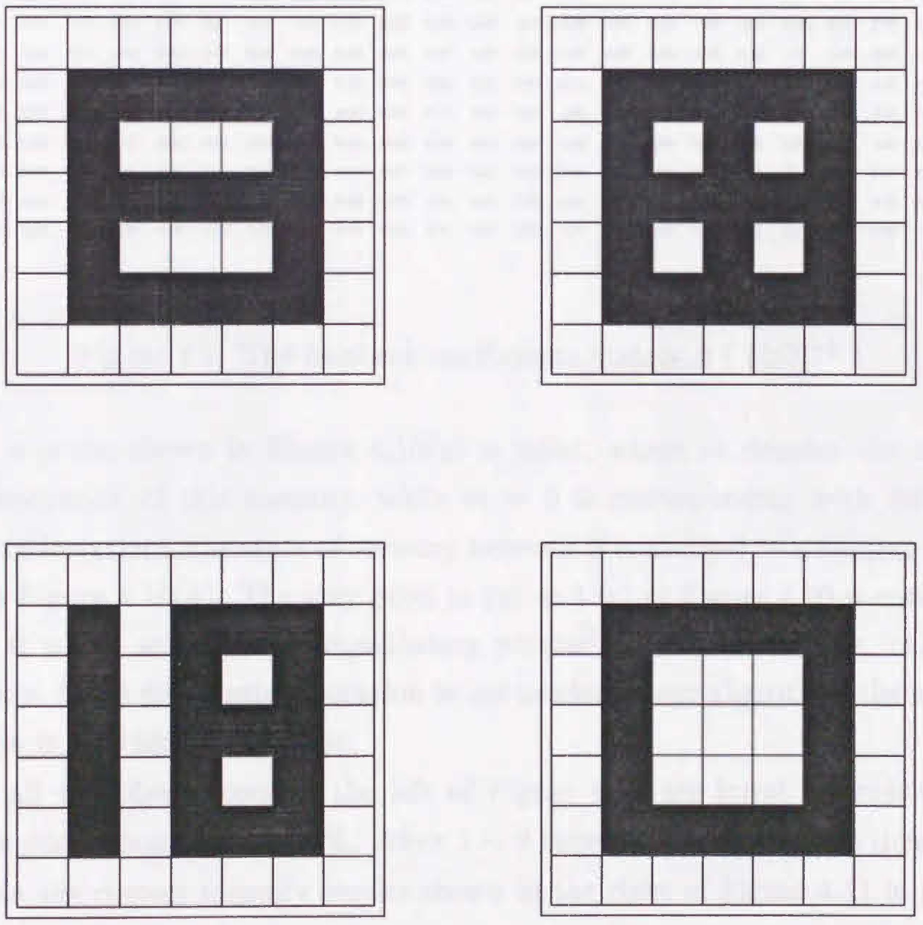


Figure 4.8: Stored 4 prototypes for Example 1

The unsymmetric weight matrix is obtained, which includes the information about the stored prototypes. Figure 4.9 illustrates this weight matrix.

$$\left[\begin{array}{ccccccccccccccccccccccccc}
0.12 & 0.00 & 0.00 & 0.00 & 0.00 & 0.12 & -.12 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.14 & 0.14 & 0.00 & 0.00 & -.14 & 0.00 & -.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.12 & 0.12 & 0.12 & 0.00 & 0.00 & 0.00 & -.12 & 0.12 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.11 & 0.11 & 0.00 & 0.00 & 0.00 & -.11 & 0.11 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.14 & -.14 & 0.00 & 0.00 & 0.00 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-.18 & -.06 & -.18 & 0.00 & 0.00 & -.18 & 0.18 & 0.00 & 0.00 & 0.00 & -.18 & 0.06 & -.06 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & -.16 & -.16 & -.16 & 0.00 & 0.00 & 0.00 & 0.16 & -.16 & 0.00 & 0.00 & 0.00 & 0.00 & -.16 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.12 & 0.12 & 0.00 & 0.00 & 0.00 & -.12 & 0.12 & 0.00 & 0.00 & 0.00 & 0.00 & 0.12 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.12 & -.12 & 0.00 & 0.00 & 0.00 & 0.12 & 0.00 & 0.00 & 0.00 & 0.00 & 0.12 & -.12 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.50 & 0.50 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.50 & 0.50 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -.12 & 0.12 & 0.00 & 0.00 & 0.00 & 0.00 & 0.12 & 0.00 & 0.00 & 0.00 & -.12 & 0.12 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 & 0.14 & -.14 & 0.00 & 0.00 & 0.00 & 0.14 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -.18 & 0.06 & -.06 & 0.00 & 0.00 & -.18 & 0.18 & 0.00 & 0.00 & 0.00 & -.18 & -.06 & -.18 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -.16 & 0.00 & 0.00 & 0.00 & 0.16 & -.16 & 0.00 & 0.00 & -.16 & -.16 & -.16 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.12 & 0.00 & 0.00 & 0.00 & -.12 & 0.12 & 0.00 & 0.00 & 0.00 & 0.12 & 0.12 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.12 & -.12 & 0.00 & 0.00 & 0.00 & 0.12 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -.14 & 0.00 & -.14 & 0.00 & 0.00 & 0.00 & 0.14 & 0.14 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -.12 & 0.12 & 0.00 & 0.00 & 0.12 & 0.12 & 0.12 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -.11 & 0.11 & 0.00 & 0.00 & 0.00 & 0.11 & 0.11
\end{array}\right]$$

Figure 4.9: The feedback coefficients matrix A ($(k\Omega)^{-1}$)

Then, a probe shown in Figure 4.10(a) is input, where m denotes the number of current operation of this memory, while $m = 0$ is corresponding with initial state. After three iterations, the state of memory network is converged to a nearest prototype shown as Figure 4.10(d). The grey pixel in (b) and (c) of Figure 4.10 is corresponded to $y_i = 0$ which are unstable equilibrium points[10]. In real circuits, they are no measurable. Since differential operation is not needed in our algorithm, the computing simulation is very simple and fast.

When all 4 probes shown in the left of Figure 4.11 are input alternatively, they have distortion about 32% $\sim$ 40%. After 3 $\sim$ 9 times of the associative iterations, we can obtain the correct memory results shown in the right of Figure 4.11 in which the complement contents are recovered.

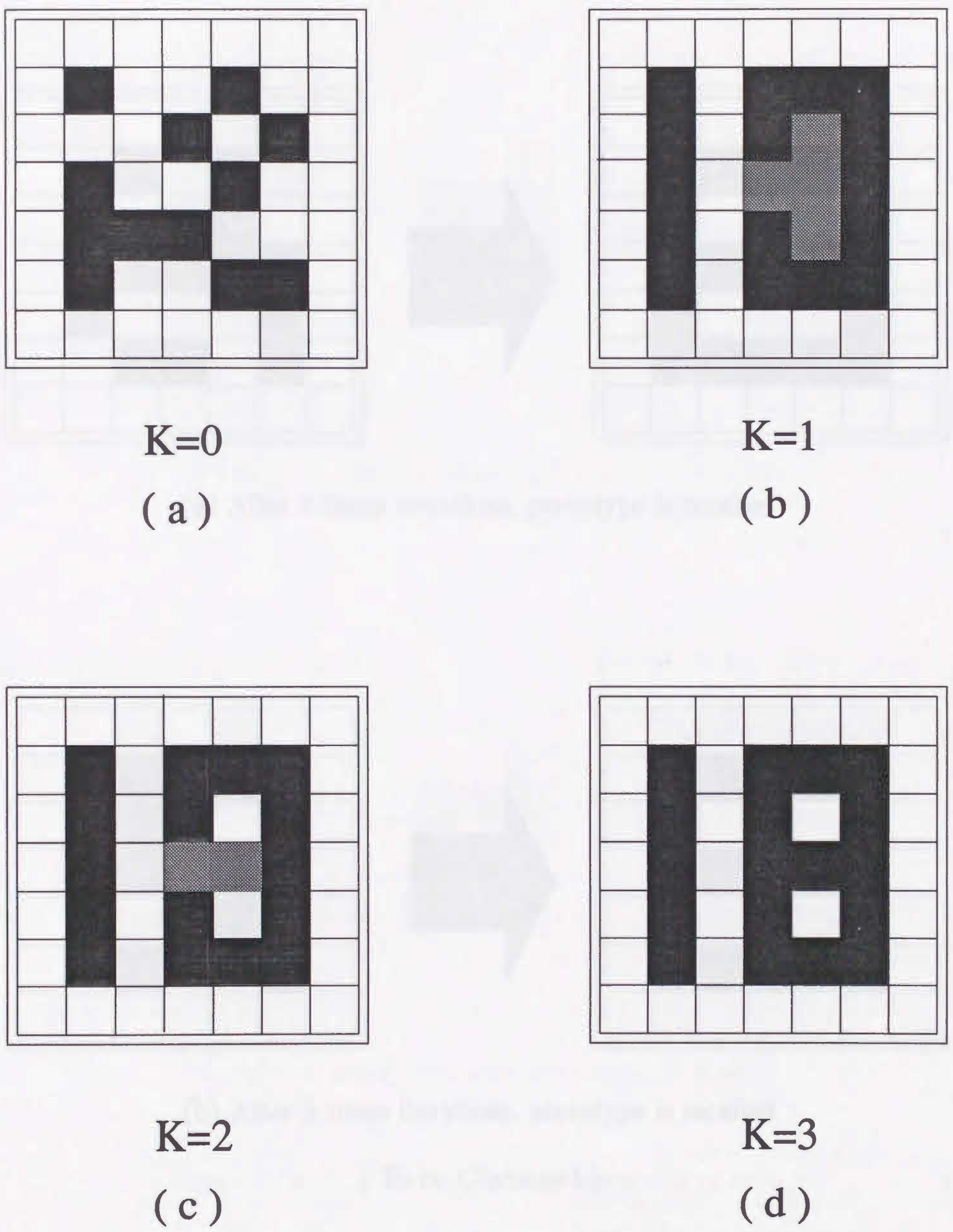
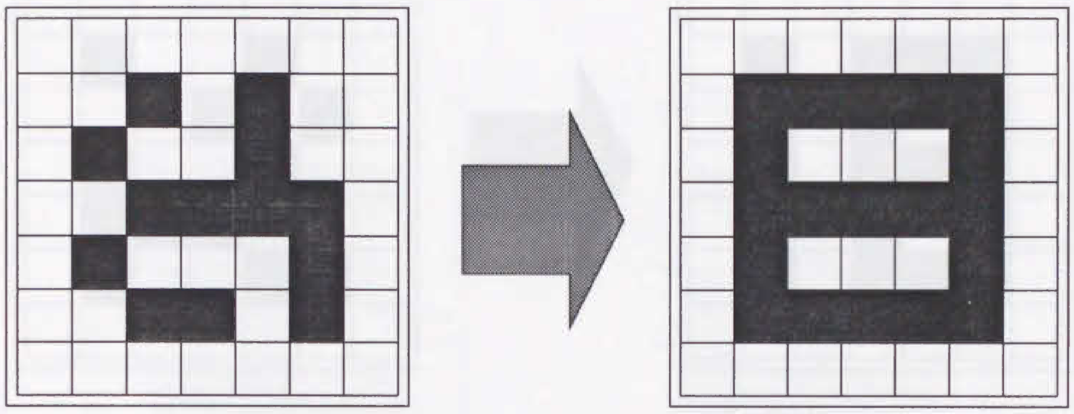
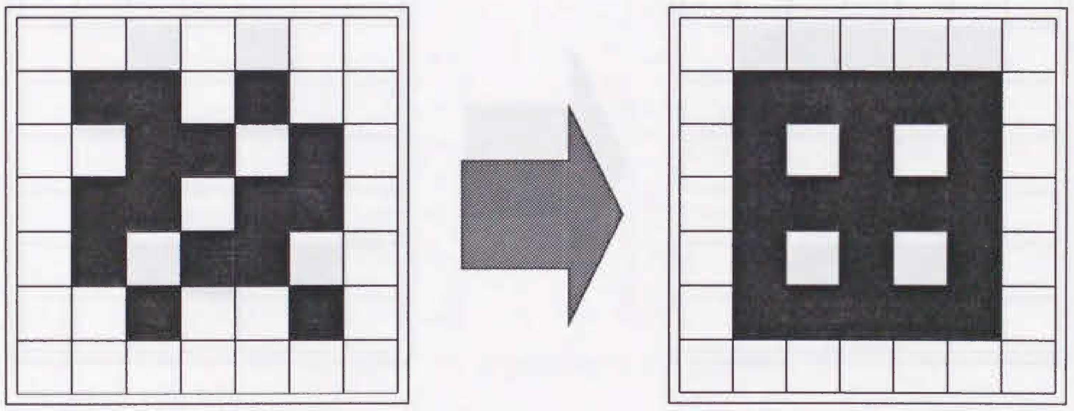
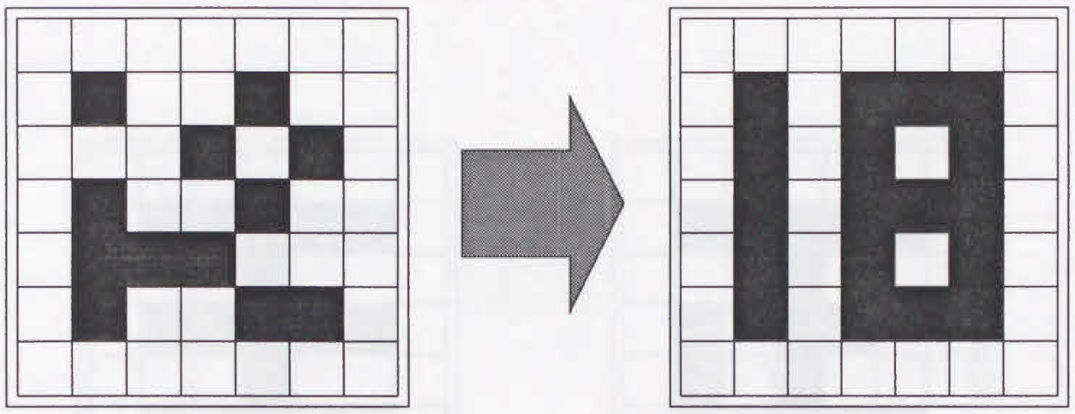


Figure 4.10: The retrieval process for one probe

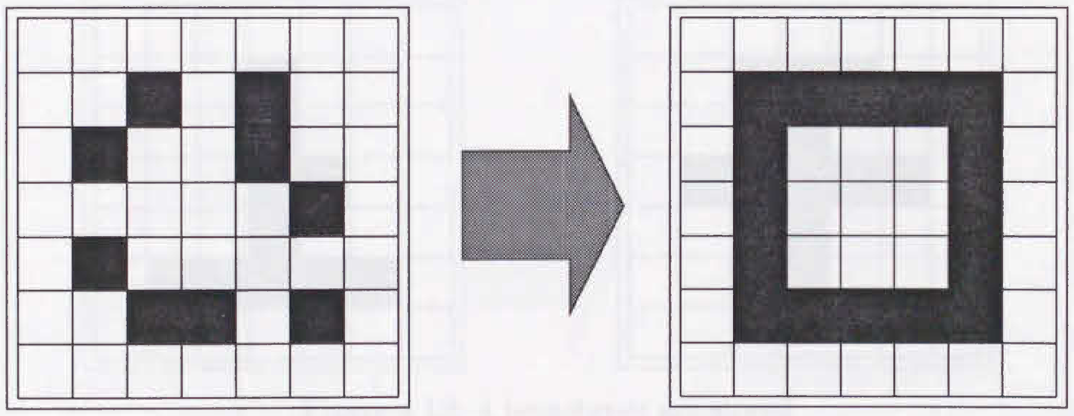

(a) After 9 times iterations, prototype is recalled


(b) After 3 times iterations, prototype is recalled

(To be Continued)



(c) After 5 times iterations, prototype is recalled



(d) After 4 times iterations, prototype is recalled

Figure 4.11: 4 probes and associative memory results

Example 2:

In this example, first, 4 prototypes shown in Figure 4.12 are stored and the interconnection weigh matrix is produced.

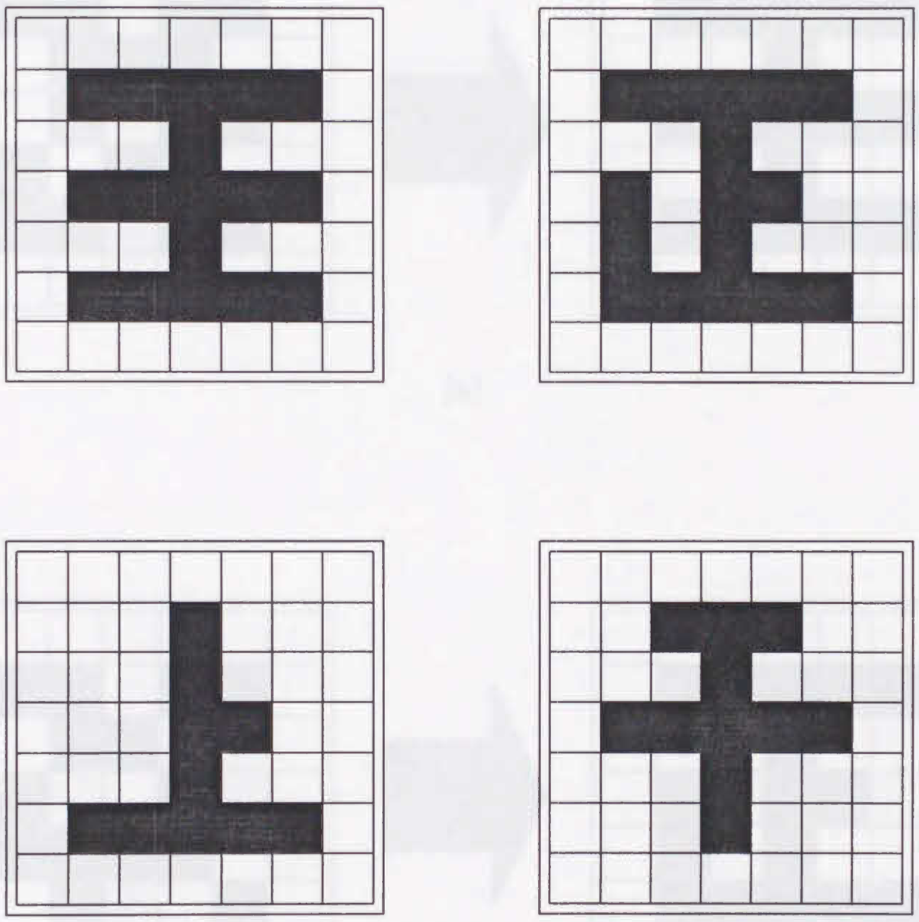
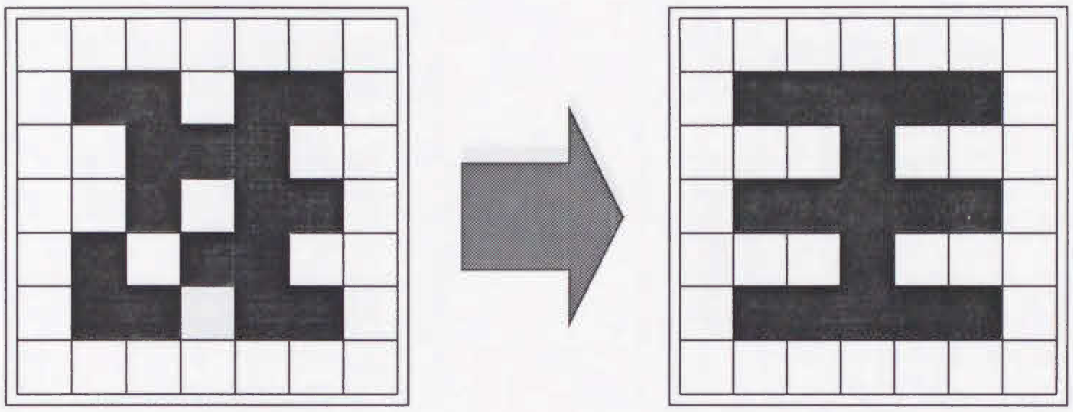
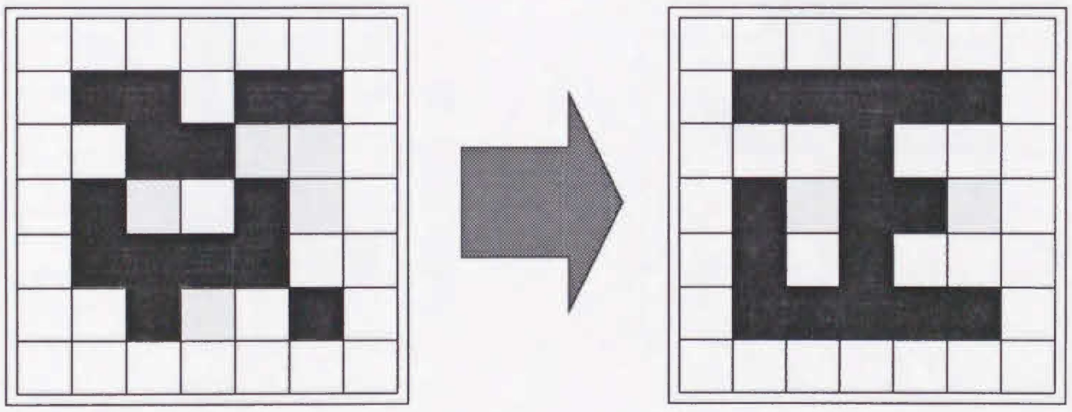


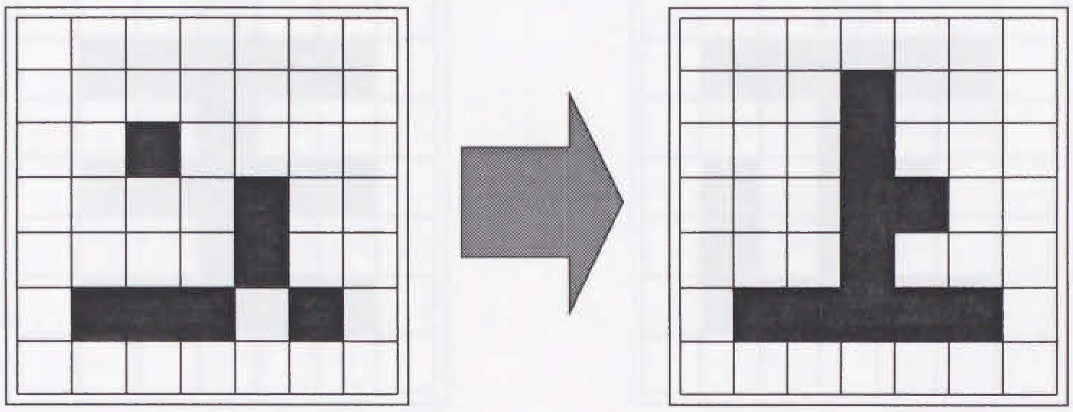
Figure 4.12: 4 prototypes are stored

Then, 4 probes shown in the left of Figure 4.13 are input respectively, whose distortions are 28% $\sim$ 36%. After some times of the updating, the complement stored patterns are retrieved which are shown in the right of Figure 4.13.

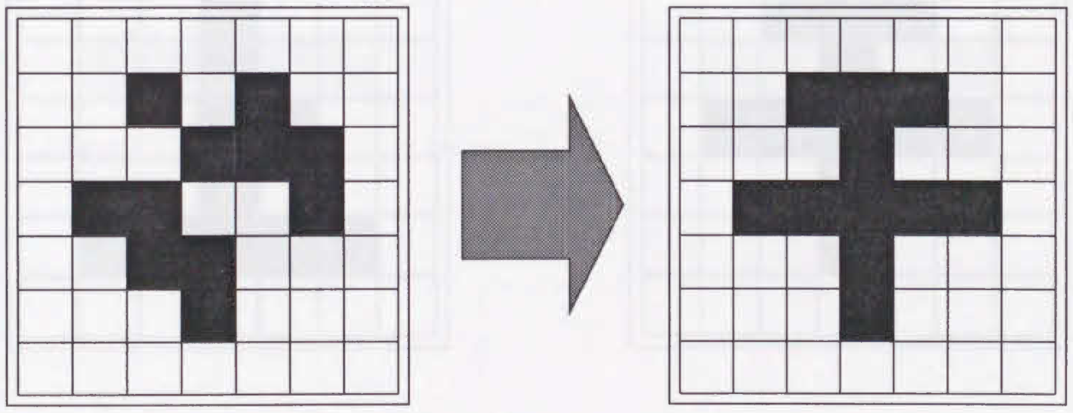

(a)


(b)

(To be continued)



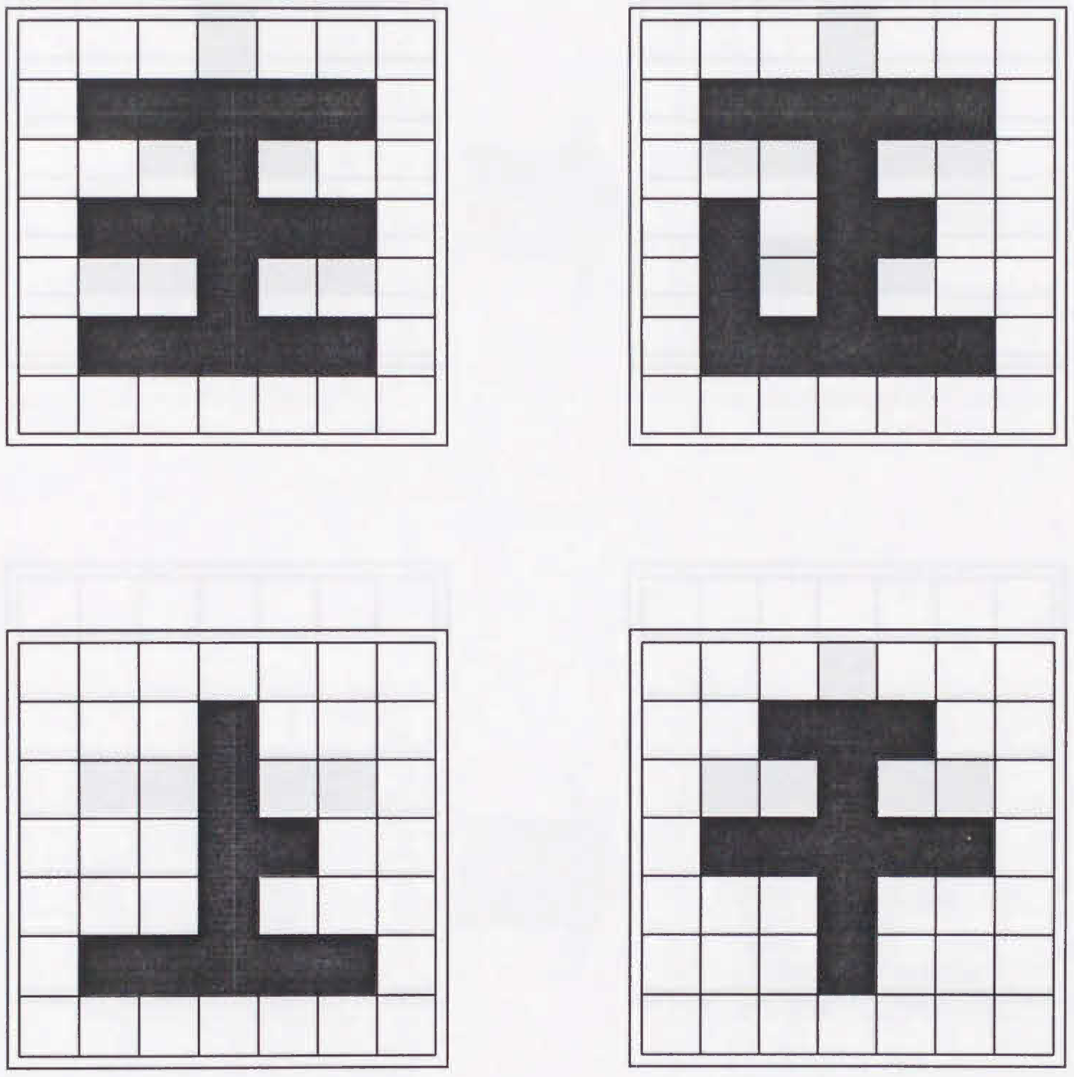
(c)



(d)

Figure 4.13: The probes and retrieval results

Next, 8 prototypes shown in Figure 4.14 are stored.



(To be continued)

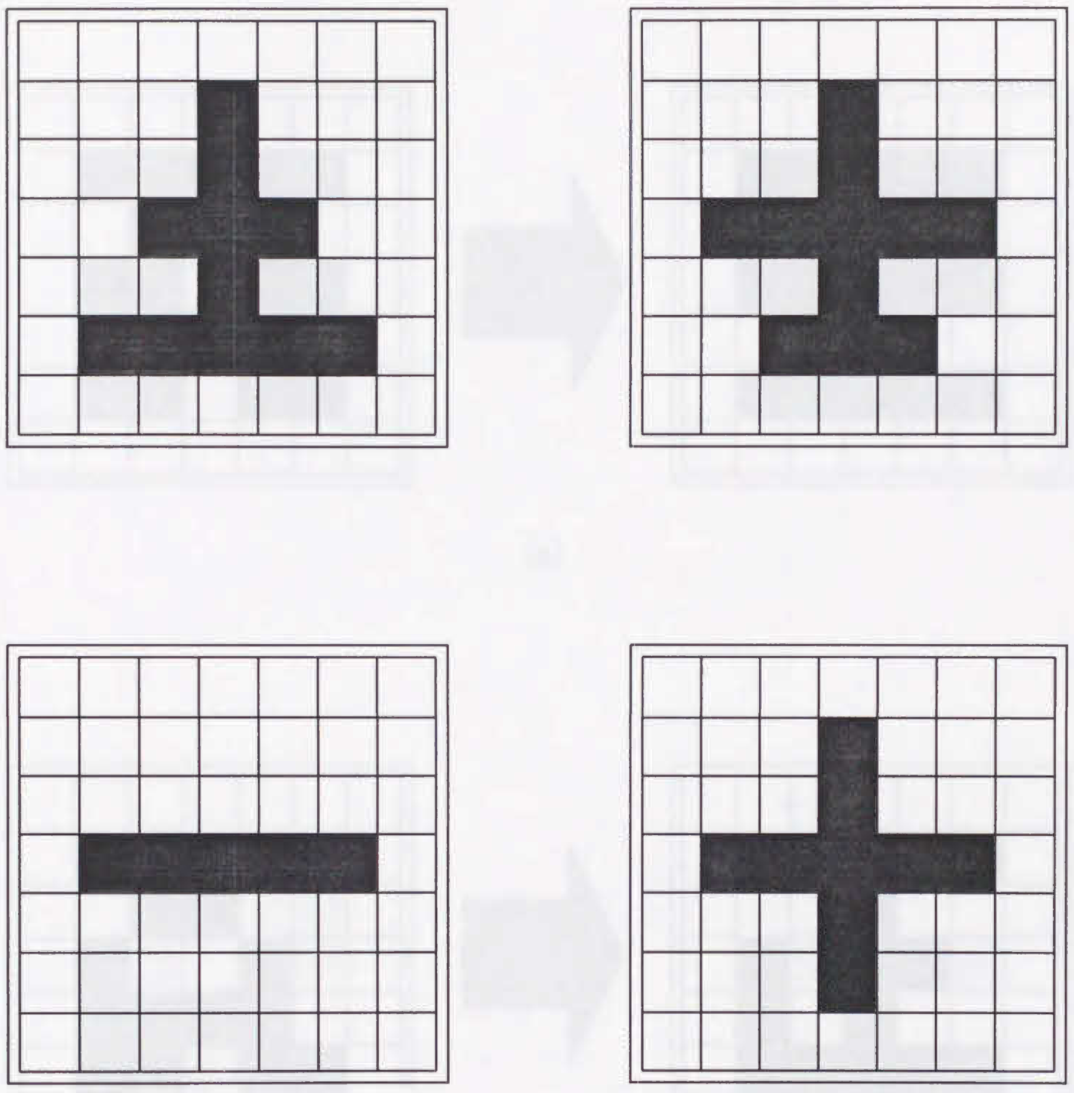
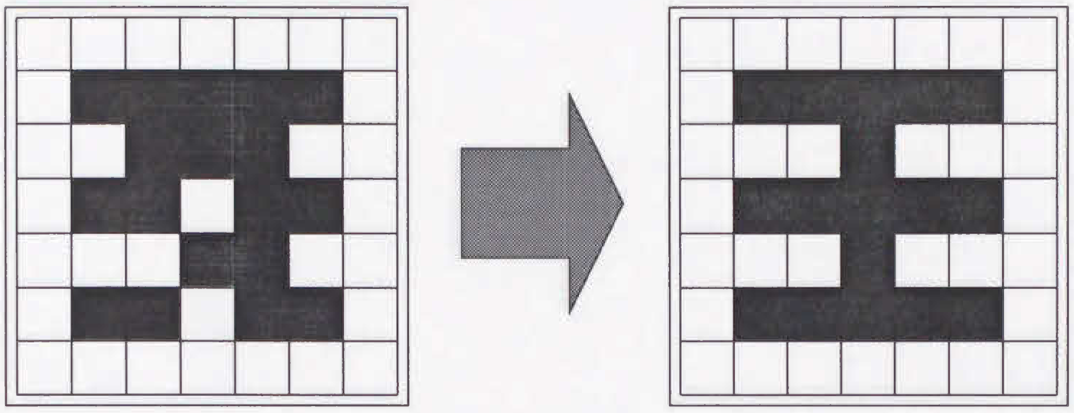


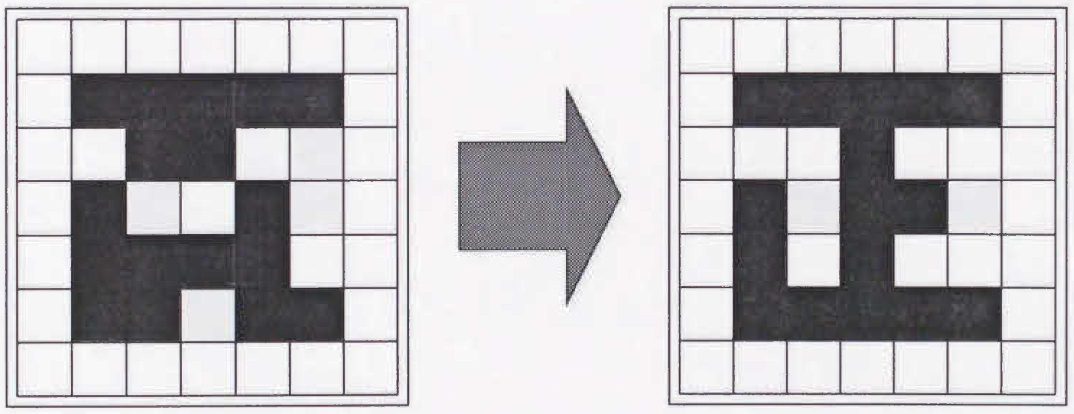
Figure 4.14: 8 prototypes are stored

Then, probes shown in the left of Figure 4.15 are selected respectively as the initial states of the network. After some times of iterations, we can obtain desired associative results as the right of Figure 4.15, but the distortions of the probes is reduced into

12% $\sim$ 24%. It illustrates that the convergent ability gets smaller in this case.

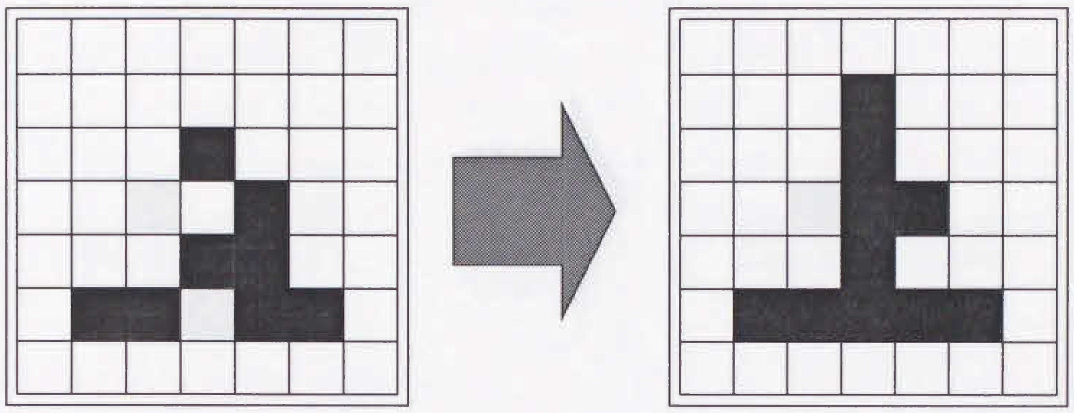


(a)

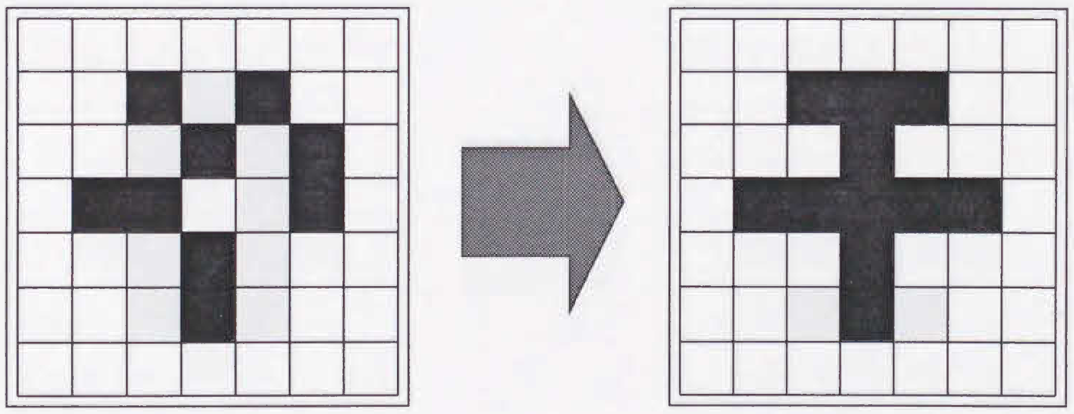


(b)

(To be continued)

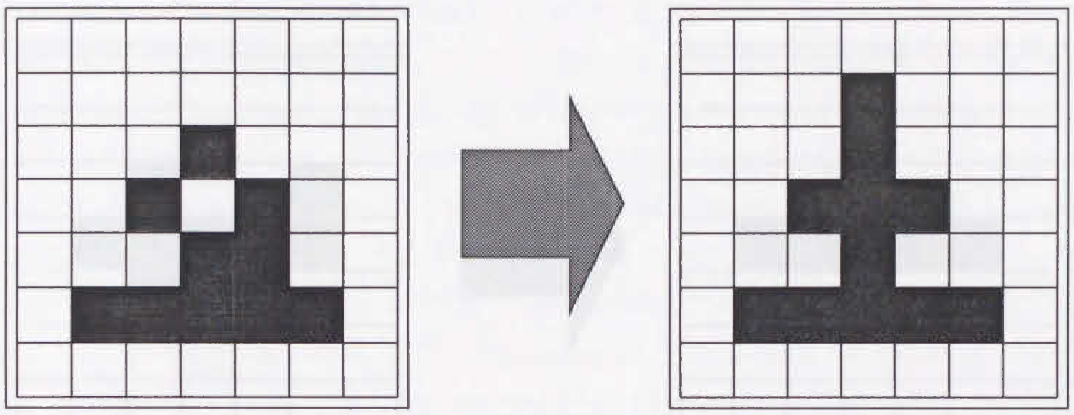


(c)

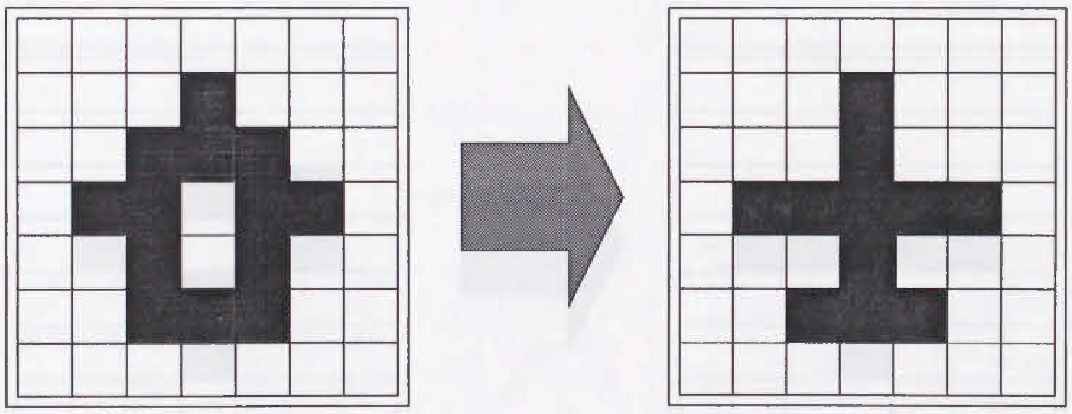


(d)

(To be continued)

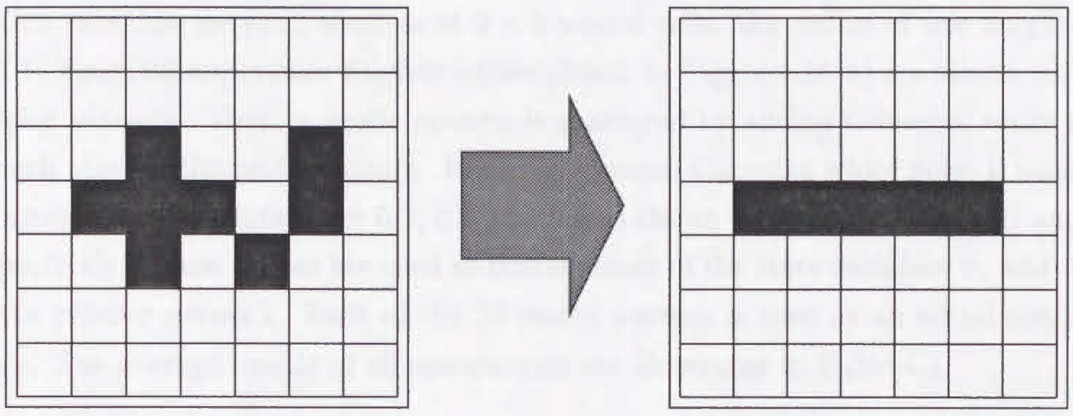


(e)

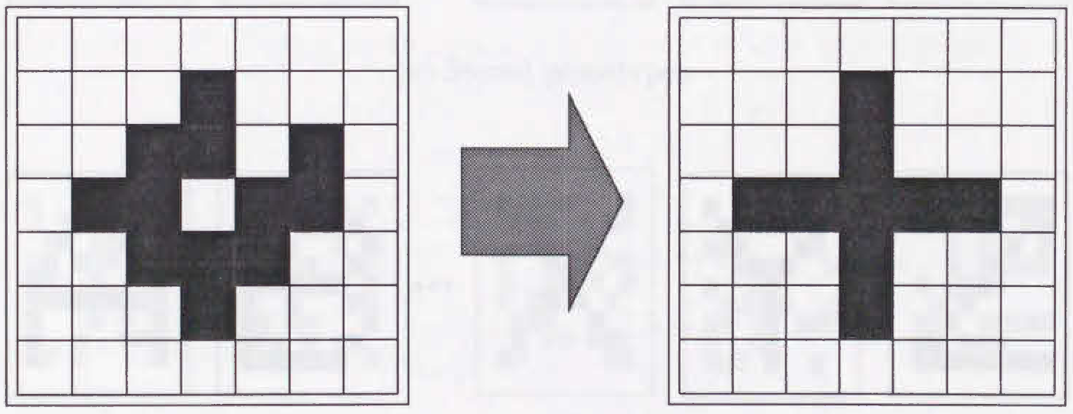


(f)

(To be continued)



(g)

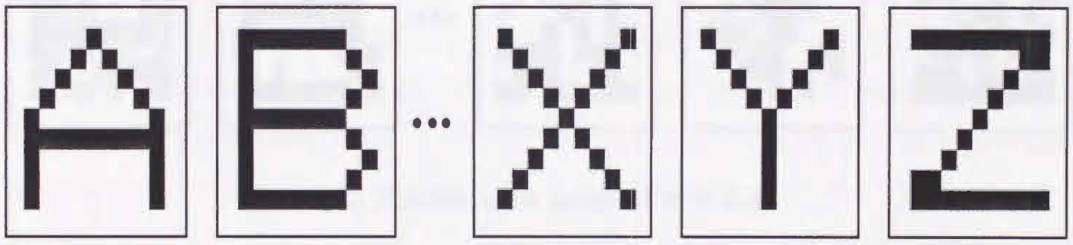


(h)

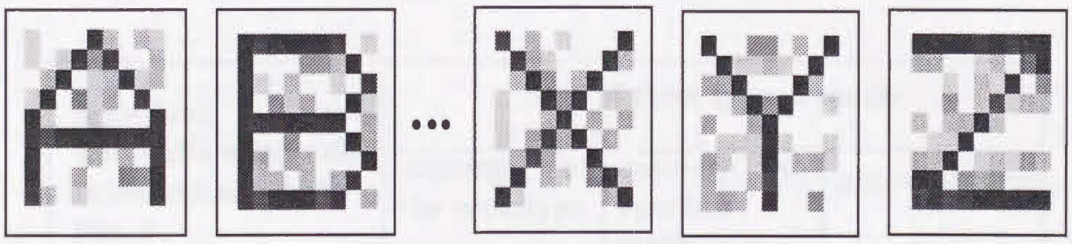
Figure 4.15: Input probes and the associative results

Example 3:

Next, the ability of cellular associative memory with middle- mapping algorithm to recover prototypes from a probe signal mixed with Gaussian white noises is displayed. In this case, the network consists of 9×9 neural cells, the radius of one neighbor is still 1. First, 26 upper-case English letters shown in Figure 4.16(a) are stored into the cellular network. Then, a probe pattern is generated by adding Gaussian white noise to each pixel in the probe pattern. Hence, zero-mean Gaussian white noise is used but its mean square deviation are 0.2, 0.3 and 0.4 as shown in Figure 4.16(b), (c) and (d) respectively. These probes are used as initial values of the state variables $\mathbf{v}_x$ and input to the cellular network. Each of the 26 stored pattern is used as an initial condition twice. The average results of all calculations are illustrated in Table 4.1.



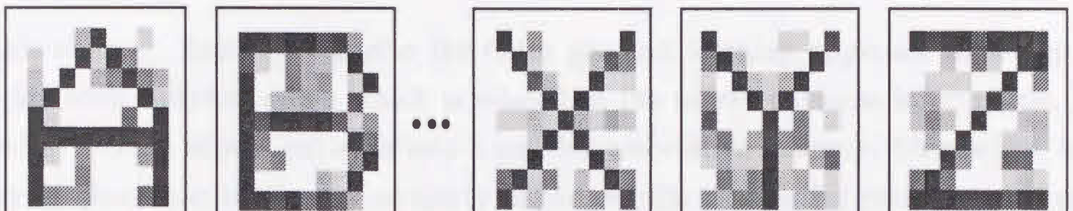
(a) Stored prototypes



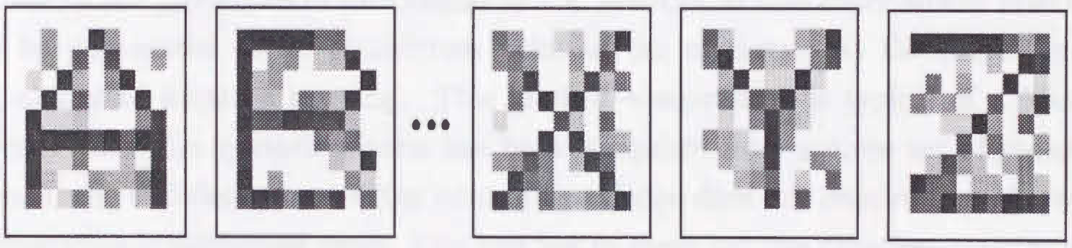
(b) Probes with noise of $\sigma = 0.2$

(To be continued)

From these calculation results, it can be found that, there are some spurious states in a cellular associative memory, while the mean square deviation of noise mixed in



(c) Probes with noise of $\sigma = 0.3$



(d) Probes with noise of $\sigma = 0.4$

Figure 4.16: Stored prototypes and initial probes with Gaussian noises

zero-mean Gaussian white noise with mean square deviation σ	Rate of three type of results		
	converge to the prototype	converge to spurious states	not convergent
$\sigma = 0.2$	100.0	0.0	0.0
$\sigma = 0.3$	98.1	1.9	0.0
$\sigma = 0.4$	71.2	28.8	0.0

Table 4.1: The simulation results for 26 English characters storing in a cellular associative memory by the middle mapping algorithm

probes is larger, it may be possibly to converge to some spurious states.

4.4 Conclusion

In this chapter, first, we describe the outer product learning approach to set up the weights with suitable values which is related to the object patterns information, it is called as storing object patterns into a cellular associative memory. Meanwhile, some analyses about the stationary property of the cellular associative memory with outer product learning rule are taken. A condition is presented which ensure the stored patterns as the stable states of a cellular associative memory. After then, a middle-mapping learning algorithm for cellular associative memory is presented, which makes full use of the properties of the cellular neural network so that every stored prototype can be guaranteed as an equilibrium point of our memory. At the same time, it has ability of iterative learning. This kind of computation is typical of a learning process: once the synaptic matrix has been computed from a given set of prototype vectors, the addition of one extra item of knowledge does not require that the whole computation is performed again. One just has to carry out one iteration, starting from the previous matrix, so that the computational efficiency can be improved. Besides, its implementation with circuits is more feasible because the weight matrix is not symmetric.

Since the synchronous updating rule is used in both of them, their associative speed very fast compared to the Hopfield associative memory.

From the simulating results, we can find that, when the number of the stored prototypes is increased or the distortion in a probe is strong, the associative ability is decreased and the probability of converging to spurious states is increased. It is similar with the situation in the other types of associative memory networks. But in a cellular associative memory, it is believable that we can extend the size of a neighborhood of our cellular associative memory to improve its associative ability. Unfortunately, the realization of circuits is get more difficult at the same time and the manufacture cost is risen. Using space-varying method[14] to select suitable neighborhood size meeting the specific requirement may solve this problem. This is a future problem. More detail researchs will be taken in the near future.

Reference

[1] T.Kohonen, *Self-Organization and Associative Memory*, Berlin: Spring-Verlag, 1988(2th Ed.).

[2] J.Zurada, *Artificial Neural Systems*, West publishing Company, 1992.

[3] A.N.Michel and J.A.Farrell, "Associative memories via artificial neural networks", *IEEE Contr. Syst. Mag.*, vol.10, pp.6-17, Apr. 1990.

[4] S.Amari, "Neural theory of association and concept formation", *Biol. Cybern.*, vol.26, pp.175-185, 1977

[5] J.J.Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proc. Nat. Acad. Sci. USA*, vol.79, pp.2554-2558, Apr. 1982.

[6] J.J.Hopfield, "Neurons with gradeed response have collective computational properties like those of two-state neurons", *Proc. Nat. Acad. Sci. USA*, vol.81, pp.3088-3092, 1984.

[7] J.Li, A.N.Michel and W.Porod, "Analysis and synthesis of a class of neural networks: variable structure systems with infinite gain", *IEEE Trans. Circ. Syst.* vol.CAS-36, pp.713-731, 1989.

[8] K.Jinno and T.Saito, "Analysis and synthesis of continuous-time hysteretic neural networks", *IEICE Trans. A*, vol.J-75-A, no.3, pp.552-556, 1992.

[9] T.Nishi and N.Takahashi, "On associative memory by neural networks from the circuit theoretic view point", *Proc. Workshop on Nonlinear Theory and Appl.*, pp.9-16, June 8, 1993.

[10] L.O.Chua and L.Yang, "Cellular neural network: theory", *IEEE Trans. Circ. Syst.*, vol.CAS-35, pp.1257-1272, 1988.

[11] Special Issue Cellular Neural Networks, *Inter. J. Cir. Theor. Appl.*, vol.20, No.5, Sept.-Oct., 1992.

[12] *Proceedings of the IEEE Inter. Workshop on CNN and Their Application*, CNNA-92, Germany, Sept. 1992.

[13] *Proceedings of the IEEE Inter. Workshop on CNN and Their Application*, CNNA-90, Budapest, Oct. 1990.

[14] G.Martinelli and R.Perfetti, "Associative design using space-varying CNN", *Proceeding of the IEEE Inter. Workshop on CNN and Their Application*, CNNA-92, Germany, pp.117-122, Sept. 1992.

[15] N.N.Aizenberg and I.N.Aizenberg, "CNN based on multi-valued neuron as a model of associative memory for grey-scale images", *Proceedings of the IEEE Inter. Workshop on CNN and Their Application*, CNNA-92, Germany, pp.36-41, Sept. 1992.

[16] S.Tan, J.Hao and J.Vandewalle, "Cellular neural networks as a model of associative memories", *Proceedings of the IEEE Inter. Workshop CNNA-90*, Budapest, pp.26-35, 1990.

[17] C.He and A.Ushida, "Stationary analysis of the associative memory with the cellular neural network", *Proceedings of Symp. on Nonlinear Theor. and Appl.*, Hakone, pp.131-134, July 1992.

[18] R.J.McEliece, F.C.Posner, E.R.Rodemich and S.S.Venkatesh, "The capacity of the Hopfield associative memory", *IEEE Trans. Inform. Theor.*, vol.IT-33, pp.461-482, 1987.

[19] L.Personnaz, I.Guyon and G.Dreyfus, "Collective computational properties of neural networks: new learning mechanism", *Phys. Rev. A*, vol.34, pp.4217-4228, Nov. 1986.

[20] D.Hebb, *The Organization of Behavior*, New York: Wiley, 1949.

[21] L.N.Cooper, F.Liberman and E.Oja, "Theory for the acquisition and loss of neuron specificity in visual cortex", *Biol. Cybern.*, vol.33, No.9, pp.9-28, 1979.

[22] A.Albert, *Regression and the Moore-Penrose Pseudoinverse*, New York: Academic, 1972.

[23] W.S.McCulloch and W.Pitts, "A logical calculus of the ideas immanent in neurons activity", *Bull. Math. Biophys.*, vol.5, pp.115-133, 1943.

[24] G.Yen and A.N.Michel, "A learning and forgetting algorithm in associative memories: results involving pseudo inverses", *IEEE Trans. Circ. Syst.*, vol.CAS-38, pp.1193-1205, 1991.

Chapter 5

Applications in Image Processing

5.1 Introduction

In previous chapter, we apply our discrete-time cellular neural network to associative memory, which is a problem to find the connection weights so that a given set of prototypes o^1, o^2, $\cdots$, o^S are the stable fixed points bedded in our discrete-time associative memory network with prescribed size of basins of attraction. That is a fixed-point programming problem.

Differing from it, in this chapter, we will apply our DTCNN to image processing with another view point. It is to consider DTCNN as a spatial operator. With appropriate choice of the connecting weights, the network can operate as a differentiator, an integrator or even more complexer operator, which include the cooperative operation, the competitive operation and the mixed operation. Although many similar tasks can be performed by current digital image processing techniques, DTCNN will operate faster than the former, generally, since it is a parallel operator.

We list the state equation of our DTCNN as follows:

$$\begin{aligned} v_i(k+1) &= R_x[\sum_{j=1}^{n} A_{ij}\,\mathrm{sat}(v_j(k)) + \sum_{j=1}^{n} B_{ij}\,u_j + I_i\,] \\ & \quad k = 0,1,2,\cdots, \quad \forall i \in \{1,2,\cdots,n\} \end{aligned} \tag{5.1}$$

This equation can be interpreted as a two-dimensional operator to map an image, described by $\mathbf{v}(k)$ into another one, represented by $\mathbf{v}(k+1)$. Obviously, this operator is nonlinear since $\mathrm{sat}(\mathbf{v}(k))$ in (5.1) is a nonlinear function. In general, some times recursive operations are required to get a desired result after an initial probe is inputted to the network. The operator for mapping an image does not achieve at once, but takes some times of iterations so it is a space-time operator really.

Based on experiences and theorem analyses, it is known that, the more elements a template has, the complexer operation it can perform. But VLSI realization and template design or learning approach of the network will become difficult for larger neighborhood. From the practical point of view, the neighborhood is always chosen to be as small as possible. The typical radius for a neighborhood is 1 or 2, which is corresponding to 3×3 neighborhood or 5×5 neighborhood, respectively.

Since each cell just connects directly its near cells in a discrete-time cellular neural network, the A matrix in Equation (5.1) is a sparse matrix, $A_{ij} = 0$, $\forall c_j \notin N_r(i)$. Hence, for one updating, it can only make use of the local image information. When the global character of an image is required, our DTCNN can be updated n times to obtain the global information from the image. It is well-known propagation property, which means that the pixel value of the output object image can be indirectly affected by a large neighbor region of the input image after n times updatings. This property can be illustrated by replacing $v_i(k)$ in (5.1) iteratively down to $v_i(k-1)$. Then we get

$$\begin{aligned} v_i(k+1) &= R_x \sum_{j=1}^{n} A_{ij}\text{sat}[\,R_x \sum_{l=1}^{n} A_{jl}\text{sat}(v_l(k-1)) + \hat{I}_j\,] + \hat{I}_i \\ & \qquad \forall i \in \{1, 2, \cdots, n\} \end{aligned} \tag{5.2}$$

It is easy to see that the state value $v_i(k+1)$ is not only affected by the value v_j though the no-zero weight coefficients A_{ij} $\forall c(j) \in N_r(i)$ directly, but also affected indirectly by the value v_l in previous updating moment because of the no-zero weight coefficients A_{jl} $\forall c(l) \in N_r(j)$ and $\forall c(j) \in N_r(i)$. In this way, the radius of receiving information region for the cell $c(i)$ is wided to two times of the original radius of the neighborhood in the network. When we iterative down k times down to $v_i(0)$, which coincides with the input image, we have

$$\begin{aligned} v_i(k) &= \sum_{c(j) \in N_{kr}(i)} g_{ij}^{k}(v_j(0)) \\ & \qquad \forall i \in \{1, 2, \cdots, n\} \end{aligned} \tag{5.3}$$

where g_{ij}^{k} is a nonlinear function, related with connection coefficients of the cells between $c(i)$ and $c(j)$. Here, we can find that the neighborhood $N_{kr}(i)$ is k times larger than $N_r(i)$. Of course, when the updating times k is large enough, the neighborhood $N_{kr}(i)$ will eventually cover the entire image. Therefore, the propagation property of DTCNN makes it possible to make use of some global features of the input image. On the other hand, the local properties are still preserved with the closer neighbors having more effects than those farther away.

5.2 Feature extraction

Feature extraction is an important problem in image processing. In this section, we illustrate the ability to realize the edge extraction of hand-writing Chinese characters and pictures with our DTCNN. By the edge extraction, we get and remain the main information coved in original messages, but the data volume and stored space size have been reduced. On the other hand, it is available in the recognition of hand-writing Chinese characters.

First, a 48x64 image composed by a diamond shape element and four square shape elements is illustrated in Figure 5.1.

The circuit parameter $R_x = 100\ k\Omega$ and 1-neighborhood are used, the template T_A T_B and T_I are chosen in which both the feedback and control operators are non zero[1].

$$T_A = \begin{array}{|c|c|c|}\hline 0.0 & 0.0 & 0.0 \\ \hline 0.0 & 2.0 & 0.0 \\ \hline 0.0 & 0.0 & 0.0 \\ \hline \end{array} \qquad T_B = \begin{array}{|c|c|c|}\hline -0.5 & -0.5 & -0.5 \\ \hline -0.5 & 1.0 & -0.5 \\ \hline -0.5 & -0.5 & -0.5 \\ \hline \end{array} \qquad T_I = -1.50$$

Table 5.1: The Template for the edge extraction

Using the input image in Figure 5.1 both as the input signal and the initial state of DTCNN, after 3 times iterative updatings, the result is obtained as shown in Figure 5.2, which is just that we want to get.

It is known that the same tasks have been performed by the continuous-time CNN [1], there a continuous-time CNN is used for extracting the edges of a 16x16 diamond image or a 16x16 square image, but the differential operations are carried, in addition of that, the iterations number is about 57. For out DTCNN, the complicated and time consumed differential operation is avoided, the iteration number is 3, it is only about 5% of the continuous-time CNN. Therefore, it is found that, when we realize a CNN

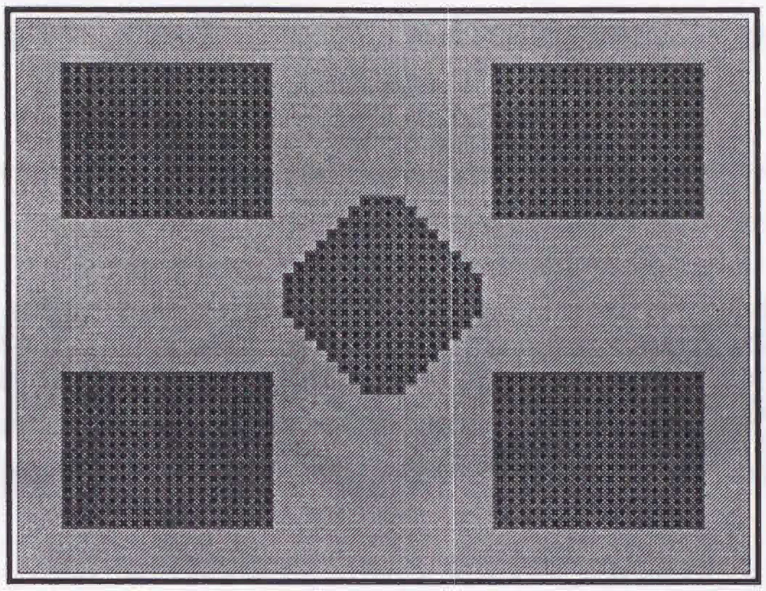


Figure 5.1: An image composed by one diamond and four squares

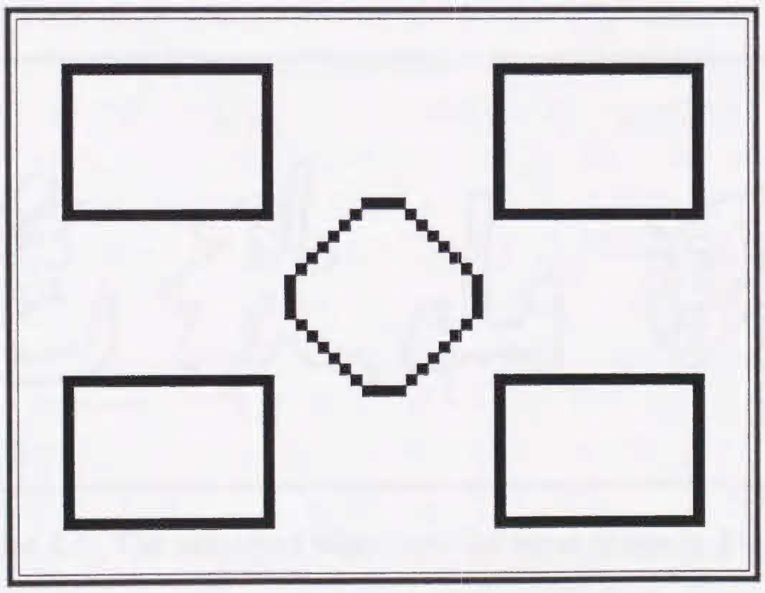


Figure 5.2: The extracted edge from the input image in Fig. 5.1

by a software simulation programs, in some cases of image processing, DTCNN is more powerful and advantageous than the continuous-time CNN.

After then, a picture of hand-writing Chinese characters shown in Figure 5.3 is chosen as both as the input signal and the initial state of DTCNN.

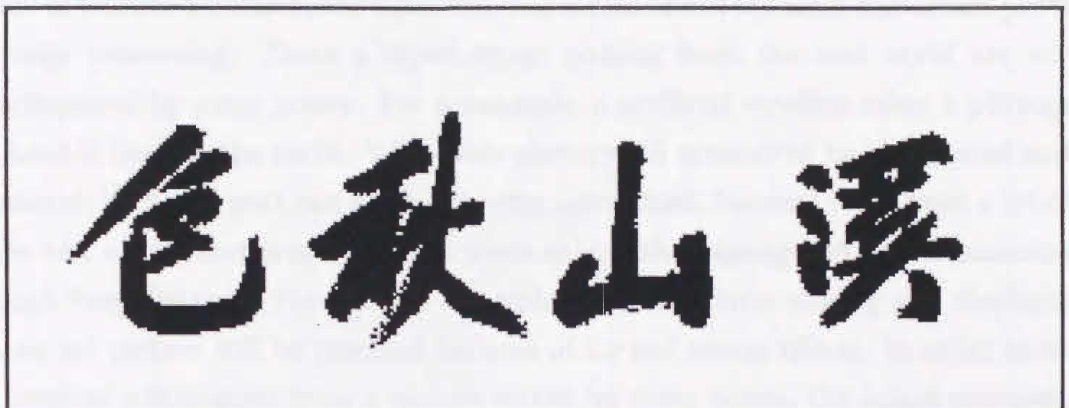


Figure 5.3: A picture of Chinese characters (Zhou SHEN 1427 – 1509)

Using the same circuit parameters and the templates T_A T_B and T_I mentioned above, after 3 times iterative updatings, the precise result is shown in Figure 5.4.

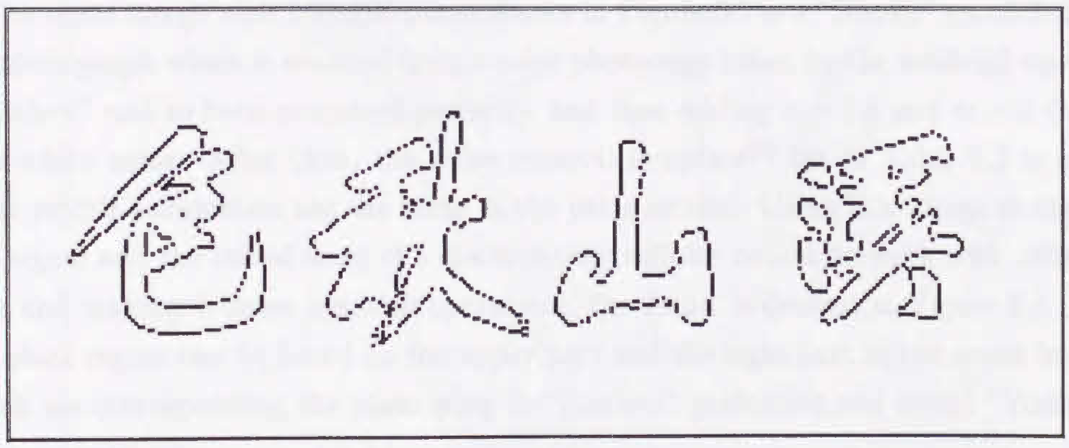


Figure 5.4: The extracted edge from the input image in Fig. 5.3

Moreover, using the same circuit parameters and the same template in Table 5.2, but using the Chinese picture in Figure 5.5 as the input signal and the initial state, just taking 3 times updates, the desired result is derived successfully and, is shown as Figure 5.6.

5.3 Noise removal

Beside of feature detection, the noise removal is also one of the most important problems in image processing. Since a input image coming from the real world are usually superimposed by some noises. For a example, a artificial satellite takes a photograph and send it back to the earth. When this photograph is received by the ground station, in general, its detail part can not be directly recognized, because there exist a lot of hot noises and other interference sources when taking the photograph and transmitting it through long distance. For another example, after long time storing and displaying, a famous art picture will be smeared because of air and steam effects. In order to recove the original information from a picture mixed by some noises, the image processing of the noise removal is required. In fact, the noise removal technique by digital filter, or digital image processing with digital computers, was been developed some years ago and has been widely applied until now. But our DTCNN will be more powerful to the noise removal image processing, because it is a parallel array with faster calculating speed and, can realize real time processing. In this section, we concentrate on the noise removal for an artificial satellite photocopy and a Chinese picture by DTCNN.

The input image with 180x260 pixels shown in Figure 5.7 is a "Sikoku" monochromical photograph which is scanned from a color photocopy taken by the artificial satellite "Landsat" and to been processed perfectly, and then adding $\sigma = 0.8$ and $m = 0$ Gaussian white noise. After then, the noise removal template[1] list in Table 5.3 is used, other circuit parameters are the same as the prior section. Using this image as the input signal and the initial state of a discrete-time cellular neural network with 180x260 cells and making 6 times iterative operations, the result is derived as Figure 5.8. The big block region can be found on the upper part and the right part in the result image, which are corresponding the plain lying in "Kagawa" prefecture and round "Yoshino" river, respectively.

In the next example of noise removal, the same circuit parameters and the noise removal template are chosen, but the input image is a Chinese picture "Yellow Mountain" with 380x200 pixels and adding $\sigma = 0.7$ $m = 0$ Guassian white noise shown as in Figure 5.9. After 6 times iterative operations, the result can be gotten as Figure 5.10.

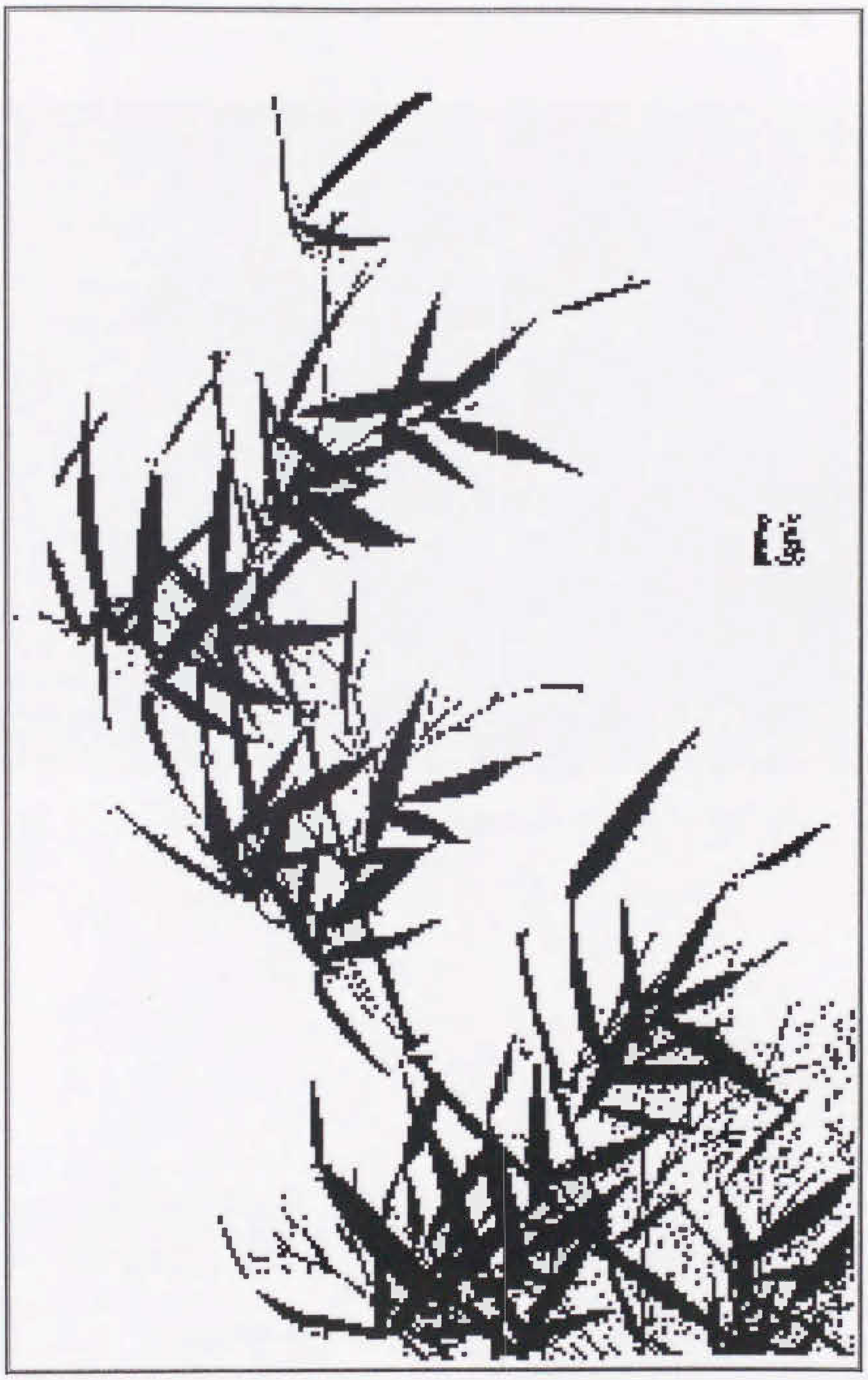


Figure 5.5: A Chinese picture "Listening Bamboo" (Zhen-ming Wen 1470 – 1560)

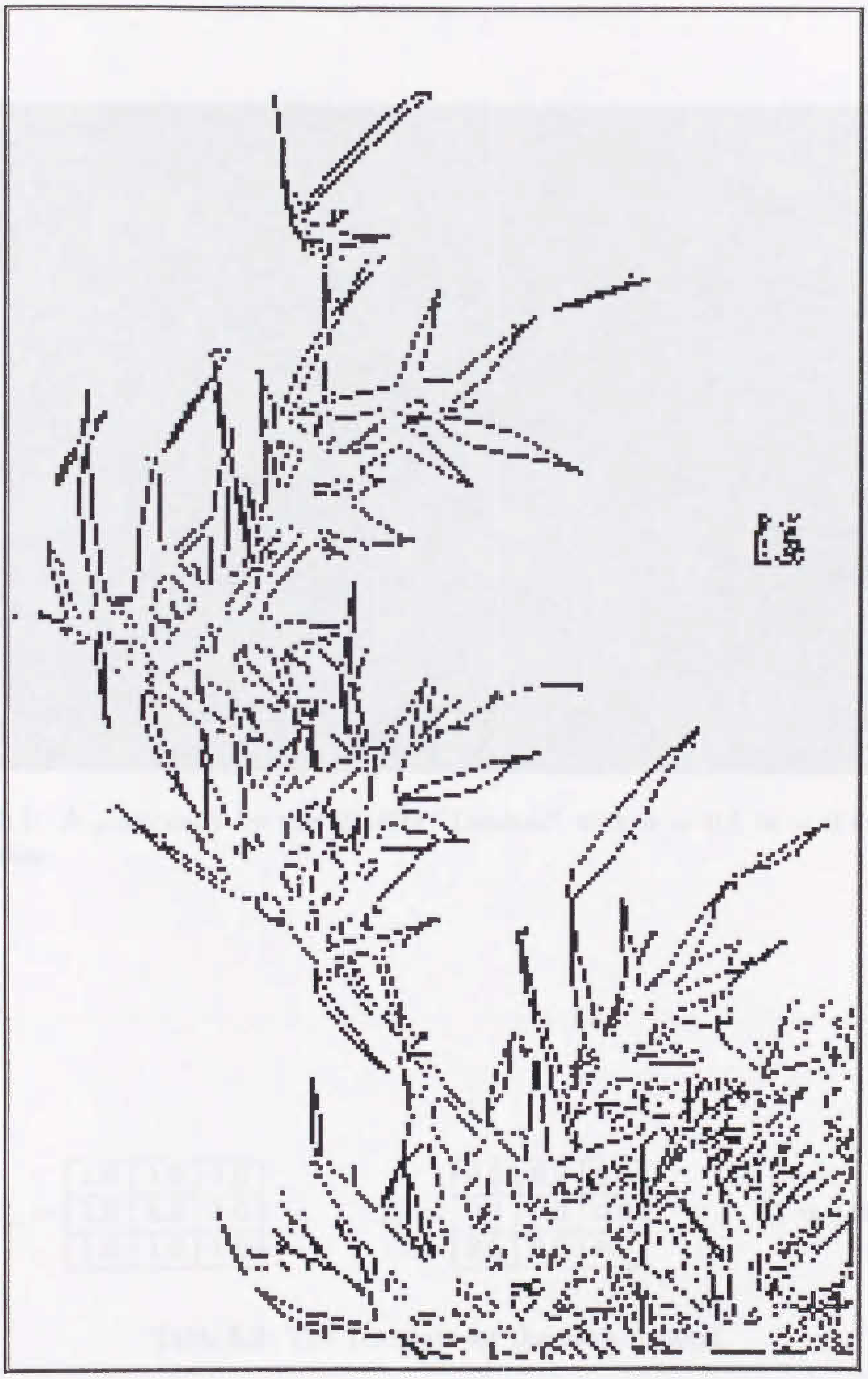


Figure 5.6: The extracted edge from the input image in Fig. 5.5

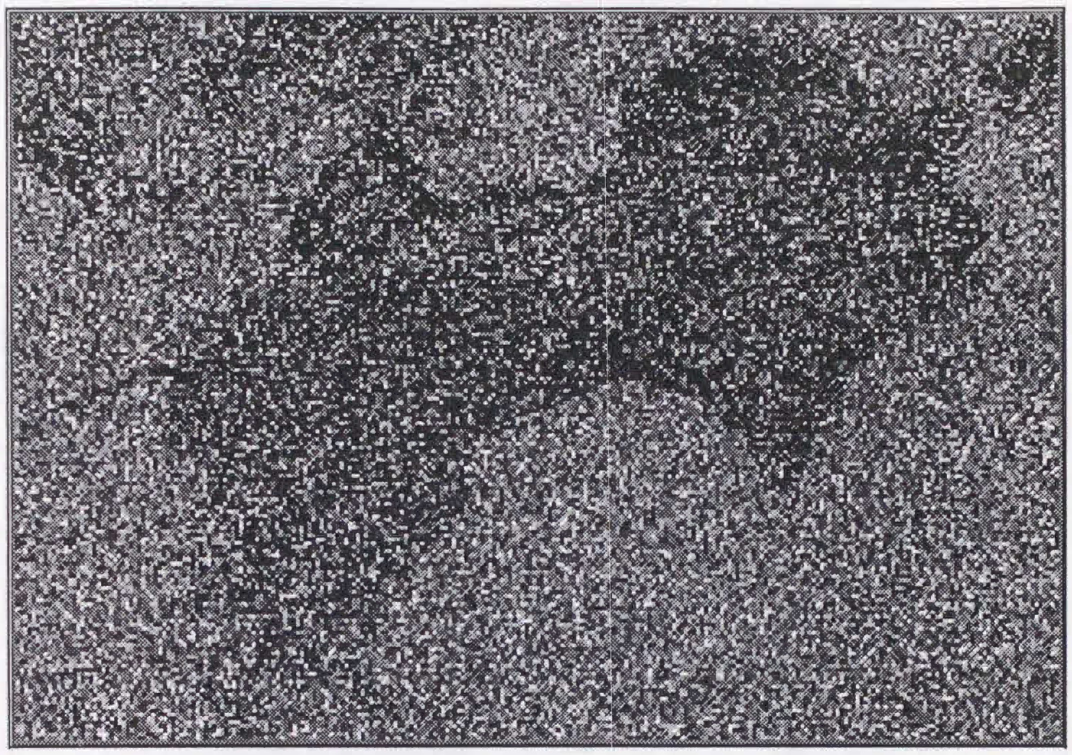


Figure 5.7: A photocopy by the satellite "Landsat" with $\sigma = 0.8$ $m = 0$ Gaussian white noise

$T_A =$

1.0	1.0	1.0
1.0	8.0	1.0
1.0	1.0	1.0

$T_B =$

0.0	0.0	0.0
0.0	0.0	0.0
0.0	0.0	0.0

$T_I = 0.00$

Table 5.2: The Template for the noise removal

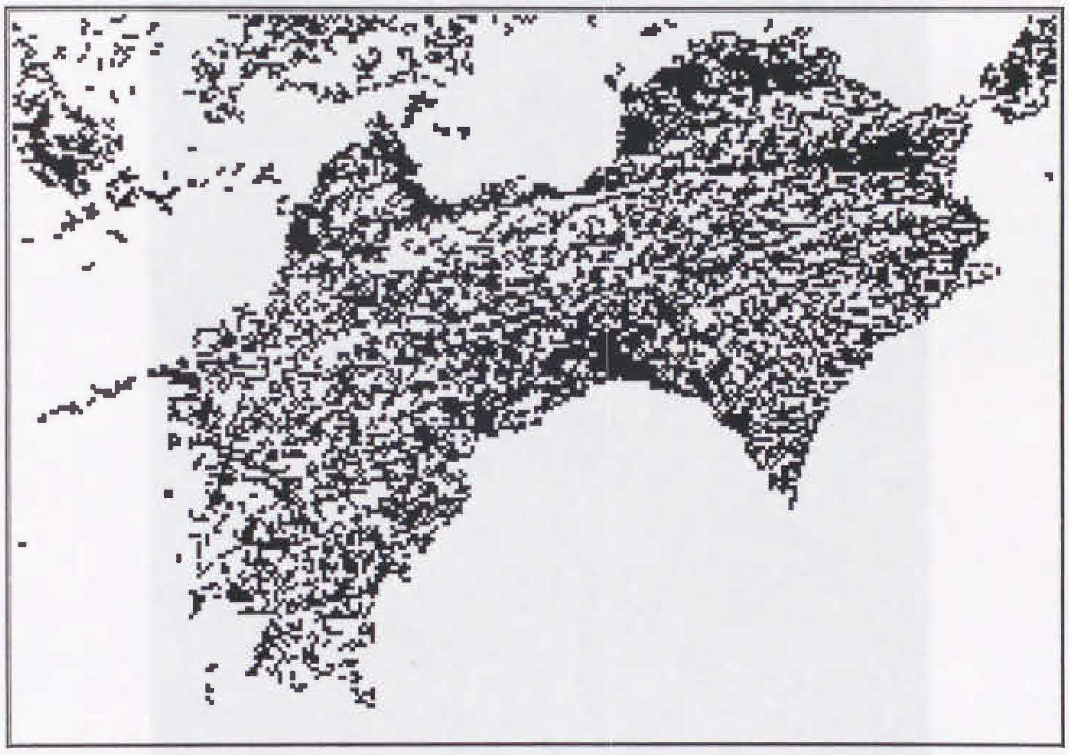


Figure 5.8: Result image from Fig. 5.7 after noise removing

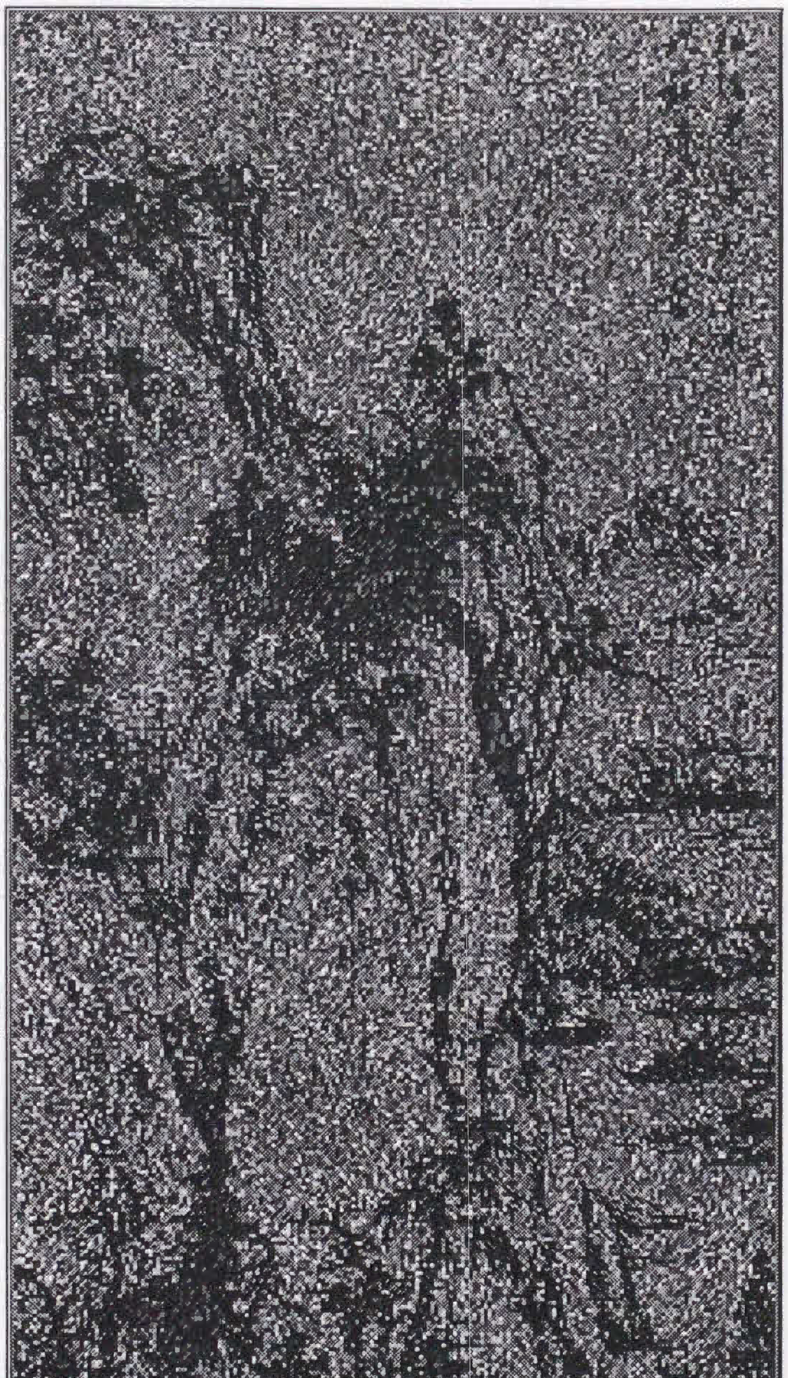


Figure 5.9: A Chinese picture "Yellow Mountain"(Zhou SHEN 1427 – 1509) with $\sigma = 0.7$ and $m = 0$ Gaussian white noise

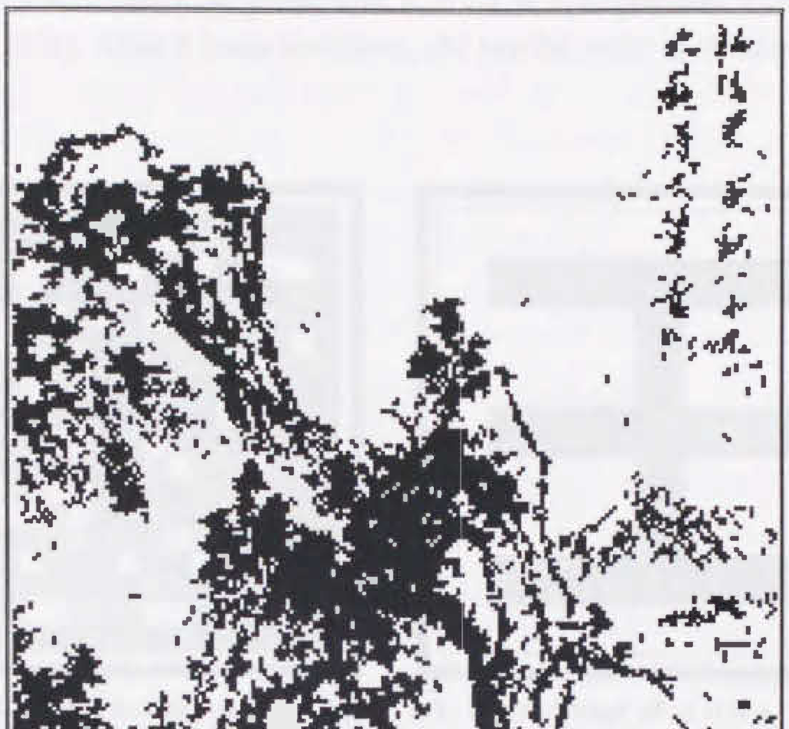


Figure 5.10: Result image from Fig. 5.9 after noise removing

Finally, using the same circuit parameters and the templates, we make noise removal processing to a 16x16 character image with $\sigma = 0.2$ $m = 0$ Guassian white noise shown as in Figure 5.11(a). After 3 times iterations, the precise result is obtained as in Figure 5.11(b).

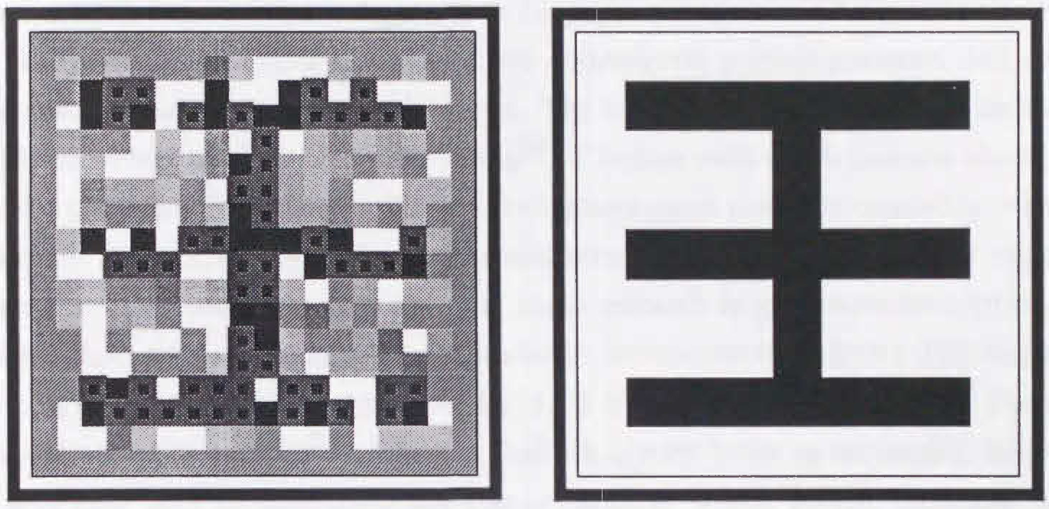

(a) A 16x16 probe image with $\sigma = 0.2$ and $m = 0$ Gaussian white noise (b) Result image after noise removing

Figure 5.11: The noise removing for a 16x16 Chinese character with $\sigma = 0.2$ and $m = 0$ Gaussian white noise

In this section, we give three examples to apply our DTCNN for noise removal image processing. Although the some examples of noise removal for Chinese characters with 16x16 pixels are illustrated in Ref. [1], but their processing is available just for $\sigma \leq 0.4$. In the case of $\sigma \geq 0.6$, the result is a full block picture. In addition of it, for getting desired results when $\sigma \leq 0.2$, the differential calculations had to be carried and about 30 times of iterative operations are required. For our DTCNN, the times of iteration is less, for example, for the same 16x16 image, the number of iteration is only 3. Even though an input image is composed by 320x200 pixels superimposed $\sigma = 0.7$ Gaussian white noise, just 6 iteration operations are carried. Since no differential calculation is required in the iterations, the calculating speed is more faster than that of continuous-time CNN.

5.4 Visual pattern recognition

Pattern recognition, naturally, is based on patterns. It comes as little surprise that much of the information that surrounds us manifests itself in the form of patterns. A pattern can be as basic as a set of feature measurements or observations, perhaps represented in two-dimensional geometric description, vector or matrix notation[11]. In this study, we just consider the case of two-dimensional graphic patterns, and classify some specific patterns from a background. The similar pattern recognition techniques have been widely used for computer vision[9]. The ease with which humans classify and describe patterns often leads to the incorrect assumption that this capability is easy to automate. Sometimes the difference between some patterns are immediately apparent, whereas in other instances they are not. Some research in geometric description have revealed that, human vision system can almost instantaneously detect differences in a few local conspicuous features without the need of complex familiarity cues. These features are called as textons[10], which include elongated blobs as rectangles, ellipses or line segments with specific color, angular orientation, width, length, binocular, movement disparity and flicker rate. The terminators and crossings of line segments also are textons. For examples, Figure 5.12 and Figure 5.13 show two testing pictures. Each of them consists of 64 × 96 pixels.

The textures in Figure 5.12 are composed of " ┌" and "↗" shaped elements with rotated orientations. The differences between " ┌" shaped element and "↗" shaped element are very obvious so that it is rather easy to pick "↗" shaped pattern out from this background. But in Figure 5.13, it will take considerable time and effort to look out the "T" shaped patterns from the background, since the "T" shaped pattern and " ┌" shaped pattern are belong to one type of texton, they are similar with each other.

Because the geometric descriptions are such different, it is easy to find that the "↗" shaped element and "T" shaped element are belong to two distinct types of textons or patterns. It is nature to have to design two distinct templates to recognize them from the same background respectively. We give the templates as follows.

The template 1 is used to each cell in DTCNN with $r = 1$. The elements in the weight matrix A of (5.1) are decided by T_A, B by T_B and I by T_I. Inputting the probe image shown as Figure 5.12 to the DTCNN with the weight matrixes A B and I coming from the template 1, after 4 times iterations, the "↗" shaped elements are classified from the background shown as Figure 5.14.

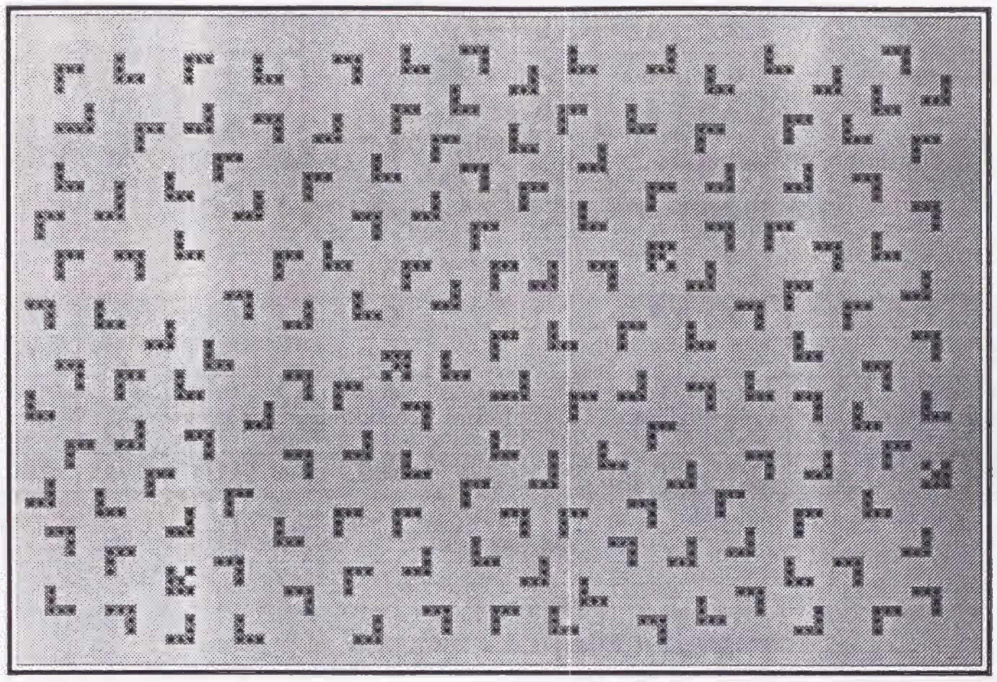


Figure 5.12: A probe pattern composed by " ┌" and "↗" shaped elements

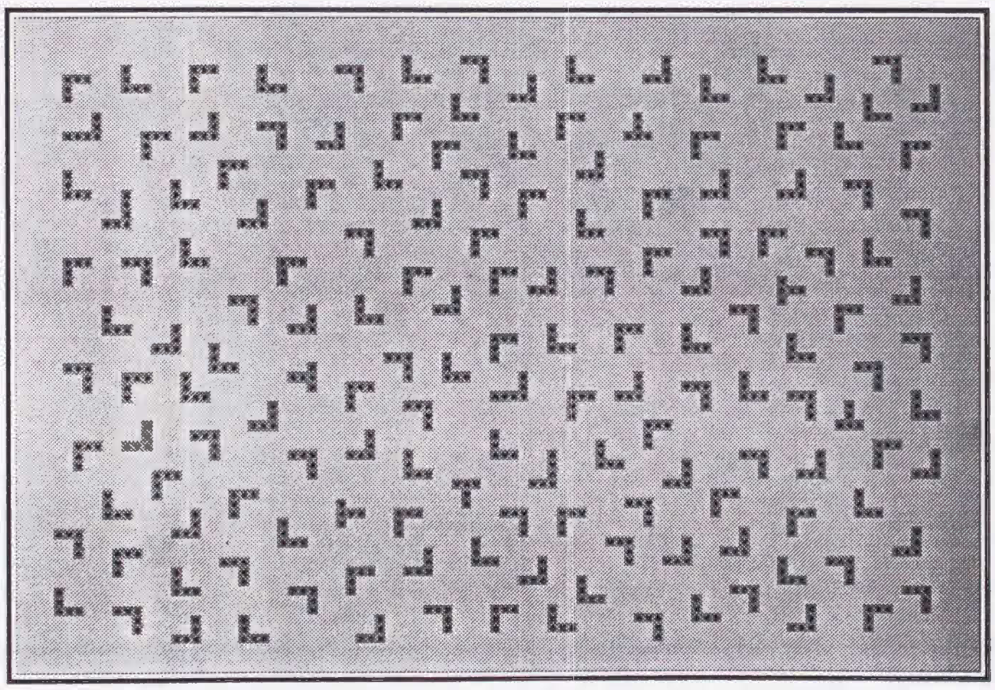


Figure 5.13: A probe pattern composed by " ┌" and "T" shaped elements

$T_A =$

0.0	0.5	0.0
0.5	2.0	0.5
0.0	0.5	0.0

$T_B =$

0.5	-0.5	0.5
-0.5	2.0	-0.5
0.5	-0.5	0.5

$T_I =$ -2.50

Table 5.3: Template 1 for pattern recognition

$T_A =$

0.00	0.00	0.00	0.00	0.00
0.00	0.25	1.50	0.25	0.00
0.00	1.50	8.00	1.50	0.00
0.00	0.25	1.50	0.25	0.00
0.00	0.00	0.00	0.00	0.00

$T_B =$

0.0	0.5	0.0	0.5	0.0
0.5	0.5	-1.0	0.5	0.5
0.0	-1.0	1.0	-1.0	0.0
0.5	0.5	-1.0	0.5	0.5
0.0	0.5	0.0	0.5	0.0

$T_I =$-2.50

Table 5.4: Template 2 for pattern recognition

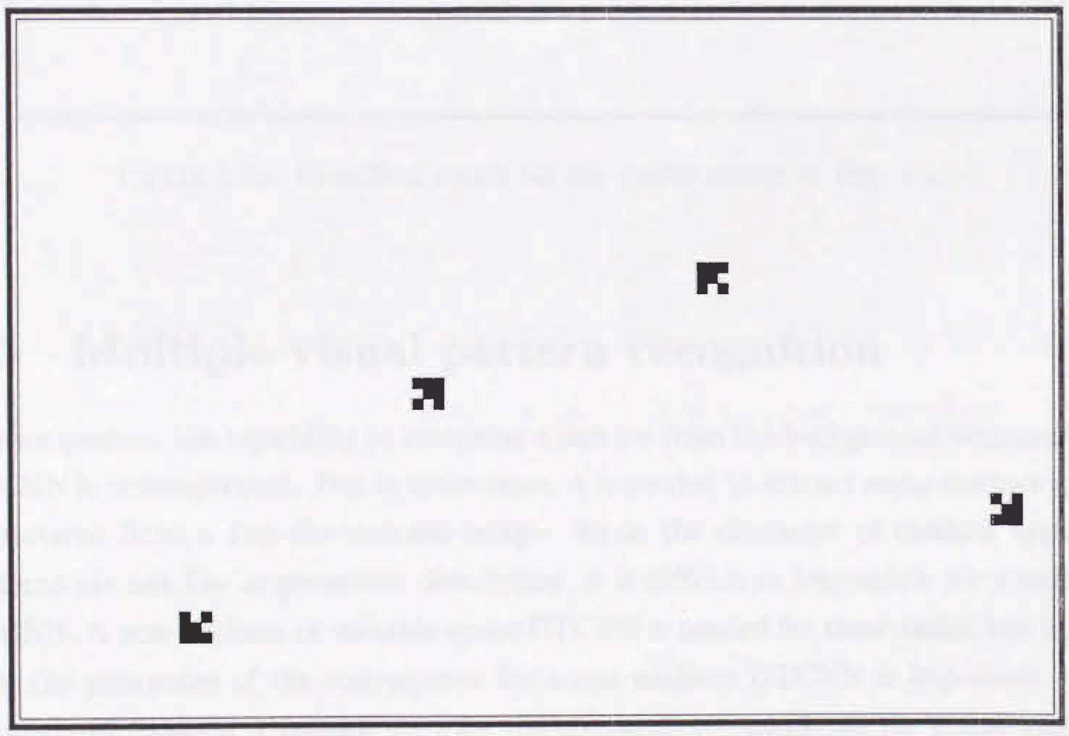


Figure 5.14: Classified result for the probe image in Fig. 5.12

Similarly, by the weight matrixes coming from the template 2 with the neighbor radius r=2, inputting the probe image shown as Figure 5.13, the "T" shaped pattern are picked up. Figure 5.15 shows the desired result.

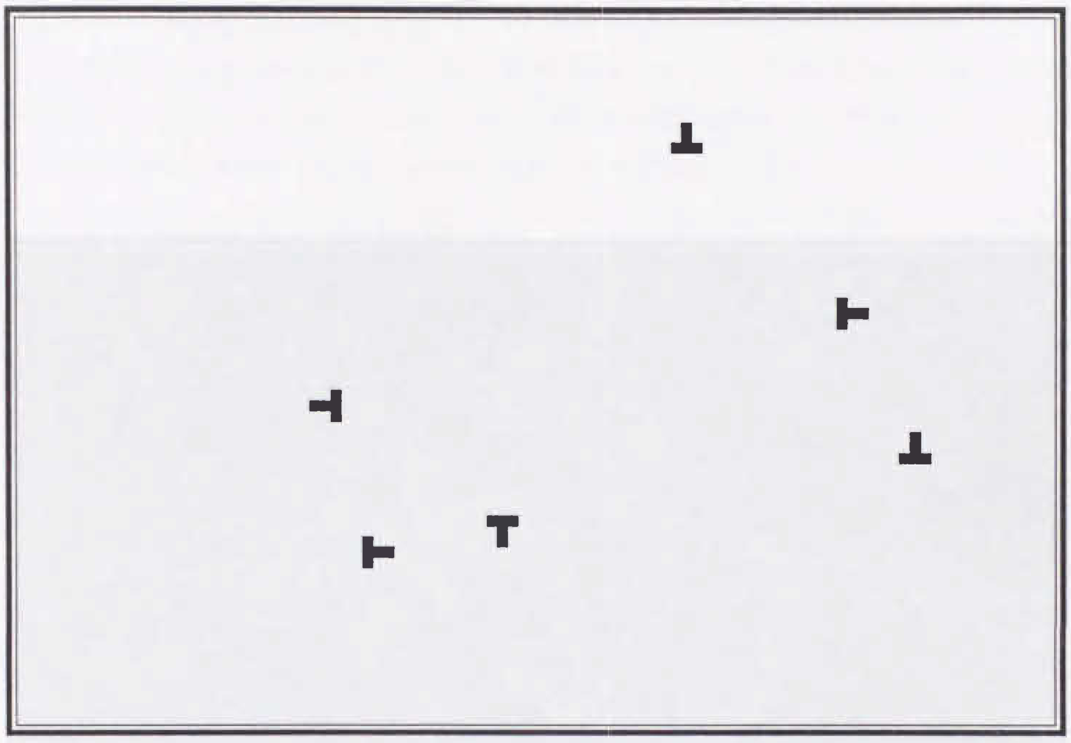


Figure 5.15: Classified result for the probe image in Fig. 5.13

5.5 Multiple visual pattern recognition

In prior section, the capability to recognize a pattern from the background with uniform DTCNN is demonstrated. But in some cases, it is needed to extract some distinct types of patterns from a two-dimensional image. Since the character of distinct types of patterns are not like in geometric description, it is difficult or impossible for a uniform DTCNN. A non-uniform or variable-space DTCNN is needed for these tasks, but before that, the guarantee of the convergence for a non-uniform DTCNN is important. The theorems in Chapter 2 provide us with the convergence conditions for a non-uniform DTCNN. Based on those analyses, we can design some non-uniform DTCNNs for multiple distinct patterns recognition.

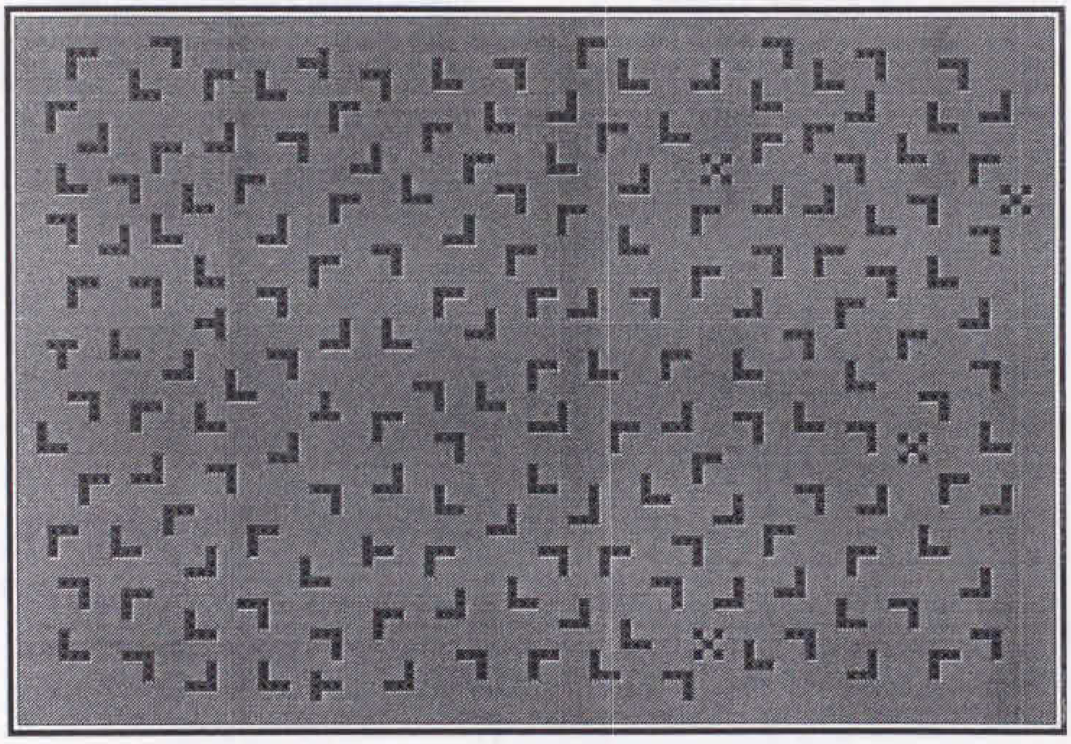


Figure 5.16: A probe pattern composed by three types of elements

Figure 5.16 is composed of the " ┌", "T" and "×" three types of elements. Assuming the "T" shaped elements only appear in the left half plane and the "×" shaped elements the right half plane. Then, for each cell lying in the left half plane, the template 3 is used. But for other cells, the template 4 are used. Each template makes a contribution to weight matrixes A, B and I by itself. In this way, we obtain a non-uniform DTCNN which template for every cell are variable with the space. This non-uniform property results in non reciprocal weight matrixes. After some times iterations, it is successful to recognize two distinct types patterns shown as Figure 5.17.

$T_A =$

0.50	0.00	0.50	0.00	0.50
0.00	1.00	0.00	1.00	0.00
0.50	0.00	8.00	0.00	0.50
0.00	1.00	0.00	1.00	0.00
0.50	0.00	0.50	0.00	0.50

$T_B =$

0.0	0.5	0.0	0.5	0.0
0.5	0.5	-1.0	0.5	0.5
0.0	-1.0	1.0	-1.0	0.0
0.5	0.5	-1.0	0.5	0.5
0.0	0.5	0.0	0.5	0.0

T_I = -2.50

Table 5.5: Template 3 for multiple visual pattern recognition

$T_A =$

0.00	0.00	0.00	0.00	0.00
0.00	0.25	1.50	0.25	0.00
0.00	1.50	8.00	1.50	0.00
0.00	0.25	1.50	0.25	0.00
0.00	0.00	0.00	0.00	0.00

$T_B =$

0.0	0.5	0.0	0.5	0.0
0.5	0.5	-1.0	0.5	0.5
0.0	-1.0	1.0	-1.0	0.0
0.5	0.5	-1.0	0.5	0.5
0.0	0.5	0.0	0.5	0.0

T_I = -2.50

Table 5.6: Template 4 for multiple visual pattern recognition

In previous example, we show that a non-uniform DTCNN has ability to recognize two types distinct visual patterns lying in the left half plane and the right half plane of a probe image at the same time. Next. we want to give another example to recognize two types distinct visual patterns lying in the upper part and the down part of a probe image at the same time. An image shown in Figure 5.17 is used as a probe image. It is similar with a reflection graph obtained by a laser radar, in which the upper part is

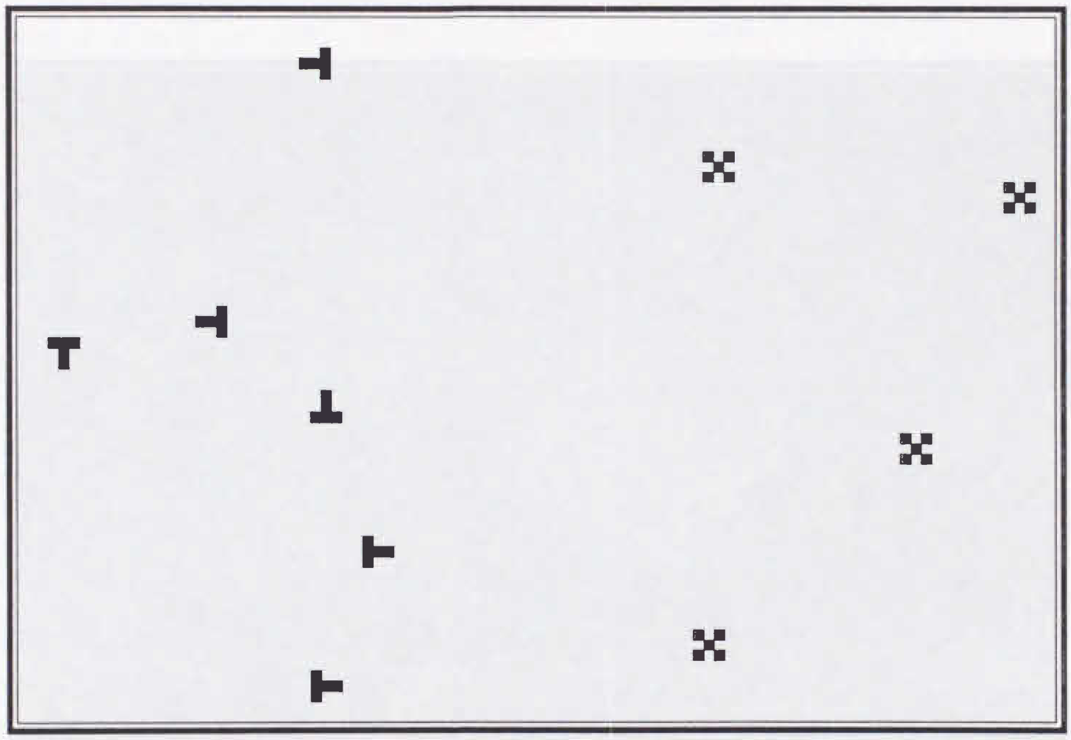


Figure 5.17: Two types of textons are picked out

supposed to be sky region so that the elements in this part is composed by the plane shaped reflect pixel and the noise reflect pixel, but the down part is sea region so the elements is composed by the shop shaped reflect pixel and the noise reflect pixel. What we want to do is to detect the plane shaped reflect pixel and the shop shaped reflect pixel at the same time. In order to do so, two templates listed in Table 5.7 and Table 5.8 are used for the cells lying the upper part and down part, respectively. Using the probe image shown in Figure 5.18 as both input signal and the initial state of DTCNN, after 11 times iterative operations, a result is derived successfully , shown as Figure 5.19.

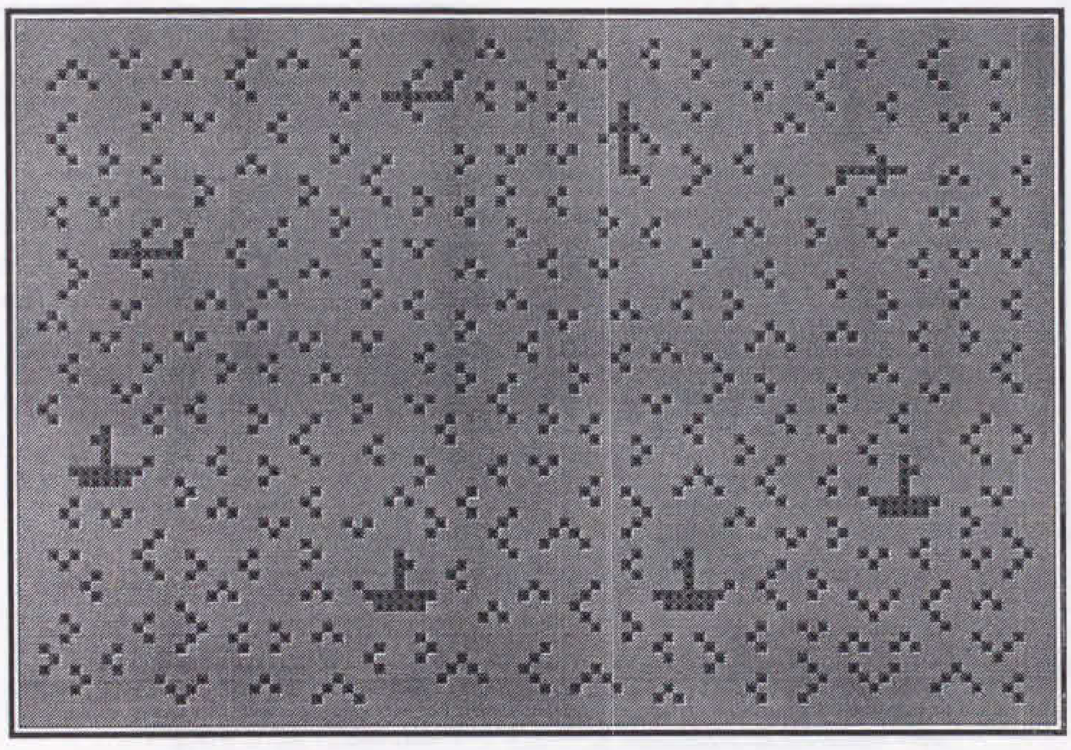


Figure 5.18: A probe image composed by three types elements

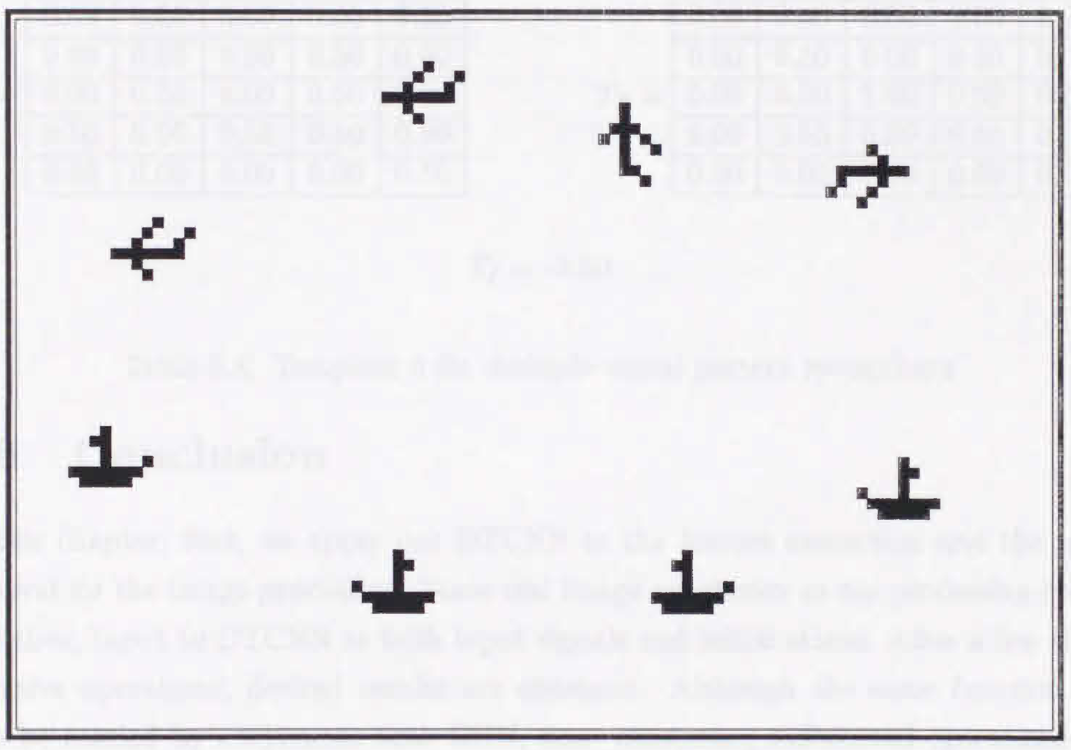


Figure 5.19: Two distinct types of elements are picked out

$T_A =$

0.00	0.00	0.50	0.00	0.00
0.00	0.50	0.50	0.50	0.00
0.50	0.50	8.00	0.50	0.50
0.00	0.50	0.50	0.50	0.00
0.00	0.00	0.50	0.00	0.00

$T_B =$

1.00	0.50	0.00	0.50	1.00
0.50	-0.50	0.00	-0.50	0.50
0.00	0.00	1.00	0.00	0.00
0.50	-0.50	0.00	-0.50	0.50
1.00	0.50	0.00	0.50	1.00

$$T_I = 0.50$$

Table 5.7: Template 5 for multiple visual pattern recognition

$T_A =$

0.50	0.00	0.00	0.00	0.50
0.00	0.50	0.50	0.50	0.00
0.00	0.50	8.00	0.50	0.00
0.50	0.50	0.50	0.50	0.50
0.50	0.00	0.00	0.00	0.50

$T_B =$

0.00	0.00	0.00	0.00	0.00
0.00	0.50	0.00	0.50	0.00
0.00	0.00	1.00	0.00	0.00
0.00	0.50	0.50	0.50	0.00
0.00	0.00	0.50	0.00	0.00

$$T_I = \text{-}2.50$$

Table 5.8: Template 6 for multiple visual pattern recognition

5.6 Conclusion

In this chapter, first, we apply our DTCNN to the feature extraction and the noise removal for the image processing. Some real image are chosen as our processing object and then, input to DTCNN as both input signals and initial states. After a few times iterative operations, desired results are obtained. Although the same function can also be carried by continuous-time CNN, time consuming differential operations are taken during the procedure and more iterative operations are required, Contrasting it, our DTCNN realized by software simulation can do them only with 5% or 10% computing cost, so it is faster and efficienter than continuous-time CNN in this case. After then, we illustrate the potential of DTCNN for the visual pattern recognition. From a prototype composed by two or more types of elements, we can detect desired visual patterns successfully. When there exist obvious differences between these two types of elements, it is easily recognized by human vision system. But for some similar composed elements, it is said to be very difficult and time consuming for human vision system. For our DTCNN, after suitable template is designed, it is easily and quickly to pick out our desired patterns from a prototype in both cases. This technique can be applied for robot vision. Finally, based on our convergent analysis result in Chapter 2, we design space-varying non-uniform DTCNN for multiple visual patterns recognition. In a non-uniform DTCNN, two or more templates are used for the cells lying in different region of 2-D processing array. Two examples are given to show the ability of non-uniform DTCNN to detect multiple visual patterns from a prototype at the same time, which have distinct geometrical character so they can not be picked out by unique template at once. It extented the application region of our DTCNN more over. Since the weight matrix A and B contributed by two or more distinct templates are not symmetrical matrixes, or, $A_{ij} \neq A_{ji}$ and $B_{ij} \neq B_{ji}$ generally, the stability analysis of unsymmetric continuous-time CNN is still open problem and dose not been solved, the similar application by continuous-time CNN has not been reported until now.

References

[1] L.O.Chua and L.Yang, "Cellular neural network: applications", *IEEE Trans. Circ. Syst.*, vol.CAS-35, pp.1273-1290, 1988.

[2] T.Roska and L.O.Chua, "Cellular neural networks with non-linear and delay-type template elements and non-uniform grids", *Int. J. Circ. Theor. Appl.*, vol.20, pp.469-481, 1992.

[3] H.Harrer and J.T.Nossek, "Discrete-time cellular neural networks", *Int. J. Circ. Theor. Appl.*, vol.20, pp.453-467, 1992.

[4] L.O.Chua and T.Roska, "Stability of a class of nonreciprocal cellular neural networks", *IEEE Trans. Circ. Syst.*, vol.CAS-37, pp.1520-1527, 1990.

[5] S.Hui and S.H.Żak, "Dynamical analysis of the brain-state-in-a-box(BSB) neural models", *IEEE Trans. on NN*, vol.3, pp.86-94, 1992.

[6] N.Fruehauf, L.O.Chua and E.Lueder, "Convergence of reciprocal time-discrete cellular neural networks with continuous nonlinearities", *Proc. 1992 IEEE Int. Workshop on CNNA*, pp.106-111, 1992.

[7] F.Zou and J.A.Nossek, "Stability of cellular neural networks with opposite-sign templates", *IEEE Trans. Circuits Syst.*, vol.38, pp.675-678, 1991.

[8] F.A.Savaci and J.Vandewalle, "On the stability of cellular neural networks", *IEEE Trans. Circuits Syst.*, vol.40, pp.213-215, 1993.

[9] T.Roska, J.Hámori at al., "The use of CNN models in the subcortical visual pathway", *IEEE Trans. Circuits Syst.*, vol.40, pp.182-195, 1993.

[10] B.Julesz and J.R.Bergen, "Textons, the fundamental elements in preattentive vision and perception of textures", *Bell Syst. Tech. J.*, vol.62, pp.1619-1645, 1983.

[11] R.Schalkoff, *Pattern recognition: statistical structural and neural approaches*, John Wiley & Sons, Inc., 1992.

Chapter 6
Overall Conclusion

As one point of our research, we presented a model of discrete-time cellular neural network in 2th chapter, and analyzed its stability property with uniform or nun-uniform, symmetric or unsymmetric weight coefficients matrix. First, we showed the cell model of the continuous-time CNN, and some typical types of 2-D array structures briefly. After introducing a two phases synchronous-updating signal into a continuous-time CNN, we obtained a synchronous-updating CNN, we called it as SCNN. By sampling the values of state variations v_i and output variations y_i at the updating moments $t = kT$, $k = 0, 1, 2.\cdots$, we derived a discrete-time CNN which topology and output function are distinct from the DTCNN presented by Harrer and Nossek. In general, the output of this DTCNN is a variable value during $(-1, +1)$, so that it can be used to image processing in which the output is a multiple grey level image. in order to guarantee the output as a binary value to meet some special applications, a sufficient condition and a necessary condition are presented here, which provide the design requirement for the matrix A and the matrix B. Moreover, in order to analyze convergence condition of this DTCNN, the generalized energy functions for our SCNN and DTCNN are defined respectively. Here, we don't directly compare the value of the energy function of DTCNN at two sequent of updating moments, which method is used by N.Fruehauf, L.O.Chua and E.Lueder for reciprocal DTCNN with the same output function. We analyze the energy function of SCNN during a clock period and around a updating moment, because the energy function is not continuous at those moments, which impact must be considered carefully. Two theorems about the convergence condition of nonreciprocal and nonuniform SCNN are described first. Meanwhile, since the energy function of DTCNN is sampled and discreted from that of SCNN, two convergence conditions are also available to nonreciprocal and nonuniform DTCNN.

The result covers the reciprocal DTCNN as a special case, and provide the potential to apply our DTCNN more widely, for examples, associative memories, multiple visual pattern recognitions and others.

Owing to the piecewise linear character of the non-linearities, cellular neural networks depend crucially their nonlinear dynamics. Proper operation often requires the existence of multiple equilibrium points or DC operating points. Therefore, it is important to have an efficient analysis method for obtaining a global picture of the dynamic behavior, the equilibrium pattern and the basins of attraction in a given network. It is a problem to find equilibrium points in CNN described by the state equation and the output equation.

In the chapter 3, we present a modified BDF curve tracing method for this problem. The result shows this algorithm could be used efficiently to trace those solution curve with some sharp turning points. Specially, we want to point out that the Brown method is a kind of the Gauss-Seidel algorithm to be used for nonlinear algebraic functions. It is known that the convergence ratio is second order near to the solution. Furthermore, a number of the function evaluations is $(N^2 + 3N)/2$ when the function consists of N functions. Observe that that the Newton method takes N^2 evaluations of the partial derivatives and N evaluations of functions. Thus, the Brown method is efficiently applied to trace solution curve, such that the approximate solution is obtained by Hermite polynomial.

The algorithm presented here can be useful in the analysis of neural networks, e.g. during the design of templates for cellular neural networks. It can be applied to large networks provided that the extreme sparity and the structure of the coefficients are exploited. The method can be applied for some types of neurons with smooth nonlinear output functions or piecewise linear output functions. In general, there does not seem to be much hope for an efficient way to find all equilibrium points in a given neural network unless appropriate guidelines are followed during the synthesis process.

The artificial realization for the associative memory is one of the important problems on the neural network applications. In several books and papers, the ability of neural networks to implement associative memories has been discussed. First, the information of several prototypes are stored into a neural network and then, a signal is inputted to the network where some information from a prototype is lossed because of the distortions and noises during the signal transmission and processing. Then, all or most of original information can be recovered with the associative memory. The researches on the associative memory can be directly applied to pattern recognitions and classifications.

In the chapter 4, first, we describe the outer product learning approach to set up the weights with suitable values which is related to the object patterns information, it is called as storing object patterns into a cellular associative memory. Meanwhile, some analyses about the stationary property of the cellular associative memory with outer product learning rule are taken. A condition is presented which ensure the stored patterns as the stable states of a cellular associative memory. After then, a middle-mapping learning algorithm for cellular associative memory is presented, which makes full use of the properties of the cellular neural network so that every stored prototype can be guaranteed as an equilibrium point of our memory. At the same time, it has ability of iterative learning. This kind of computation is typical of a learning process: once the synaptic matrix has been computed from a given set of prototype vectors, the addition of one extra item of knowledge does not require that the whole computation is performed again. One just has to carry out one iteration, starting from the previous matrix, so that the computational efficiency can be improved. Besides, its implementation with circuits is more feasible because the weight matrix is not symmetric.

Since the synchronous updating rule is used in both of them, their associative speeds are very fast compared to the Hopfield associative memory.

From the simulating results, we can find that, when the number of the stored prototypes is increased or the distortion in a probe is strong, the associative ability is decreased and the probability of converging to spurious states is increased. It is similar with the situation in the other types of associative memory networks. But in a cellular associative memory, it is believable that we can extend the size of a neighborhood of our cellular associative memory to improve its associative ability. Unfortunately, the realization of circuits is get more difficult at the same time and the manufacture cost is risen.

Differing with prior chapter, in the chapter 5, we apply our DTCNN to image processing with another view point. It is to consider DTCNN as a spatial operator. With appropriate choice of the connecting weights, the network can operate as a differentiator, an integrator or even more complexer operator, which include the cooperative operation, the competitive operation and the mixed operation. Although many similar tasks can be performed by current digital image processing techniques, DTCNN will operate faster than the former, generally, since it is a parallel operator. First, we apply our DTCNN to the feature extraction and noise removal for the image processing. Some real image are chosen as our processing object and input to DTCNN as both input signals and initial states. After a few times iterative operations, desired results are obtained. Although the same function can also be carried by continuous-time CNN, more iterative operations are required and, moreover, time consuming differential operations are taken during each iterative. Contrasting it, our DTCNN by software simulation is faster and efficienter than continuous-time CNN in this case. After then, we illustrate the potential of DTCNN for the visual pattern recognition. From a prototype composed by two or more types of elements, we can detect desired visual patterns successfully. When there exist obvious differences between these two types of elements, it is easily recognized by human vision system. But for some similar composed elements, it is said to be very difficult and time consuming for human vision system. For our DTCNN, after suitable template is designed, it is easily and quickly to pick out our desired patterns from a prototype in both cases. This technique can be applied for robot vision. Finally, based on our convergent analysis result in Chapter 2, we design space-varying non-uniform DTCNN for multiple visual patterns recognition. In a non-uniform DTCNN, two or more templates are used for the cells lying in different region of 2-D processing array. Two examples are given to show the ability of non-uniform DTCNN to detect multiple visual patterns from a prototype at the same time, which have distinct geometrical character so they can not be picked out by unique template at once. It extented the application region of our DTCNN more over. Since the weight matrix A and B contributed by two or more distinct templates are not symmetrical matrixes, or, $A_{ij} \neq A_{ji}$ and $B_{ij} \neq B_{ji}$ generally, the stability analysis of unsymmetric continuous-time CNN is still open problem and dose not been solved, the similar application by continuous-time CNN has not been reported until now.

Appendix A

A List of the Related Papers by the Author

A.1 Publications

1. Chen He and Akio Ushida, "Iterative middle mapping learning algorithm for cellular neural network", *Institute of Electronics, Information and Communication Engineers Trans.*, vol.E-77A, no.4, pp.706-715, 1994.

2. Chen He and Akio Ushida, "A modified predictor-corrector tracing curve algorithm for nonlinear resistive circuits", *Institute of Electronics, Information and Communication Engineers Trans.*, vol.E-74, no.6, pp.1455-1462, 1991.

A.2 International Conferences

1. Chen He and Akio Ushida, "Convergence analysis of synchronous-updating CNN and related DTCNN", *Proc. of 1993 Int. Symp. on Nonlinear Theor. and its Appl.*, pp.29-34, Hawaii, American, 1993.

2. Chen He and Akio Ushida, "Cellular associative memory with middle mapping learning algorithm", *Proc. of 1993 Int. Symp. on Neural Network and Signal Processing*, pp.160-165, Guangzhou, China, 1993.

3. Chen He and Akio Ushida, "An efficient algorithm for solving nonlinear resistive circuits", *Proc. of 1991 IEEE International Symposium on Circuit and System*, pp.2328-2331, Singapore, 1991.

A.3 Technical Reports and Other Presentations

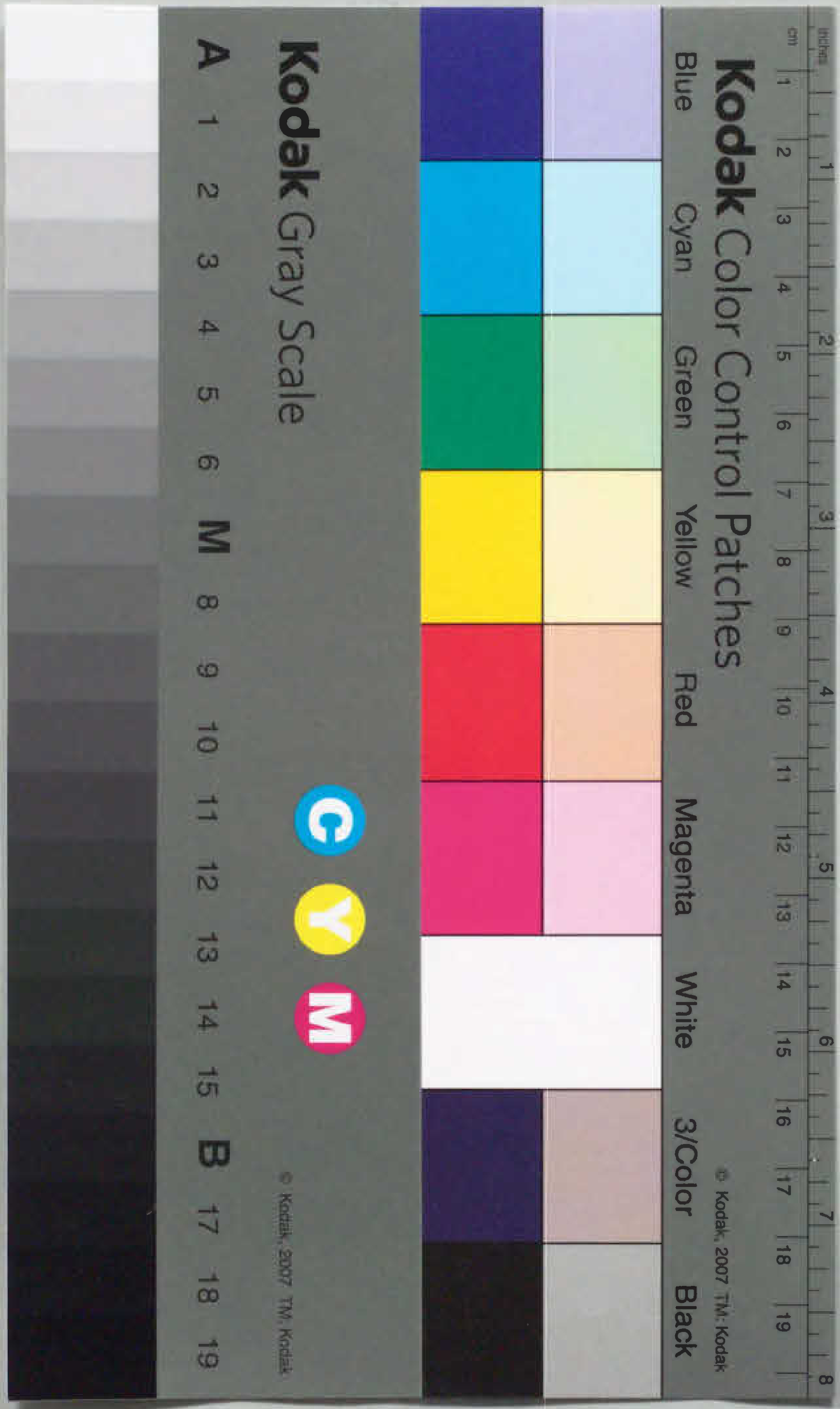
1. Chen He and Akio Ushida, "Non-uniform discrete-time cellular neural network and multi-patterns recognition", *Technical Report of the Institute of Electronics, Information and Communication Engineers*, vol.NLP94-11, pp,1-8, 1994.

2. Chen He and Akio Ushida, "Cellular associative memory with the pseudo projection learning algorithm", *Technical Report of the Institute of Electronics, Information and Communication Engineers*, vol.NLP92-97, pp.41-46, 1993.

3. Chen He and Akio Ushida, "Cellular associative memory with middle mapping learning algorithm", *Proc. of IEICE 1993 National Conference*, p. SA-1-1, Nagoya, 1993.

4. Chen He and Akio Ushida, "The stationary analysis of the associative memory with the cellular neural network", *Proc. of 1992 IEICE Symp. on Nonlinear Theor. and its Appl.*, pp.131-135, Hakone, 1992.

5. Chen He and Akio Ushida, "Analysis of the model of Hopfield circuit with homotopy method", *Proc. of IEICE 1991 National Conference*, p.(1)84, Tokushima, 1991.



Kodak Color Control Patches
Blue
Cyan
Green
Yellow
Red
Magenta
White
3/Color
Black
© Kodak, 2007 TM: Kodak
Kodak Gray Scale
A 1 2 3 4 5 6 M 8 9 10 11 12 13 14 15 B 17 18 19
C Y M
© Kodak, 2007 TM: Kodak