國 立 交 通 大 學

電機與控制工程學系

博 士 論 文

分子類神經網路於數位影像處理的應用

Applications of Cellular Neural Networks in digital image processing

研 究 生：壽 宇 文

指導教授：林 進 燈

中 華 民 國 九 十 五 年 六 月

分子類神經網路於影像處理的應用

Applications of Cellular Neural Networks in digital image processing

研 究 生：壽宇文　　Student：Yu-Wen Shou

指導教授：林進燈 博士　　Advisor：Dr. Chin-Teng Lin

國 立 交 通 大 學

電 機 與 控 制 工 程 學 系

博 士 論 文

A Dissertation
Submitted to Department of Electrical and Control Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in
Electrical and Control Engineering
June 2006
Hsinchu, Taiwan, Republic of China

中華民國九十五年六月

分子類神經網路於數位影像處理的應用

研究生： 壽宇文　　指導教授： 林進燈 博士

國立交通大學電機與控制工程學系(研究所)博士班

摘　要

在這篇論文裡，我們將分子類神經網路(Cellular Neural Networks)應用於複雜且具有代表性地數位影像處理；分子類神經網路一直在學術界有著其特殊且不可取代的地位，其原因主要在於其具備了完整的理論基礎以及在實用時穩定性(stability)和堅固性(robustness)的易於操控，當然最吸引人地莫過於分子類神經網路可硬體實現化的優勢，不過由於硬體實現可能性的考量，分子類神經網路中樣版(template)的設計往往是愈單純愈容易達到硬體實現的目的，但此一設計限制卻和一般數位影像處理演算法的需求大異其逕，也因此使得在過去的文獻裡分子類神經網路只侷限於應用在一些簡單的影像處理技術，為了突破此一瓶頸，我們所提出的這篇論文不但清楚詳細地討論分子類神經網路於高階影像處裡的可能應用演算法更提出實際案例來證明分子類神經網路應用的可能性，所以我們所提出的方法不僅可以解決過去一些高階影像處理的問題，同時也為未來種種數位影像處理於硬體實現的可能提供了一個完整及實際的實現策略。

這篇論文主要可以分為三大部分：在第一部份裡，我們會詳細地說明並討論在過去到現在大部分將分子類神經網路應用於影像處理的相關文獻及未來所有可能的發展和技術，另外也將分子類神經網路作一完整的介紹，除此之外，我們也會特別著重於分子類神經網路在影像處理相關應用理論的討論以及其硬體實現化的考量；在第二部分裡，我們提出了一個將分子類神經網路應用於影像辨識處理的基礎分析—紋路分析(Texture Analysis)，這是由於紋路分析的複雜性和普遍性會使得分子類神經網路於高階影像處理的應用不會只侷限在單一的影像處理技術，其中我們也提出了一個相當有用的空間特徵(spatial feature)，此一特徵不但可以使複雜地高階影像處理能夠應用分子類神經網路，也為影像辨識技術提供了一個很好的辨識機制；在最後一部分裡，我們也將文件影像分析做了一個完整的剖析，並以文件影像的去網點為例來說明在實際情況下的分子類神經網路的應用，如此演算法的開發也為文件影像處理提供了更多實際的應用，更考量了文件影像處理若以軟體實現時的計算量負荷，而對未來高階數位影像處理能夠以硬體實現來提高處理速度提供了無限的可能。

Applications of Cellular Neural Networks in digital image processing

Student: Yu-Wen Shou **Advisor: Dr. Chin-Teng Lin**

Department of Electrical and Control Engineering
National Chiao-Tung University

Abstract

This dissertation tackles the all-time challenging research field of digital image processing by using Cellular Neural Network as means of its application. As we all know, Cellular Neural Network has been critically acclaimed by the academia for its impeccable theoretical structure and the stability and robustness its applications speak for. Aside from these advantages it presents, Cellular Neural Network appears to be compelling in that it can be practically realized for hardware compilation. However, this does not mean Cellular Neural Network is without limitations. When it comes to hardware compilation of Cellular Neural Network, decent and satisfactory results only come with easy and simple template design, which on the contrary contradicts the algorithmic expectations we have for image processing. This explains why, for years, among all those respectable academic papers and researches, Cellular Neural Network has been applied only for simple, low-level image processing technology. In this dissertation, I will not only dig into the possible algorithms of applications of Cellular Neural Network in higher-level image processing, but use practical case study to justify how these applications may turn out with unexpectedly outstanding performance. In such doing, this dissertation serves as a step stone for papers of its counterparts to come, and, more importantly, it proposes a strategic alternative to the

realization of models for image processing.

This dissertation consists of three major parts. In the first part, detailed discussions and delicate analyses of academic papers on Cellular Neural Network will be provided in the hope of helping us see the potentiality of Cellular Neural Network in the applications of image processing. I will focus on the aforementioned limitations on hardware compilation as well. In the second part, I will put forth “texture analysis” as one basic model of analysis when we apply Cellular Neural Network to image processing. In this so-called texture analysis, a useful “spatial feature” is especially drawn to help us overcome possible problems of more complicated Cellular Neural Network applications in image processing. “Spatial feature” also serves as a well-functioning mechanism for technology of image identification. In the last part of this thesis, I will look into a case study, where Cellular Neural Network is applied to help de-screen document image. Using it as an example, we will see how algorithms of Cellular Neural Network may be of marvelous use in applications in document image processing, since it would reduce a great deal of calculation and computation when applied to software compilation, yet opens up unlimited possibilities for higher-speed hardware compilation of high-level image processing.

Acknowledgment

這篇論文的完成除了要特別謝謝我的父母及家人在我的博士求學期間所給予我的所有經濟上以及精神上的幫助，當然我也要在這裡特別謝謝我的指導教授林進燈博士所給予我的所有指導，謝謝他在我的博士求學期間提供了一個獨立及自由的研究空間，由於他的啟發與指導激發了我在研究討論的空間裡更多的創新想法及不同於傳統的思考模式，沒有他們不會有這篇論文的完成，另外在這裡我也要特別謝謝我的好友陳德良和沈聿德，他們在我的求學期間也給予了我很多精神上及實質上的幫忙與支持，最後我也要對所有曾經幫忙過我的朋友、同學及實驗室裡的學弟妹們至上我最深的謝意，由於他們的無私幫忙使得我可以在沒有顧慮的情況下做研究，因而促進了此篇論文的完成。

Contents

List of Figures

List of Tables

1. Introduction

In this chapter, we will introduce the related surveys in the applications based on Cellular Neural Networks (CNN's), the applied image processing techniques in particular. We mainly propose the methodology based on CNN's for some image processing techniques in this thesis. In this section, we shall begin with what motivated this thesis, and then introduce state-of-the-art in the related areas of the main content of this thesis. In addition, we explicitly define our contributions of the proposed methodology at the end of this section, which might enhance the readability of this thesis.

1.1 Motivation

In the past studies, CNN's have been well-known for its hardware implementability and the well-connected interactions between the corresponding cells. The basic theory and foundation of CNN's is not the only issue for most researchers, but how to apply CNN to any industrial or academic areas is yet more interesting and significant. Therefore, the characteristics of CNN's in hardware implementations and digital image processing motivate us to produce the works related to the discussions in high-order image processing techniques. As we know, the applications of CNN's are always restricted since we have to take into consideration the hardware implementations and chip design in the development of our algorithms for more complicated applications. In this way, the design of CNN structure has to be as simple as possible. That is the reason that the current status of CNN's focuses on the chip design for simpler image files like binary ones. It is obvious that most problems in image processing are difficult and unpredictable. In order to avoid using CNN's too theoretically, we have been dedicated to searching for more possibilities in many image processing applications by CNN's. Besides, we need to find more solutions for

enhancing the image processing performance and speeding up the algorithms in more complicated images and issues, image documentation in particular. We thus try to carry out all the above requirements and inspire more ideas for readers by this thesis.

1.2 Related Works

1.2.1 Cellular Neural Networks

We shall introduce the interesting topics and various applied fields of CNN's in this section. In the beginning of developing CNN structure, the studies of the basic theory and the distinguished characteristics in CNN's are the most important issues in its related research areas. It is surely that most works related to CNN's cannot simulate its reactions without regarding the stability and robustness of CNN's [1] ~ [2]. However, the equivalent accounts will be beyond the scope of this thesis. We do not need to highly stress on this issue simply because the problems of hardware implementations can be overcome if the CNN template can be optimized in the restricted range. In addition, the discussions of the steady and transient responses attract the attention of most researchers. When it comes to the applications of CNN's, the transient response especially matters since it can reflect the connected relationship of corresponding cells of the specific location in the input signal. As for the related applications of CNN, how to determine a better CNN template is quite a critical topic. Apparently the intuitive design and the learning process both lead to the technical approach for deciding a specific CNN template. We have broadly two kinds of approaches in search of CNN templates in the past literature, including analytical and learning methods. The analytical approaches should be composed of a set of local rules which characterize the dynamics of a cell, depending on its neighboring cells. And the operating templates can be obtained correctly by an affine set of inequalities from the transformed local rules [3] ~ [4]. According to different ways of training, the

learning approaches can be classified as local and global learning algorithms. The local learning algorithms [5] ~ [6] take advantage of the neural-based training process like neural networks in back propagation while the global ones [7] ~ [8] optimize CNN templates in a stochastic form like genetic algorithms or simulated annealing approaches.

The difficulty of CNN template design lies in the fulfillment of a given task with implementable templates. As what we mentioned above, the analytical approaches indeed provide a systematical way to look for a better template more simply and directly [9] ~ [10]. Nevertheless, a simpler approach must not give a satisfactory performance or results for complex tasks. There also exist some useful learning approaches such as genetic algorithms [11] or combined strategy for finding robust and stable CNN templates. For the hybrid methods, they combined stochastic optimization schemes with hill climbing algorithms [12]. The combined strategy can solve some specified problems as long as its design fits the requirements of a given issue in advance, but the design would be sometimes more complicated and result in the betray of CNN kernel concept. Thus and so, we need to find not only a systematical and standard approach but also a flexible algorithm for determining CNN templates in an easier fashion. That is the reason that we would adopt genetic algorithms to optimize a better template for any given task. No doubt, this kind of learning methods like genetic algorithms have been studied in the past researches, which makes the related material more sufficient and the researching timing more mutual.

CNN templates can be regarded as coupled or uncoupled by the center element of the control template. In fact, the uncoupled template has ensured the center cell to be free from the surrounding influence. Hence most techniques based on CNN's for digital image processing would be defined in the CNN template library by this right

kind of template form. Take the applications in shadow projection, global connectivity detection, and connected component scanning for instance, the form of the relative CNN template would be distributed more regularly [13]. As the counterpart, the distribution of template would be irregular if the defined local rules cannot describe the dynamic behavior of input signals. Since the form of template cannot be indicated by its regularity, we would rather simulate our required CNN template by the learning algorithms in this thesis.

1.2.2 Texture Analysis

In this section, we will survey the related material in texture analysis. For texture characterization and classification, statistical approaches have the major impact on the advanced studies of texture analysis. Texture classification has long been one of the most difficult problems to tackle in image processing in terms of the characteristics of various texture patterns, say, uniformity, regularity, coarseness, just to name a few of it. Therefore, many different approaches have been put forth to solve the problems of texture classifications. To carry out a better performance and more comprehensive results in classifying different textures, a great amount of solutions to the analyses of texture patterns are proposed to in the hope of revealing some inherent natures of these patterns and being proven useful for further applications to the higher level processing. There totally exist eight kinds of analytical approaches, inclusive of optical and digital transforms, high-frequency-component analysis, autocorrelation functions, structural elements, co-occurrence probabilities and run length measurement by spatial gray toning, and autoregressive models [14]. Among these approaches, some of them directly or indirectly estimate the spatial frequency of image textures and some of them make use of the structural methods. This is because texture patterns in the different uniformity could be sensitive to different ranges of spatial frequency band. The structural approaches introduced in [15] ~ [16] generally

take the matching procedure to detect the spatial regularity of shapes called structural elements in a binary image, and the descriptions of auto–correlated textures could be carried out by more complicated texture patterns. The co-occurrence probabilities of textures can be described by spatial gray toning, and this approach utilizes various changes of distances between different textures. Also, the approach by measuring run lengths in different gray scales can characterize coarser or finer textures as more or less dominant pixels in a fixed gray toning run length. As to the autoregressive model, it estimates the gray toning proportion of any given pixel in the predefined neighborhood for the representative texture patterns. It can be easily observed that the distribution of coefficients has a very large variation for different image textures. These approaches naturally have different pros and cons and can be made up for each other. For the details of related comparison, [17] concludes that spatial frequency approaches generally have worse performance than the other approaches, and the structural approaches can only apply to the binary images. Besides, the co-occurrence approach can describe the spatial within-relationship of textural patterns by gray toning and can be less influential by monotonic gray toning transformations. The defection of this approach is the inability of describing the shape aspects for tonal primitives, which makes it hard to work well in texture patterns composed of large-area primitives. Finally, the auto-regression model by a given linear estimator can synthesize texture patterns more correctly, so it is more representative for macro-textures. On the other hand, this approach will complicate the analyzed texture patterns by micro-textured segments.

As the popular models and approaches which we have stressed on, it is not an easy task for researchers to find an absolute solution for texture classification and discrimination. And it is uneasy to develop a set of useful features for texture representation. This thesis hence strives itself for finding an analyzing feature set for

texture analysis in addition to working on the practical applications based on CNN's. As the matter of fact, texture analysis can be seen as the fundamental research to any other image processing techniques. That is to say not only the structure presented in texture analysis can motivate the methodology of other issuing problems but also the developed features for texture representation may be applied to the proper processing approaches of related academic fields. We here try our best to look for more applications based on CNN's for future possibilities in hardware, and in pursuit of solving these significant problems more useful tools and corresponding features have to be developed to fit the requirements of them.

1.2.3 Genetic Algorithms

In the past, genetic algorithms (GAs) could be characterized by the practical applications, robust optimization and search methods. Many researchers applied this learning algorithm to various areas such as genetic synthesis, chip design, strategy planning, machine learning, image and speech processing. The search methods by GA come from the mechanism of evolution and the combination of natural genetics [18]. The evolution of GA began from the heuristic search approach to simulated annealing algorithm. Simulated annealing algorithm takes thermodynamics into consideration and annealing here can be used as the optimization process for mathematical simulation [19] ~ [20]. Actually, simulated annealing and genetic algorithms are similar in the sense that they use the probabilities for searching the maximum or minimum of functions. However, genetic algorithms would be much different and superior for generating a sequence of populations by using selection, crossover and mutation. For extending the nature of GA, only the individuals with proper chromosomes can be well adapted to competition and survive, so adapting to a changing environment is essential for the survival of each species. The characteristics of genes can be dominated by the respective genetic content if the survival capability

is characterized by the features of corresponding individuals. In some defined case, the observed features can be controlled by a bottom unit like the role of a gene, and the set of genes dominates the features observed from the chromosomes. The evolution process though manifests itself as a succession of changes in sensing the variation of features, and its driving force causes the joint reaction of genetic reproduction mainly coming from the recombination of each other and the appropriate selection. As to the principle regulations of GA, only the fittest individuals can survive and reproduce, and this kind of natural phenomenon can be described as “the survival of the fittest” which truly reflects how GA can be related to the way of genetic combination. As a result, the genes of the fittest survive while the weaker ones fade away. Natural selection implies the survival of the fittest genes, and simultaneously the reproduction process generates diverse choices in the gene pool. The first step of evolution usually combines the chromosomes from the parent generation for reproduction, and then a new combination of genes as well as a new gene pool will be generated. For the combined genes with a worse representative power, the exchange of genetic chromosomes called crossover provides a better combination of genes. Finally, the iterative selection and crossover keep on the evolution process of this gene pool and the generation of competitive individuals will make the learning process terminate properly. Hence, GA can be applied to some problems like optimization or local max/min searching since solving these analytical problems is equivalent to finding the best numerical solution by GA. The reference [21] illustrates the standard form of GA which represents a binary alphabet as the strings of bits in the encoding process and provides the necessary driving power for better solutions to survive in the selection process. The evolved genetic chromosomes should be associated with the higher fitness value, and then could be compared with other reproduced generation. The higher the fitness values of the produced population,

the higher chances of survival for the competitive genes. Also, the crossover operation exchanges the proportion of transferred bits between strings to enhance the performance of reproduction. The mutation operation in GA causes sporadic and random alteration of the bits of strings, and it also implies a direct connection with outer world. Mutation hence at times helps to look for the better combination of chromosomes especially in the difficulty of getting the optimized solution.

We have introduced the related studies and the fundamental concept of GA in this section. As what we described above, GA is not only an adaptive learning algorithm but also a deterministic approach for global optimization. In this thesis, we use GA as the training tool for looking for the related parameters in CNN settings. GA here plays an important role in the design of templates when no *a-priori* information can be given in advance. The nature of GA makes the processed results from the optimized template approach the desirable ones of our applications. Therefore, GA indeed provides a better solution for us to optimize an arbitrary CNN template in the specific field of applications.

1.2.4 Image Documentation

Image documentation can be regarded as one significant step of image preprocessing before dealing with any complicated applications. As we know, there are many key topics in this field such as noisy background removal, image descreening, image deskewing, auto-cropping, and so forth. Each of them has some crucial influence on some issues in the related research fields. We take image descreening for example to illustrate how we can handle this difficult problem by hardware orientated CNN's. More complex issues in image documentation can also be solved by the generalized approach. In fact, image descreening takes an important part in the analysis of document images. It could make much easier the following

processes of document images such as texture or graphic separation. The existence of screening signals will make it harder to handle and analyze the document images, and the specific kind of problems in dealing with document images are the general ones for photo preprocessing or image enhancement.

There exist many methods for image documentation such as wavelet transforms, thresholding techniques, and the statistical analysis methods, etc. Wavelet transforms use the decomposition analysis of different levels to remove these undesirable screening signals. Its key point lies on the choice of basis functions. From the papers [22] ~ [24], we could find the applications restricted by using wavelet transforms with common basis functions such like Haar function and Daubetch spline function. In addition, the quality of documental images may suffer degradation during transformation from a scanned halftoned image to electronic formats through the introduction of artifacts such as morie patterns [25] ~ [27]. Numerous inverse halftoning or descreening methods have been used to eliminate these artifacts regardless of the causes for generation of screening noisy patterns [27] ~ [29]. In most studies, (inverse) halftoning techniques can be generally classified into two categories: frequency [30] and spatial [31] domain approaches. As a matter of fact, frequency domain approaches could keep more textured information in the screened images; whereas spatial domain approaches retain more properties for the spread of locations of screenings. That is also the reason that most papers tend to use such frequency domain methods like FFT, wavelet, or even Gabor filtering methods. Unfortunately, their descreening results are still restricted even if complex filters are used through time consuming procedures. This mainly results from that no fixed filter could be successfully employed in every kind of screened images. Thus, in chapter 3, we introduce a unique mechanism including two parts: the classification of screening

textures and the descreening procedure by the selected adaptive lowpass filter based on the classified screening textures. We hence propose a new strategy for image descreening in image documentation based on an innovative analyzing model in this thesis. It consists of the design of CNN structure and studies of characteristics in document images. Due to the different intensity of screening signals in different images, the screenings can be detected and processed by the proper solutions. And also, our proposed scheme for an optimal selection of filters can remove the screening signals more efficiently. We demonstrate our experimental results with a large number of various document images to show the robustness and stability of the proposed method. In the chapter 3, we will give the details of our proposed methods in the design of CNN structure for the representative topic in image documentation – image descreening.

1.2.5 Genetic Algorithm based Cellular Neural Network

In this section, we would like to survey the related applications of CNN by GA. Especially, we would focus on the topics introduced in this thesis, texture analysis and image descreening. Like neural networks [32] ~ [35], CNN is a large-scale nonlinear analog circuit, which processes signals in real time. CNN is made of a massive aggregate of regularly spaced circuit clones, called cells, which communicate each other directly only through its nearest neighbors [13]. To avoid the results from being confined and thus leading to an unsatisfactory outcome as shown in [36], [37], GA is therefore preferred to solve the problems of stability and adaptation in CNN's for two reasons. First, GA here carries with itself a dual function in deciding template elements for CNN: it serves to minimize the objective function, the optimization while avoiding the occurrence of oscillation and chaos when testing the pros and cons of working templates for CNN, that is, adaptation. Second, the design of template elements for CNN based on GA is no longer subject to the types of objective

functions and minima, i.e., differentiability of the cost function and the existence of local, global, separate, multiple minima, etc. Another advantage is the fact that GA not only applies itself to single layer CNN, but also can be used for the parametric design of multilayer CNN. Like what is mentioned in [38], the template design of multilayer CNN is necessary at times for complex problems that cannot be solved or realized in the easier manner of single layer CNN particularly. Selecting CNN templates by GA, therefore, has been widely adopted in every field of applications, regardless of single- or multilayer CNN, and shown to be powerful and robust in theory and practice [38] ~ [41]. Stochastic learning approaches, GA in particular, have become a crucial alternative to deterministic ones, which replace the classical methods by using the independent properties of initial conditions and the domain of applications combined with the implicit parallelism [39]. And the detailed descriptions about GA for choosing template elements of multilayer CNN have been given in reference [42] where only the global responses to the input images of the system would be available. The multilayer CNN design is certainly employed in the near future if the texture patterns were much more complicated than those we expected or no *a prior* information about the structure of the system had been given, or the separate operation of every layer had been in need. In [43], three different CNN templates trained by GA were proposed and carried out to give us a comparative index for performance of the system in different combinations of evolutional ways among generations, i.e. the average, inverse, and time-interpolated templates. Also, a modified assumption of parameters in GA like crossover or mutation rate in [43] made it practical to push the responses out of the way giving rise to the difficulty of convergence. Beside of the changes of fitness functions in GA, GA could be amended in other evaluation forms like the penalty functions mentioned in [39], [42] to give a punishment assessment between layers of CNN if the structure of multilayer CNN is

required. Now that the precise adjustment of parameters in our CNN template design based on GA is not necessary for our screening pattern classification in the descreening process, some slight changes about GA like the adaptation of fitness functions, how to set the parameters in the evolutional flows, etc. would be very useful and applicable to the issue of image descreening addressed in the chapter 3. Besides, the advanced studies on GA based CNN will be given in chapter 4 in more details.

1.3 Contributions

We will organize this thesis by giving our contributed parts and the whole constructed structure of our strategic approach in this section. This thesis contributes to applying CNN's in digital image processing. CNN's have been only used in the simple and direct operations of image processing techniques. In this thesis, we try to see CNN's in various realized simulations for image analysis. More clearly, this thesis deals with image processing problems by a separate mechanism which makes use of CNN architecture and original information extracted from image data. At this place, CNN plays an important role in image analysis – feature extraction. CNN is not only distinguished in its hardware implementability but also reveals some information of digital images in some sense by placing different CNN templates. To be more systematically, we shall firstly introduce our structure of CNN and our significant contributions in two aspects.

A. Image analysis

CNN has been used in many areas including digital image processing. Since using CNN is subject to the relationship between the original image files and CNN parameters, the image processing techniques by CNN should be simple and the convolved results from CNN should be reflected directly. This thesis pursues using

CNN to analyze the information of image data. Later in the chapter 3 of this thesis, we demonstrated how to use CNN in some high-level image processing like image documentation. CNN here is a classifier for specific predefined patterns to determine the following strategy after classification where we have referred the CNN structure as one part of the whole software simulations and algorithm development. Simultaneously, CNN is used to decide the related arguments that could reflect the linking relationship between image data and CNN parameter setting. This thesis thus introduces a complete strategic approach to understand the image natures for image analysis. We then explicitly list the main contributions in this identified category which have been made by our proposed structure as follows.

A.1 Image preprocessing

CNN used to handle some simple image applications, so many studies only focused on the preprocessing steps of digital image processing. In this thesis, we have clearly defined the range of space that CNN could be applied to and the corresponding adjustments and critical points in CNN parameter setting. That is, our applications in image processing by CNN would focus on algorithm development rather than hardware implementation only because the lack of CNN applications makes it hard to use the advantages of CNN structure well.

A.2 Image documentation

We applied CNN to image documentation since this field is a very important one in various researching fields of image processing. Also many problems in image documentation have been challenging and difficult even in software engineering and simulations. Therefore, we prove that CNN could not only be valid for simple operations of image analysis but also useful for more complicated applications. This thesis makes use of CNN structure to discuss the natures of image data for understanding the characteristics of documental images. In this way, CNN is like an

analytical tool for illustrating image data rather than a specific tool for some specific application. This thesis successfully uses CNN in image processing in a different point of view. For the other significant contribution proposed by this thesis, we have the following category.

B. Feature extraction

Unlike the traditional approaches using CNN's in the literature, this thesis does not focus on finding a specific CNN template for some specific image applications. CNN is taken as an analytical tool for digital image processing, in which CNN could be regarded as a transformation mapping like FFT, wavelet transform, and so forth in software engineering. The previous researches by CNN put a higher stress on hardware construction than being applied to more complicated applications and understanding the nature of images. This thesis hence tries to work on applying CNN to more complicated image applications and associating CNN templates with the characteristics of image files. Therefore, we have proposed a new approach in feature extraction by using CNN's and provided a transformation process to project the original image information into another feature space in this thesis. And this mainly focuses on the following subparts.

B.1 A pre-classified mechanism for different screening images

In the past, the performance of image descreening has been restricted only because the same strategy has to be applied to for many kinds of images. We thus in this thesis proposed a very different structure – the pre-classified mechanism for different screening images. This mechanism makes it easier to deal with various screening images and made a great improvement in image descreening performance. It implies that the CNN structure can also combine with any motivated mechanism from software-oriented status.

B.2 Illustrating CNN outputs in image natures

Our proposed approach in this thesis illustrates the simple CNN outputs in a very different way and also associates the image natures with every adjusted arguments of CNN cell. For image analysis and texture discrimination, the illustrative way of CNN outputs makes the applications by CNN more prevailing and flexible.

B.3 A proliferated structure of CNN templates

The optimization of CNN templates was difficult and insufficient. Thus, we try to propose a proliferated structure to determine CNN templates more adaptively and efficiently. In our proposed structure, CNN templates could be flexibly proliferated for any set of texture patterns. The number of CNN templates that we have to optimize for any prepared texture patterns could be effectively reduced and the optimization process would be shortened in the predefinition of CNN templates.

B.4 A new feature series for texture analysis – Transition Bit String (TBS)

The most significant matters in the difficulty of applying CNN to the higher-level image processing lie in the lack of image information extracted from CNN. Because of this, we proposed the one-dimensional feature curve as well as its feature series in this thesis to provide more chances of using CNN for more applications of image processing. Our proposed feature curves help to understand the natures of textured image patterns and also offer an evaluation index to determine whether the texture patterns that we have to classify in our database are discriminated enough and to decide the number of CNN templates and the corresponding parameters in the optimized templates. As above, we have roughly introduced in this thesis how we could apply CNN in complicated applications of image processing and what kind of relative CNN arguments we have to predefine and determine in advance. To sum up, we have introduced in here the main contributions before getting in the kernel part of this thesis. We did this to make the organization of this thesis more clear and definite.

1.4 Concluding Remarks

In this chapter, we have introduced the related surveys and state-of-the-art of each essential item presented in this thesis. Also, we illustrate the main contributions of this thesis. The structure of CNN's in the past was only applied to some simple and intuitive applications of image processing. Hence we develop more algorithms based on CNN's in order to increase the future applications for hardware implementation. In this whole framework, we need to look for an efficient method to optimize the related parameters which have to be determined in CNN structure. The detailed information of GA has been given in this section and the prominent features of using GA in the design of arguments have also been introduced. As long as an effective approach to deciding CNN parameters can be determined, the key solutions for our chosen applications by CNN's would depend on understanding the characteristics of digital images. In this way, more applications based on CNN's could be brought in the CNN field much easier. In addition, we also work on image documentation, image descreening in particular, since this kind of issues have been the most difficult problems that need to be dealt with by CNN's. So we try to get to know more about the internal characteristics of document images and associate this relationship with CNN's. It is natural for us to design the related arguments in image descreening when implemented by CNN's. In fact, image descreening can be regarded as one of high-order image applications because we cannot handle this problem by a simple strategy for all cases. We firstly give the survey of image documentation in this chapter and later in chapter 3 and 4 we will show the systematic approach based on CNN's in getting this problem clarified. After applying CNN's to image documentation, we focus on incorporating CNN's with some fundamental analysis in image processing. Thus, texture analysis we showed in this thesis provides a versatile

solution for any other image processing techniques when it always plays an essential role in the fundamental researches related to properties of digital images. Unlike the traditional approaches based on CNN's for texture analysis, the structure proposed in this thesis gives a more flexible mechanism for CNN argument optimization and the developed features make texture analysis based on CNN's apply to more complicated problems in image processing. To make our main content of this thesis more clearly, we organized this thesis in the following order: fundamentals of Cellular Neural Network (CNN) and Genetic Algorithm (GA), image descreening based on GA-CNN texture classification, texture discrimination based on GA-CNN proliferation structure, and conclusions and perspectives.

2. Fundamentals of Cellular Neural Network and Genetic Algorithm

Since this thesis describes the methodology and applications that can be implemented by CNN's, we here illustrate the basic structure of CNN and the formation of CNN basic theory. Also, a more generalized model and formulas have to be given in this chapter for high-order image processing. We do not put our emphasis on the discussions of CNN stability and robustness, and we will not over focus on the derivation of dynamic plots in CNN structure because the related proofs of theories have been given in the literature. Instead, we will bring in the image processing field more practical uses by illustrating some basic and simple examples of applications based on CNN's. These simple instances could give broad descriptions about how the elements in CNN templates can be associated with the properties of digital images, which also speed up optimizing CNN templates by GA. That is why we in this chapter introduce the selected CNN templates in the original CNN library for edge and corner detection, image thresholding, connected component detection, half-toning and inverse half-toning, histogram generation. Besides, we will introduce some essentials and important adjustments about GA in the back of this chapter.

2.1 The Fundamental Architecture of CNN

CNN is more appropriate to be the acronym for Cellular Nonlinear Networks than Cellular Neural Network in this thesis since we would use CNN as a nonlinear processing mechanism rather than a learning mapping in some complicated applications. CNN still has the characteristics of networks, a spatial arrangement of locally-coupled cells where each cell can be regarded as a dynamic system with an input, an output, and a defined state which can be prescribed by some dynamic rules. In this thesis, we will put our stress on some applications of image processing based

on CNN's. We hence only illustrate the two-dimensional CNN architecture throughout this section. Before we introduce the whole CNN structure, we would describe the components and variables of an isolated cell. For any arbitrary isolated cell, there are four general variables for the definition of a standard CNN structure. They are the input $\mathbf{u}_{ij}$, output $\mathbf{y}_{ij}$, state $\mathbf{x}_{ij}$, and threshold $\mathbf{z}_{ij}$, respectively, where each variable represents an essential segment in the standard formulas of CNN's, and the subscript indicates the corresponding location in the neighborhood of this isolated cell. In general, the threshold is usually a constant scalar for simplicity and the other three variables can be considered as the functions of continuous time t or the discrete time in the special case. The initial state can be given for some specific application or adjusted at any time during the processes. That is to say, for any fixed threshold $\mathbf{z}_{ij}$, the given initial state $\mathbf{x}_{ij}$, and the processed input $\mathbf{u}_{ij}$, the state $\mathbf{x}_{ij}$ at time t of the isolated cell $\mathbf{C}_{ij}$ will be evolved according to a time variant state function defined below.

$$\dot{x}_{ij}(t) = f_{ij}(x_{ij}(t), z_{ij}(t), u_{ij}(t)) \quad for\ 1 \le i \le M\ and\ \ 1 \le j \le N \tag{1}$$

where the current state $\mathbf{x}_{ij}$ can be seen as the function of combination of the last state $\mathbf{x}_{ij}$, the threshold $\mathbf{z}_{ij}$, and the input $\mathbf{u}_{ij}$. It can be easily observed that all the cells are identical in image applications. Also, (1) can be regarded as the formal ordinary differential equations. Without loss of generality, we have used the simplest output function which is defined as (2).

$$y_{ij}(t) = g_{ij}(x_{ij}(t)) \tag{2}$$

The output function here is simply the predefined transformation from the current state $\mathbf{x}_{ij}$ to the output $\mathbf{y}_{ij}$. The detailed mathematical formulas of an isolated CNN cell were proposed by Chua and Yang in 1988 and widely used in the past applications, which could be listed as follows.

State equation:
$$\frac{dx_{ij}}{dt} = -x_{ij} + a_{ij} f(x_{ij}) + b_{ij} u_{ij} + z_{ij} \tag{3}$$

where a_{ij} and b_{ij} are the weighting coefficients.

Output function:

$$y_{ij} = g(x_{ij})$$

where
$$g(x_{ij}) = \frac{1}{2}(|x_{ij}+1| - |x_{ij}-1|) = \begin{cases} 1, & x_{ij} \geq 1 \\ x_{ij}, & |x_{ij}| < 1 \\ -1, & x_{ij} \leq -1 \end{cases} \tag{4}$$

To illustrate the isolated cell and its related variables, we have Fig. 2.1_1 to simply describe the relationship among theses four variables.

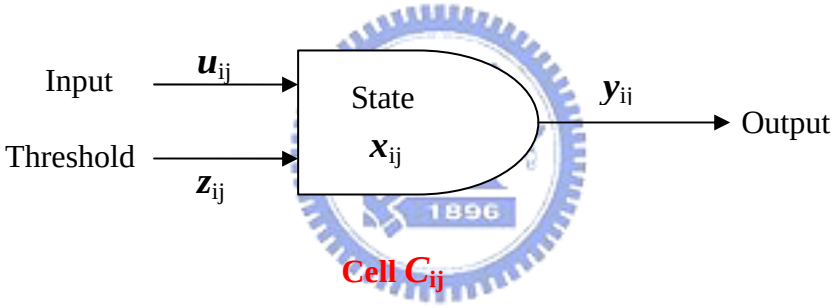


Fig. 2.1_1 Expression of an isolated cell

We have described the simple architecture and the basic theory of CNN, and we will specially introduce the basic CNN formulas and the general CNN model for the digital image applications in the following two sections.

2.1.1 Basic CNN formula

For the image size MxN in the CNN applications, the image size is equivalent to the CNN array and the state equation (3) can be rewritten in a more formal way as (5).

$$\dot{x}_{ij} = -x_{ij} + z_{ij} + \sum_{kl \in S_{ij}} a_{kl} y_{kl} + \sum_{kl \in S_{ij}} b_{kl} u_{kl} \tag{5}$$

where the indices *kl* moves accordingly in the neighborhood of S_{ij} and *ij* can be referred to as each pixel of an image. Hence *i* and *j* would run from 1 to M and N,

respectively. Equation (5) would totally has MxN nonlinear ordinary differential equations. For the basic CNN formulas, the same equation can be defined as (4). In definition, (5) only extends its generality only inside the boundary of CNN array. In order to make (5) more complete, the additional boundary conditions have to be defined. The three boundary conditions which have been used widely in the literature could be listed as follows.

A. Fixed boundary condition

This boundary condition assigns the state x_{kl} of each cell x_{kl} in (5) inside the boundary to be a fixed constant.

B. Zero flux boundary condition

This boundary condition confines the states x_{kl} of the corresponding cells in the neighbor perpendicular to the boundaries to be the same.

C. Periodic boundary condition

In this boundary condition, the first and last rows/columns of the CNN array are similar to be periodic.

After giving the boundary conditions, the preparation of CNN could be done if the initial state for all cells were specified. For image processing in particular, the initial state could be assigned via the gray-level in an image which has to be normalized first between -1 and 1. From all the above essentials in the standard CNN structure, we would have the specific CNN template for the specific application.

2.1.2 General CNN model

The equations that we have introduced for a standard CNN array could not be applied to all applications, so a more generalized CNN model would be preferred by giving different dynamics and coupling laws of each cell. Hence, a general MxN CNN model could be given as long as the following conditions would be specified, i.e. the state equations and the coupling laws of each isolated cell, the boundary and

initial conditions. With the specified definitions in more details, equations (1) and (2) would well describe the general CNN model. As for image processing applications, the ordinary differential equations can be recast in the following matrix form. The matrix representations for CNN arrays provide more choices for image processing, and more image applications could be carried out by incorporating CNN into the simulated processes.

$$\dot{X} = F(X) \tag{6}$$

where

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \cdots & x_{2n} \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ x_{M1} & x_{M2} & \cdots & \cdots & x_{MN} \end{bmatrix}$$

$$F(X) = \begin{bmatrix} f(x_{11}) & f(x_{12}) & \cdots & \cdots & f(x_{1n}) \\ f(x_{21}) & f(x_{22}) & \cdots & \cdots & f(x_{2n}) \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ f(x_{M1}) & f(x_{M2}) & \cdots & \cdots & f(x_{MN}) \end{bmatrix}$$

2.2 CNN Templates for Digital Image Processing

As we know, CNN's can be applied to many applications by specifying the weighting coefficients defined in (3). More systematically, there are several requirements that have to be assigned for a specific image application. Since the specific image processing can be looked upon as a transformation from an input image U (u_{ij} representing the mapped value of each pixel of the input image) to an output image Y (y_{ij} representing the mapped value of each pixel of the output image) with a series of operations, we have to specify the types of input image for using CNN simulation in the first step. Take the gray-scale image and true color image for

instance, we have the first-order and third-order of cells defined for the state x_{ij}, respectively. After that, the initial state and the boundary condition for the specific image application have to be selected or designed according to the specified requirements. Besides, the most important matter in the design of CNN structure associating with some image application lies in the decision of a combination of parameters in CNN's, called CNN template. Once the CNN template could be determined, the problem of image could also be overcome. In the following sections, we would survey several predefined CNN templates for image processing in binary and gray-scale format. We selected these examples of templates in the literature simply because we regard these CNN templates in some basic image applications as the shortcut to the higher-order image applications based on CNN's.

2.2.1 Edge Detection CNN Template

Descriptions: Extract edges of the objects of the input image.

Rules: Each black pixel which has at least one white pixel in the defined neighborhood is defined to be an edge cell.

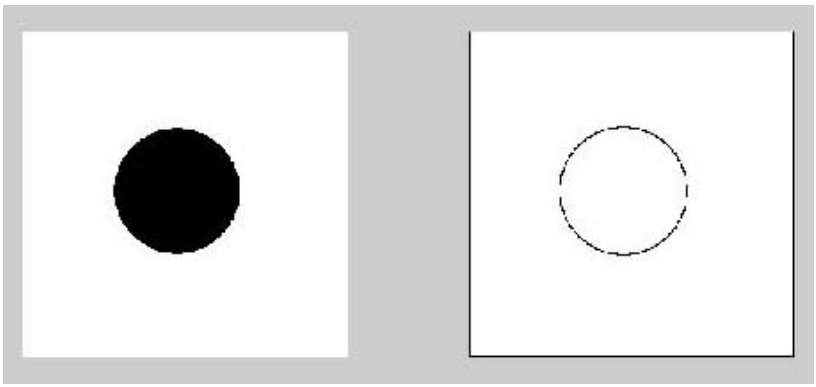
Boundary Condition: Fixed boundary condition

Initial State: $x_{ij}(0) = 0$ (0 means white in an image)

Cloning Template:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad z = -0.5$$

Example:



(a)

(b)

Fig. 2.2.1_1 Edge detection example by some specified template for (a) the binary image (b) the gray-level image (Left: the original image, Right: the processed image)

2.2.2 Color Inverse CNN Template

Descriptions: Inverse colors of the objects in a digital image.

Rules: The color of the input image would be equivalently inversed via the middle of the color range in CNN format.

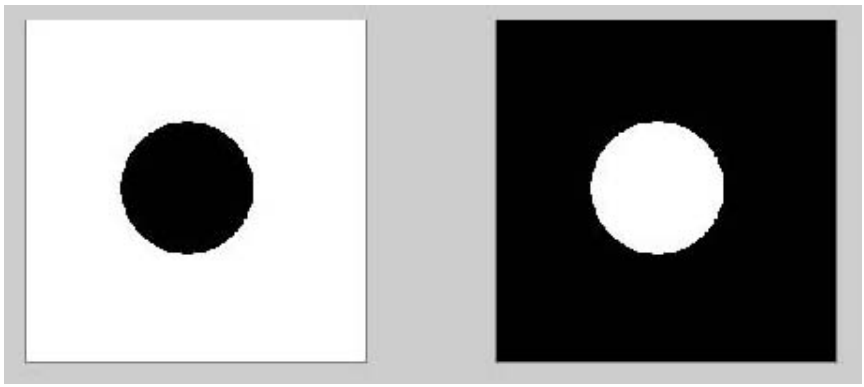
Boundary Condition: Fixed boundary condition

Initial State: $x_{ij}(0) = 0$ (0 means white in an image)

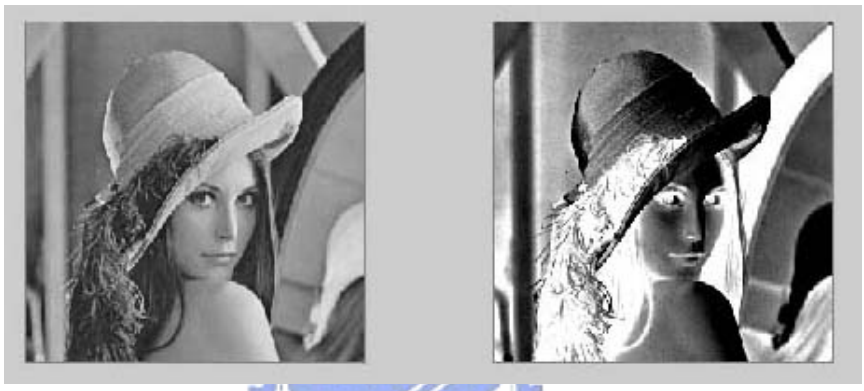
Cloning Template:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -0.25 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2.5 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad z = 0$$

Example:



(a)



(b)

Fig. 2.2.2_1 Inverse operation example by some specified template for (a) the binary image (b) the gray-level image (Left: the original image, Right: the processed image)

2.2.3 Image Thresholding CNN Template

Descriptions: Transform a gray-level image into a binary image by some predefined threshold.

Rules: Each pixel of a gray-level image can be labeled as 'pure white' *if and only* if its transformed intensity in CNN format (between -1 and 1) is higher than some predefined threshold $\boldsymbol{z}^*$.

Boundary Condition: Fixed boundary condition

Initial State: $x_{ij}(0) = u_{ij}(0)$

Cloning Template:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad z = z^*$$

Example:

Fig. 2.2.3_1 Thresholding example by some specified template (Left: the original image, Right: the processed image)

2.2.4 Low-pass Filtering CNN Template

Descriptions: Do the low-pass filtering for the gray-level image..

Rules: The pixels with some specific color range (usually in the low band) can be labeled as 'black' pixels while the others are labeled as 'white' pixels.

Boundary Condition: Fixed boundary condition

Initial State: $x_{ij}(0) = 0$ (0 means white in an image)

Cloning Template:

$$A = not\ specified \qquad B = \frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad z = not\ specified$$

Example:

Fig. 2.2.4_1 Low-pass filtering example by some specified template (Left: the original image, Right: the processed image)

2.2.5 Laplacian CNN Template

Descriptions: Laplacian operation for the digital image.

Rules: Each pixel of the input image would be filtered via the defined Laplace filter.

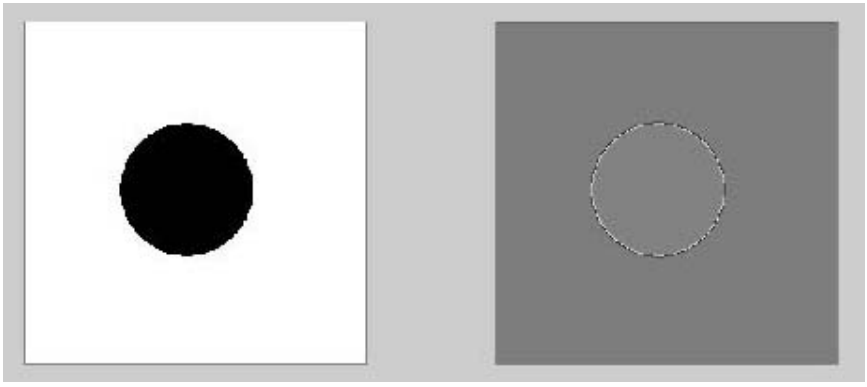
Boundary Condition: Fixed boundary condition

Initial State: $x_{ij}(0) = 0$ (0 means white in an image)

Cloning Template:

$$A = \textit{not specified} \qquad B = \begin{bmatrix} 0 & -0.25 & 0 \\ -0.25 & 1 & -0.25 \\ 0 & -0.25 & 0 \end{bmatrix} \qquad z = \textit{not specified}$$

Example:



(a)

(b)

Fig. 2.2.5_1 Laplacian operation example by some specified template for (a) the binary image (b) the gray-level image (Left: the original image, Right: the processed image)

2.2.6 Half-toning CNN Template

Descriptions: Transform a gray-level image into a half-toned binary image which preserves the major characteristics of the original image.

Rules: The gray-level intensity of an input image would be re-sampled to obtain a binary image via half-toning which could resume the original image intensity by inverse half-toning.

Boundary Condition: Fixed boundary condition defined by the constant values of the initial state and the input image.

Initial State: $x_{ij}(0) = u_{ij}(0)$

Cloning Template:

$$A = \begin{bmatrix} -0.07 & -0.1 & -0.07 \\ -0.1 & 1.15 & -0.1 \\ -0.07 & -0.1 & -0.07 \end{bmatrix} \quad B = \begin{bmatrix} 0.07 & 0.1 & 0.07 \\ 0.1 & 0.32 & 0.1 \\ 0.07 & 0.1 & 0.07 \end{bmatrix} \quad z = 0$$

Example:

(a)

(b)

Fig. 2.2.6_1 Image half-toning by some specified template for (a) the natural gray-level image (b) the human gray-level image (Left: the original image, Right: the processed image)

2.3 CNN Justifications

In this section, we shall discuss the related justifications about CNN. Also, we would like to elucidate some querying points for those who might be concerned about using CNN in the related applications of image processing in three aspects.

A. Discussions of CNN equations

As what we have mentioned in the earlier part of this chapter, the most crucial part of applying CNN to image processing is to determine the corresponding arguments of CNN templates. In addition, the structure of CNN depends on the state and output equations as (3) and (4). Those who are good at image processing but not familiar with CNN might doubt the relationship between these two equations and might be curious about the resultant images after using CNN. In fact, many original studies

about CNN have proved that the processed images after CNN should be in the binary form on account of the following theorem [68].

Theorem 2.3.1.

The output y_{ij} *of every cell at any stable equilibrium point of a completely stable standard CNN defined by (3) and (4) is equal to either plus 1, or minus 1, if the center element* a_{ij} *of the A template satisfies* $a_{ij} > 1$.

Also, without loss of generality, the processed images could be binarized if $a_{ij} \neq 1$ as to meet the bistable criterion of CNN.

B. Justifications of CNN circuit

Some might be concerned about the pros of cons by using CNN. Therefore, in this section we shall briefly introduce some significant reasons for applying CNN's to applications of images. As we know, CNN is very attractive in its successful implementation of an analog input/output CNN universal machine which can also be referred to as a CNN universal chip. The analog circuit is very different from the digital one in the sense that the analog could truly transmit and process most signals in the undistorted way. Furthermore, a single silicon chip is regarded as a completely dynamic array stored-program computer where the CNN chromosome can be executed and programmed on the chip at a tremendously high speed (given 10^{12} analog instructions per second based on a 100x100 CNN chip). Also, the chip design by the CNN universal machine would differ from that by a digital circuit because it could be carried out in the completely nonlinear dynamics. Hence, we would like to take advantage of some dominant and distinguishing characteristics of CNN's in more complicated applications of images.

C. Sensitivity of CNN's

In addition to some justifications of CNN's stated above, some might be

concerned about its sensitivity when applying CNN to various applications. In this thesis, we used CNN's in image documentation and texture analysis. For CNN applications that we discussed in this thesis, we have to find the appropriate templates of CNN with respect to some specific purpose. We in here make use of GA to optimize the related arguments of CNN templates. Naturally, the desired patterns in some sense should be specified beforehand. Unlike the concept of classic cellular automata, CNN cannot be defined by local rules directly but can be simply defined by iterating the CNN genes. That is why GA could be broadly applied to every sort of image applications. Speaking to the proposed applications in the chapter 3, we do not need to accurately pre-specify the desired patterns of CNN templates because only the ratio of black and white pixels in the CNN output would be quite sufficient to provide a proper decision in the later processes. The related explanations about sensitivity in optimizing CNN arguments when using GA have been given in some important works of the literature [13], [43]. As they concluded, the form of desired patterns in black/white or white/black arrangement has no influence on the final results even for some applications in need of high accuracy. Therefore, in the point of view in using CNN for pattern selection, the arrangement of black/white or white/black in setting the desired patterns shows no difference in the final performance if GA is applied to optimize the CNN templates.

2.4 Concluding Remarks of CNN

It is observed that the CNN templates in all introduced examples are defined by the sphere of influence in radius $r = 1$. That is to say, only 19 parameters in the triplet template $\{A, B, z\}$ have to be determined. The main reason is that this kind of CNN template would be much more efficient for various image applications by experienced rules and practical uses. Besides, the examples demonstrated above try to associate

the design of CNN templates with some image characteristics in the specific application. These kinds of arrangement in CNN templates not only reveal the intuitive decision of CNN templates but also motivate our approach in analyzing the nature of images and inspiring the design of CNN templates. After that, we especially state the justifications of CNN in three aspects in order to make those who like to use CNN in image applications more comprehensive on these concerning topics. We hence make use of a more effective training and learning approach for optimizing CNN templates in our proposed high-order image applications based on CNN's. This optimization method, genetic algorithm (GA), would be given in more details in the next section.

2.5 Genetic Algorithm (GA)

There exist two kinds of optimization approaches where one is the specific optimization rule and the other is the generalized one. The specific design rule is defined by some specific characteristics of object functions. The object functions have to satisfy some properties such as linear, invariant, differential properties, so the differential and gradient approaches are of this type. As for the generalized one, the design rules are independent of object functions, and random searching as well as genetic algorithm should be of this type. For the design of CNN templates, GA would be a better choice since no direct relation between CNN templates and image applications can be found by intuition. The most significant advantage of GA lies in the wide range of searching, and this makes optimization of CNN templates more effective and accurately. Later in this section, we will introduce some necessary components and important properties of GA, which have a great influence on the design of CNN templates.

Genetic algorithms are based upon a kind of optimization searching mechanism

during the process of natural selection which spiritually imitates the natural rule, the survival of the fittest. As the evolution rule of genes, GA can choose a better parent generation in the species which has the superior properties and randomly exchanges the genetic information with each other. In this way, the children generation could be expected to be superior to the last parent generation and the best species would be generated by this repetitive way. In fact, there are three major operators for GA including reproduction, crossover, and mutation. The fundamental concept of GA applied to optimization lies in the following essentials, i.e. encoding all the searching arguments to so called chromosomes, design of fitness function by the given acquirements, selection of the species with higher fitness values for mating pool, and crossover and mutation for the worse reproduction result. Fig. 2.4_1 illustrates the whole process of GA.

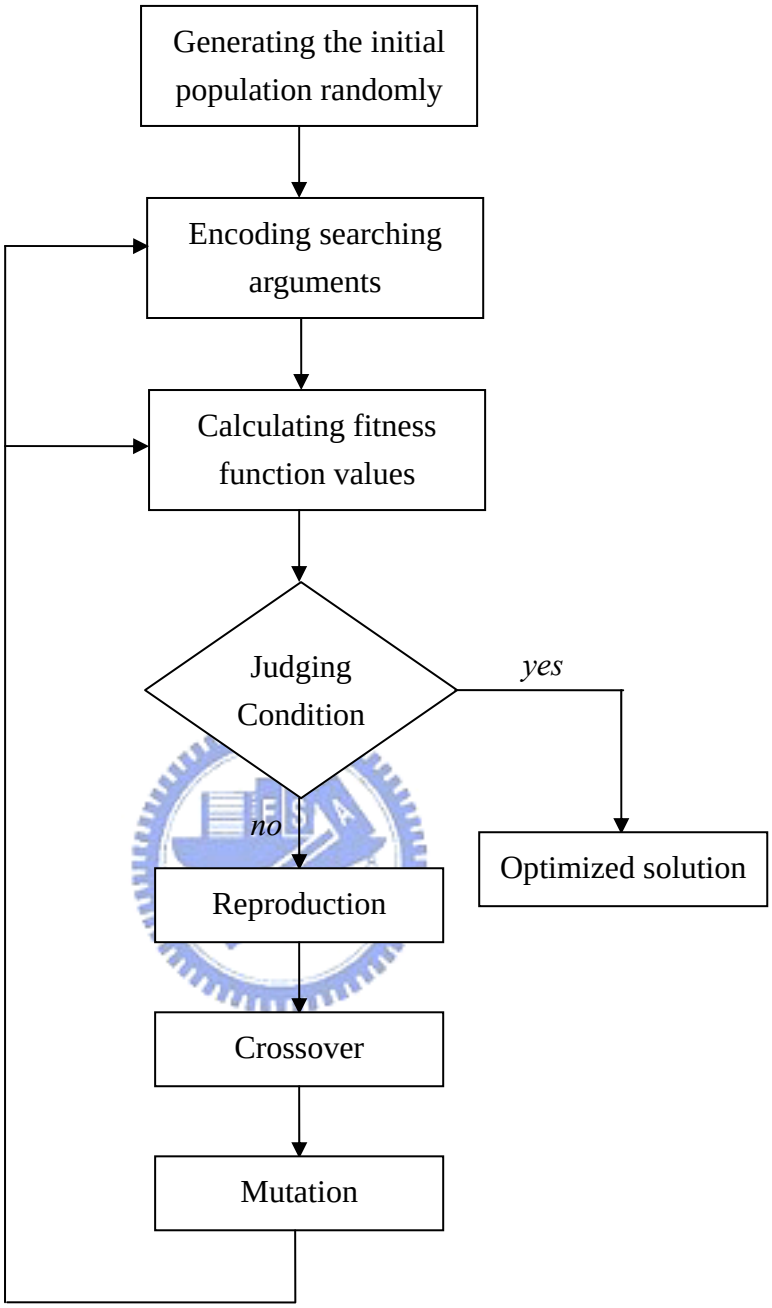


Fig. 2.4_1 Flow chart of the systematical GA structure

2.6 Essentials in GA

After introducing the whole structure of GA, in this section we will give more detailed information about the essentials in GA which are organized as below.

A. Reproduction

Reproduction is an operation process which can decide the quantum of

species of the next generation that need to be eliminated or reproduced according to how fitting the species of the parent generation can be. The species in higher fitting capability would be reproduced much more while those in lower fitting capability would be excluded in the next generation. The fitting capability that we mentioned above can be estimated by the responses of fitness function. In general, there exist two types of reproduction where one is the roulette wheel selection and the other is the tournament selection.

a. Roulette wheel selection

During the process of evolution, the area of each component of the roulette wheel can be determined by its corresponding fitness function value. The bigger the fitness function value, the larger the corresponding area of roulette wheel. The proportion of occupied area of roulette wheel also indicates the probability of being selected for the mating pool. Assume that there are N species and f_i represents the fitness function value of the ith species. Theoretically, the total number of this kind of species, $N \cdot f_i / \sum_i f_i = f_i / \bar{f}$ *where* $\bar{f} = \frac{1}{N}\sum_i f_i$ and $\bar{f}$ means the average fitness function value of all species, would be reproduced and selected to the mating pool.

b. Tournament selection

The major difference between tournament and roulette wheel selection is that tournament selection randomly selects two or more species at the first run and the species with the biggest fitness function value would be sent to the mating pool during the process of evolution.

Basically speaking, these two selection methods both can achieve the goal of survival of the fittest. Obviously, tournament selection is good at its processing speed on account of less complexity and roulette wheel selection is fascinating in its

generality and wide applications if calculation complexity is not the most significant matter in the discussing issues.

B. Crossover

Crossover would randomly select two strings of species in the parent generation from mating pool and exchange their information to generate two new species. In pursuit of generating the superior children generation, crossover could take advantage of outstanding information from the last generation. The probability of crossover depends on how often the process of crossover could occur which includes the following three types, one-point crossover, two-point crossover, and masking crossover. For an easier comprehension, it is sufficient for us describe these three types of crossover by illustrating one example encoded by strings as below.

a. One-point crossover

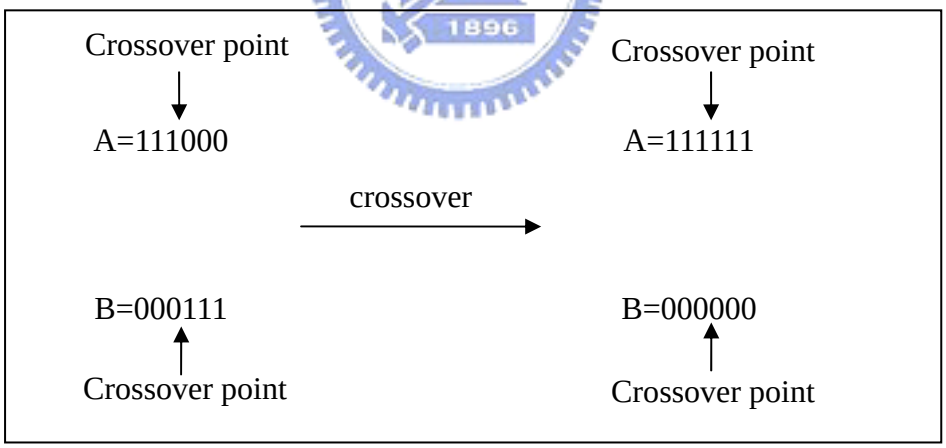


Fig. 2.5_1 Illustrating figure for one-point crossover

b. Two-point crossover

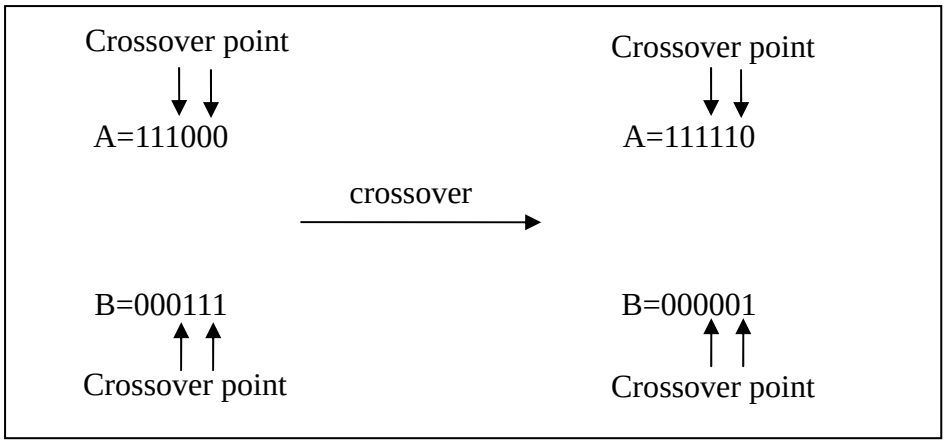


Fig. 2.5_2 Illustrating figure for two-point crossover

c. Masking crossover

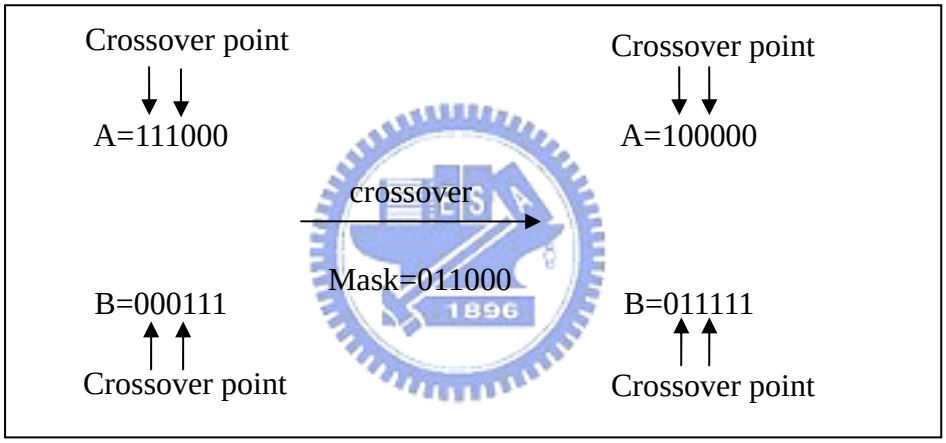


Fig. 2.5_3 Illustrating figure for masking crossover

C. Mutation

During the process of mutation, one string/ species and the location of mutation point would be chosen and then the information of corresponding species at mutation point would also be changed. The probability of mutation depends on the process of occurring mutation, and the location of mutation point could be defined as a bit, a whole string, or by a specific mask. The types of mutation hence could be defined by the location of mutation point as what we have introduced for the crossover process.

GA could be equipped with these three introduced essentials and its type is

chosen by the characteristics of some specific image application. In our practical use by GA, we would give more information about how we could determine the types of each essential and its pros and cons in the later topics of this thesis.

2.7 Discussions of GA

GA is basically different from the traditional approaches and can be characterized by the following issues.

A. GA cannot be operated on the parameter itself but the set of parameters, so it can jump away from the constraint of analysis in search space.

B. GA cannot search a single point but many points in search space, so it can obtain the global optimum sooner and avoid finding the local optimum.

C. All GA needs is the information of fitness function, that is to say, it is not necessary for GA to use other aided information like gradient. In this way, GA can use many kinds of fitness functions so as to save complexities resulted from over mathematical computations.

D. GA can be applied to various sorts of optimization problems since it does not solve problems by well-defined rules but use the probabilistic rules to make the searching direction more clear.

In the remained part of this section, we will focus on some important discussions about the adjustment of GA structure.

2.7.1 Parameter settings

The most complicated and difficult matters in GA would be the settings of related parameters. It is impossible for us to develop a standard method for all cases to set all the parameters of GA here. Instead, we will give broad descriptions about where we could put more emphasis and how the parameters can be related to each other. The major parts that one needs to design GA parameters for a specific case are

the range of encoding, length of string, size of population, mutation and crossover rate, and the design of fitness function. These aspects would have a great impact on the final searching result and between them the close relationship and dependence exist in process of optimization. For the setting of rates in some essentials of GA, we will show in the practical applications of this thesis. As for the kernel part of GA, we will discuss the problems appeared frequently when using GA and we will also introduce some existed types of fitness functions in this section.

2.7.2 Global optimization

The convergence speed of GA could be governed by the population size, crossover rate, and mutation rate. A higher value may speed up convergence of GA and then result in finding the local minimum. On the contrary, a lower value may cause the difficulty of convergence. Therefore, we have to take into consideration the dimensionality of search space, the range of parameters, and the string length while encoding. To avoid giving a higher crossover or mutation rate, we can predefine its corresponding maximum and minimum. In one way, the setting of theses two parameters can be initiated at the maximum and then decayed to the minimum along with the growing generations. Another flexible approach is using the floating mutation or crossover rate, since the local optimization might be achieved if the species have stopped evolving. This approach invokes to add some artificial interference in order to simulate the generation of new species, and this can be implemented by switching the mutation and crossover rate. For simplicity, we can increase the generated speed of new species (mutation and crossover rates) if the best fitness function value has kept unchanged through generations, and also we will get mutation and crossover rates back if the best fitness function value has increased. It is apparent that we can avoid finding the local optimization value as long as these adjustable parameters in GA could be effectively determined. Besides, the other

crucial essential of GA, fitness function, will be introduced in the next section in much more details.

2.7.3 Fitness Function

The selected fitness function should be capable of reflecting the differences between different species. Besides, fitness function ought to exclude the inferior species and offer more chances for new species to get in the populations. Sometimes, diversity of species in the mating pool can help to generate a better optimized solution since choosing the same superior species from the mating pool might bring about a worse optimized result. To achieve the above requirements, how to adjust the fitness function becomes more important. Thus, there exist three kinds of methods to adjust fitness functions in the history of GA. In the followings, we shall introduce the basis and the pros and cons of these three methods.

A. Linear scaling

Linear scaling can transform the values calculated from fitness function by the following (7).

$$f' = af + b \tag{7}$$

where f and f' are the original and transformed values of fitness function, respectively, and the coefficients *a* and *b* can be chosen such that

$$f_{avg} = f'_{avg} \qquad and \qquad f'_{\max} = \kappa \cdot f_{avg} \tag{8}$$

where f_{avg} , f'_{avg} and $f'_{\max}$ are the original and transformed average values of fitness function, and the transformed max fitness value, respectively. The coefficient κ is the multiplier between f_{avg} and $f'_{\max}$. The linear scaling function is shown in Fig. 3.3.3_1. In the point of view by statistics, (8) can ensure one of the chromosomes with the average fitness value to be selected to the next generation because of the

condition $f_{avg} / f'_{avg} = 1$. Also, the total number κ of chromosomes could be reproduced to the next generation by the assumption $f'_{\max} / f'_{avg} = \kappa$. In using this scaling method, we have to avoid the negative fitness values after transformation.

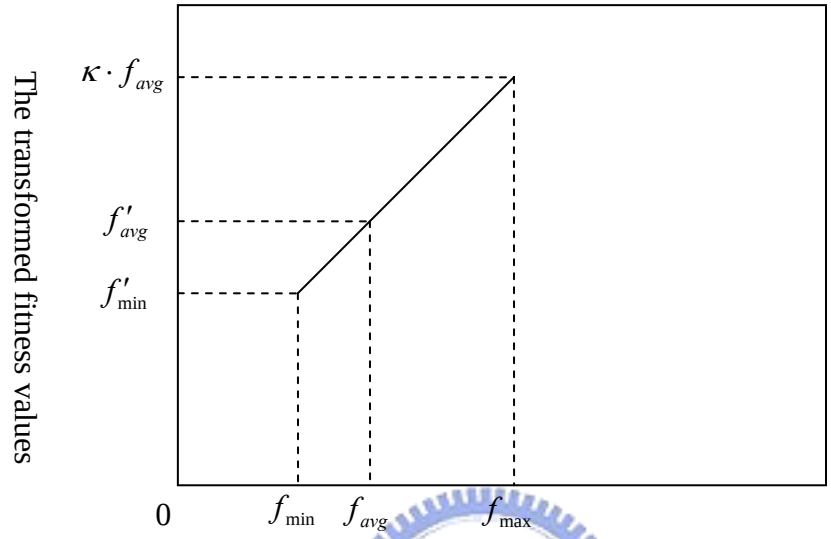


Fig. 2.6.3_1 Illustrating figure for linear scaling

B. Sigma truncation

In addition to avoiding the negative fitness values, linear scaling can map the fitness values into our expected range. However, most chromosomes of the population will have the higher fitness values when the searching process of GA tends to be mature, which might cause the negative fitness values if the minority of chromosomes has a very low fitness value. To overcome this problem, sigma truncation uses the information of variance before scaling the fitness function. This can be carried out by the following equation (9).

$$f' = f - (\bar{f} - m \cdot \sigma) \tag{9}$$

where the notation is like linear scaling and *m* depends upon a multiplier of variance of the fitness values in the population. By the help of sigma truncation, we can avoid the negative fitness values resulted from linear scaling.

C. Power law scaling

Beside of sigma truncation, power law scaling can also help adjust the fitness function of GA. This scaling method makes the transformed fitness values be the power law of the original fitness value like (10).

$$f' = f^{p} \tag{10}$$

where the decision of p is up to the issuing problem. Also, p has to keep changing during the searching process of GA to meet the requirements of extending the range space of fitness values.

2.8 Concluding Remarks of GA

In this section, we not only illustrate the fundamentals of GA but also describe the pros and cons of related essentials in GA. For some essentials in GA such as crossover, mutation, and fitness function, we have given a detailed introduction about these components. It is simply because any specific applications can be solved by GA only depends on the right choices of fitness function and the appropriate GA settings. For the specific image application, we shall induce a complete methodology by using GA in the following two chapters. With the equipment of GA information introduced in this chapter, we could make the following explanations about using GA much easier and offer more ways of thinking for the adjustments of GA parameters applied in our problems.

3. Image Descreening based on GA-CNN Texture Classification

3.1 Introduction

We in this chapter propose a new image descreening technique based on texture classification using cellular neural network (CNN) with template trained by genetic algorithm (GA), called GA-CNN. Instead of using the fixed filters for image descreening, we are equipped with a more pliable mechanism for classifications in screening patterns. Using CNN makes it possible to get an accurate texture classification result in a faster speed by its superiority of implementable hardware and the flexible choices of templates. The use of GA here helps us to look for the most appropriate template for CNN more adaptively and methodically. The evolved parameters in the template for CNN can not only provide a quicker classification mechanism but also help us with a better texture classification for screening patterns. After the class of screening patterns in the querying images is determined by the trained GA-CNN-based texture classification system, the recommendatory filters are induced to solve the screening problems. The induction of the classification in screening patterns has simplified the choice of filters and made it valueless to determine a new structured filter. Eventually in this section, our comprehensive methodology is going to be topped off with more desirable results and the indication for the decrease in time complexity. This chapter is organized in the following section orders: the proposed system architecture, screening texture classification, design of CNN templates for screening texture classification by GA, selection of descreening filters, adaptive determination of arguments in the chosen descreening filter, experimental results, and concluding remarks.

3.2 The Proposed System Architecture

The proposed image descreening system can be divided into three major processes. At first, we make use of GA to determine the working template elements for the required operational function of CNN. After the working templates are determined, the required function of CNN is determined at the same time. And then, in the execution of CNN, the output can only be a binary image. For this reason, several useful indices extracted from the output image of CNN are introduced to indicate the property of screening in the target image. By the information obtained from the output of CNN, the types of descreening filters and the parameters in the filter are then obtained as desired. Finally, in the descreening stage, we apply the selected filter to the screened images to filter out the screenings.

The proposed image descreening technique consists of the training phase (Fig. 3.2_1) and descreening phases (Fig. 3.2_2). At the beginning, we have to prepare screened images on which the features of the screening patterns are extracted for training purpose. In the training phase, a block in the size of 64*64 pixel^2 is extracted manually from the original screened images in our database. A block in the same size, of course, would be cropped automatically in the testing phase. Also, in the training phase, the proper template of CNN is determined by GA to perform texture classification with the derived template. Two screening estimates calculated from the output of CNN in the training phase can be utilized to be the matching indices for that in the testing phase. The closer the calculated values in the testing images to those in the training images are, the higher the level of tendency toward that screening pattern will be. These two screening estimates not only suggest the types of descreening filters, but also determine such arguments like width, radius, or size in the selected filter. Once the type of the screened image has been recognized according to the

extracted screening pattern, this screened image will be convolved with the suggested filter to acquire the final descreened image.

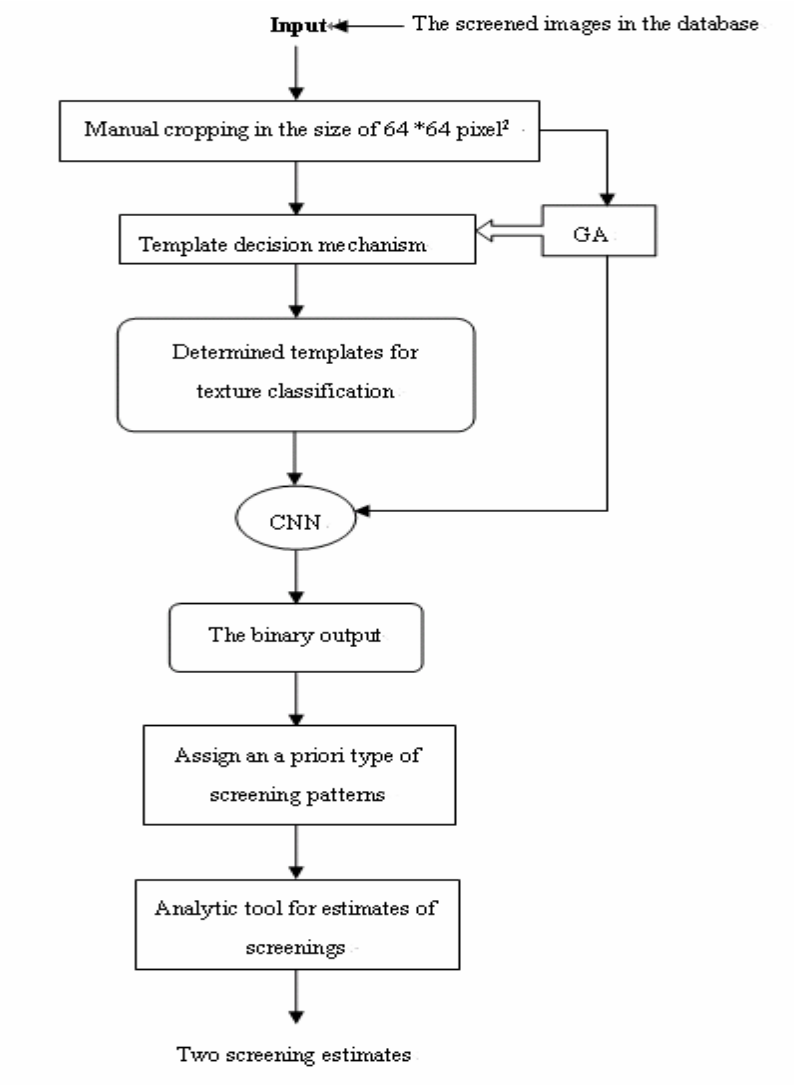


Fig. 3.2_1. Flowchart of the training phase of the proposed GA-CNN-based texture classification scheme.

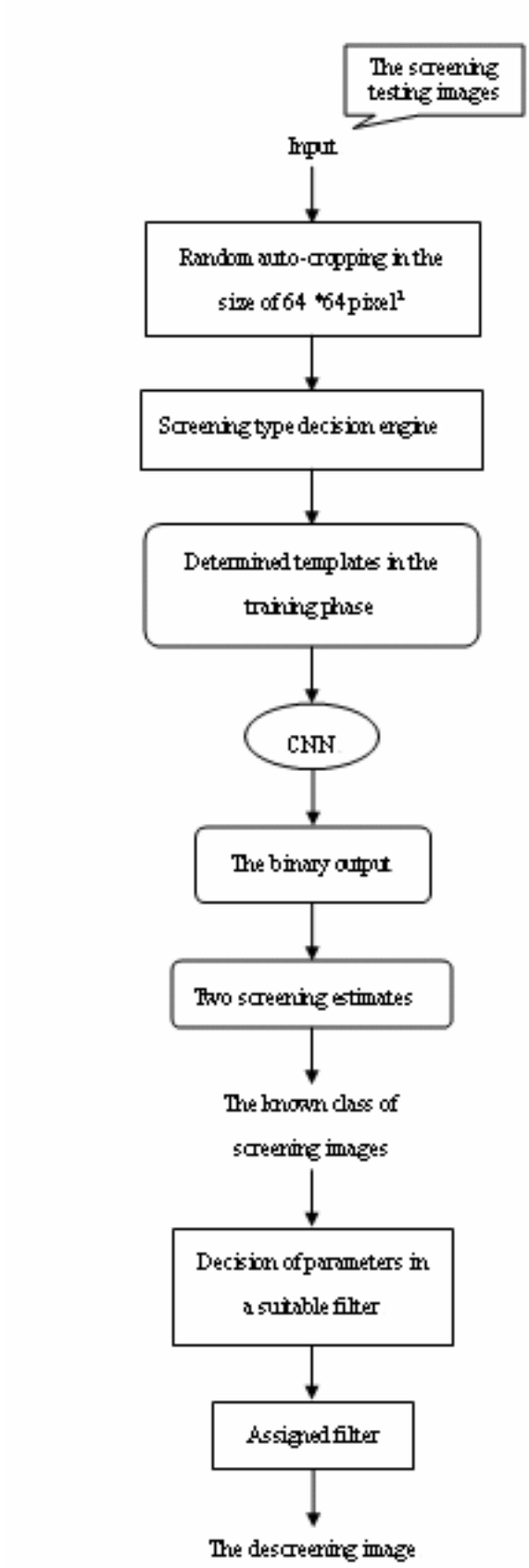


Fig. 3.2_2. Flowchart of the proposed image descreening technique.

3.3 Screening Texture Classification

There are diverse screening patterns in the screened images; each can be best removed by a specific kind of filters, so the lack of the screening information in the screened images restrains the descreening results and complicates this kind of problems. Textures can be viewed as complex visual patterns composed of entities, or sub-patterns, that have characteristic brightness, color, slope, size, etc. [44]. Generally speaking, four kinds of popular approaches nearly dominate all the researches in the texture analysis, i.e., structural, statistical, model-based, and transform methods. Structural approaches [44] ~ [45] represent texture by well-defined primitives (*micro-texture*) and a hierarchy of spatial arrangements (*macro-texture*) of those primitives. As to model-based texture analysis [46] ~ [52], it uses fractal and stochastic models, and attempts to interpret an image texture by using generative image model and stochastic model, respectively. Finally, transform methods for texture analysis, such as Fourier [51], Gabor [52] ~ [53] and wavelet transforms [54] ~ [56] represent an image in a space whose co-ordinate system has a strong understanding that is closely connected with the characteristics of a texture, like scales or frequency.

Our proposed texture analysis scheme combines the advantages of structural methods with those of statistical ones. The screening patterns can be referred to as primitives in structural methods since the classification mechanism is a supervised one. In the descreening phase, smooth indices are used to determine whether the screening patterns could stand for the screenings in the original images. If so, the screening pattern extracted from the original image will be fed into our classification engine, GA-CNN. As soon as the type of screenings in the testing image is identified, the following processes will be much easier and the descreening performance will be

better.

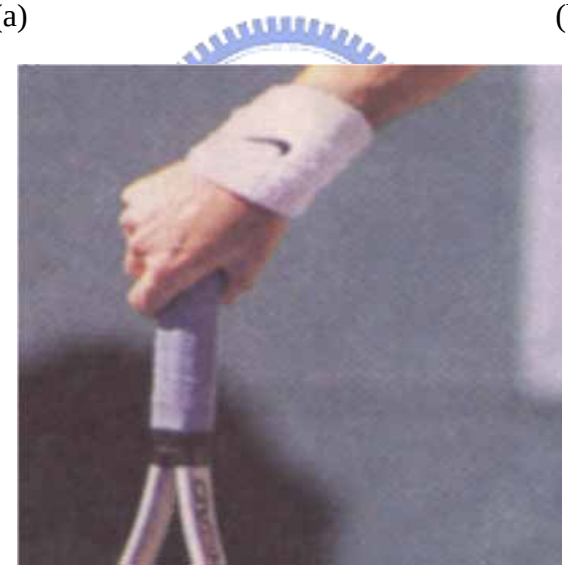
3.3.1 Screening-Texture Patterns

Screening patterns can be viewed as one kind of textures that represents the similarity grouping in an image. It may be difficult to get this sort of similarity in the screened images accurately. It takes no effort, however, to observe the regularity of the same screened images by human perception. For example, Fig. 3.3.1_1 (a)-(c) contain different types of screening patterns, each of which has its own regularity or uniformity, called texture of its own. Fig. 3.3.1_1 (a) contains screenings with smaller granulations; while Fig. 3.3.1_1 (b) has bigger or squared screenings. For the image corrupted by a specific screening pattern, a proper filter should be used for descreening. The total number of screening patterns for all the screened images can not be known beforehand. How to determine the number of screening patterns depends on the desired functional performance. In our experiments for the descreening purpose, only two classes of screening patterns are classified. Our experiments have showed that the descreening performance after two-class screening classification is very satisfactory and acceptable to human perception. The two classes of screening patterns that are cropped manually from the screened images in our database are showed in Fig. 3.3.1_2 and Fig. 3.3.1_3, respectively. These patterns and manual classification results will be used for the training of the proposed GA-CNN texture classifier.

In the testing phase of the trained GA-CNN texture classifier, it is essential to identify the proper block(s) in the testing image for screening texture identification and classification. We shall propose a set of *smooth indices* calculated from an image block for determining whether the extracted block is qualified to be one of the screening patterns in the testing images. Smooth indices thus play a critical role in the screening classification. The derivation in more details for smooth indices will be

depicted in the next subsection.

(a) (b)



(c)

Fig. 3.3.1_1 Three example screened images. Images (a) – (c) contain different types of screening patterns.

Fig. 3.3.1_2 Screening Example 1

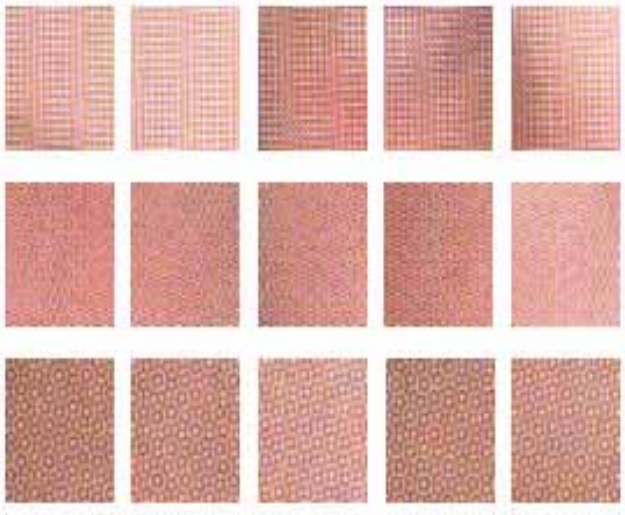


Fig. 3.3.1_3 Screening Example 2

3.3.2 Smooth Indices for Screening-Texture Block Detection

The standard deviations in statistical approaches were always used as the analytical tool for signal processing and image processing. The smooth index used in our approach can be also represented in terms of the standard deviation; it is obtained from the proportion of difference in the standard deviations. The standard deviation in an image represents the extent of difference in intensities of an image. Looking for an index which can best describe how smooth a block will be, we have made use of the proportion of difference in the standard deviations between our defined blocks, named the difference ratio of standard deviations.

The size of the screening pattern that we extracted from one of screened images is 64 by 64 pixels. We partition the extracted screening pattern into five parts (i.e., sub-screening patterns). Figure 3.3.2_1 illustrates this partition of the screening pattern. For each of them, a standard deviation value has to be calculated, and then the calculated value needs to be compared with that of the original screening pattern. With regard to all of these five sub-screening patterns, five difference ratios can be acquired for a screening pattern in a screened image. In the same way, four difference ratios can also be obtained about the central part from which is partitioned off the screening pattern. None of these nine numerical values being larger than ten percents totally makes sure of the smoothness of the extracted screening pattern. This smooth index can be tuned higher if the original document image is not that smooth and the components of edges may be more dominant. Although the smooth index is relevant to the screened images, it is ranged from 10% to 20% without respect to the complexity of the screened images. As Fig. 3.3.2_1 shows, each of the five blocks has half of the size of the original screening pattern, 32 by 32 pixels. The higher the smooth index is, the coarser the extracted screening pattern is. A smaller smooth index implies the uniformity and regularity of the extracted screening pattern. The similarity among blocks can truly reveal the agreement to the screening pattern coming from the testing screened image.

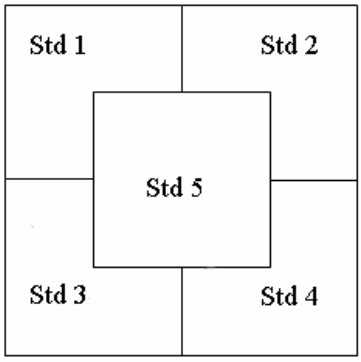


Fig. 3.3.2_1 Partition in the screening pattern for calculating the smooth indices

3.4 Design of CNN Templates for Screening Texture Classification by GA

The core technique in the proposed descreening method is to design a CNN-based screening texture classifier. The subject of template design or learning is the most important topic in CNN researches. The methods which have been investigated may be classified as local learning algorithms [5], [6], [57], global learning algorithms (GA) [7] ~ [8], and analytical methods [3], [4], [58], [59]. Local learning algorithms are derived from training methods developed for other neural networks such as multilayer perceptrons, and their global counterparts mostly use stochastic optimization techniques such as genetic algorithms [7] or simulated annealing [8]. The analytical approaches are based upon a set of local rules characterizing the dynamics of a cell, depending on its neighboring cells. These rules are transformed into an affine set of inequalities that need be solved to get correctly operating templates. Actually, GAs inherit the properties of analytical methods and global methods. At the same time the local information can also be retained in the template training stage.

For texture analysis by using CNN based upon GAs, some relevant and representative studies as [43], [60] ~ [65] have to be taken for further discussions. Like [43], a strategic approach was proposed to provide a simple but complete methodology for texture classification and segmentation. Indeed, the features mentioned in this reference, ROB (ratio of black pixels) and average gray-levels, give a very helpful tool to classify up to sixteen texture patterns after a series of processing units or analytic steps. But for some specific textures like screening patterns mentioned in this chapter, the characteristics proposed earlier are not sufficient to classify and segment these screening texture patterns since there may be various causes for screenings such

like different resolutions, sampling rates by documental scanning, printing processes, and so on. In [62], the hardware implementation for texture segmentation was developed, which sped up the research and discovery of texture analysis and other related applications, and meanwhile made texture-specific filtering and evaluation processes facilitated with parallel handling capability and consecutive template training of CNN's. The uses of gray level histograms or other statistical methods in [43], [60] ~ [62] also modulate the decision procedure after texture classification. In addition to filtering textures in some orientation, having a process of cross-correlation between the state and input, and using the halftone-like output of CNN as in [65], our proposed approach turns the statistical analysis of CNN and the classification of screening patterns in advance for descreening on a differently breaking view. Two useful estimates, therefore, have been deduced here to solve this kind of descreening problem exactly.

3.4.1 The Settings of CNN Templates

A cellular neural network is innovated with implementable circuits and the basic characteristics of structures of neural networks. For the image processing in CNN, the dynamics of the network is governed by a system of $n=MN$ differential equation (11):

$$\frac{dx_{ij}(t)}{dt} = -x_{ij}(t) + \sum_{k,l \in N_{ij}} (a_{k-i,l-j} f(x_{kl}(t)) + b_{k-i,l-j} u_{kl}) + I + n_{ij}(t),\ (i,j) \in \{1,2,\cdots M\} \times \{1,2,\cdots N\} \quad (11)$$

where N_{ij} denotes the neighborhood of the cell C_{ij}, a_{kl} and b_{kl} are the feedback and the control template parameters, respectively, M and N are the height and width of an image, respectively, and $n_{ij}(t)$ is the disturbance term. CNN has presumed circumscribed by a virtual contour of cells with the constant input and output for the lack of a complete set of regular neighbors on the boundary cells. The correspondence subsumed under $\partial_{ij} f(\bullet)$ is the piecewise linear saturation function defined as (12).

$$f(x) = \frac{1}{2}(|x+1| - |x-1|) \tag{12}$$

and $y_{ij}(t) = f(x_{ij}(t))$ is referred to as the output of C_{ij}.

A template contains the combination of a triplet {A, B, z} for the template learning, where A consists of all the arguments a_{kl}, and B represents the values b_{kl}. In the structure of CNN, A and B both are 3*3 or 5*5 matrices. The size of matrices depends on the functions of CNN. The threshold z is a one-dimensional scalar. In our system, we apply 3*3 matrices to both A and B in the consideration of functional performance for the texture classification. The constant input is chosen to be the original image; i.e., the extracted screening pattern in gray scales. The initial state in CNN for the texture classification can also be set as the original screening pattern due to the ability of convergence. The task now is to design proper templates such that the binary output of the corresponding CNN can indicate the class that the input screening pattern in gray scales belongs to.

3.4.2 Parameter Adjustments in GA

A genetic algorithm (GA) is a kind of methodology, which can accomplish the specific task or processing by a series of rigorously mathematical operations and logical decisions. With the ability of selection, crossover, mutation, and reproduction in genes, the superior offspring will survive while the inferior one being excluded through competition. The most important part in GAs lies in the “fitness” function, a kind of objective or cost function that covers all the information about the discussions, which is in turn yielded to evaluate the fitness value of all the possible solutions and their performances. Trying to reach a higher fitness value by recombining chromosomes has then become an essential constituent in this parameter space.

When it comes to the operational theories in GAs, schema or template theorem should have been taken to explain how it works. In the followings, some basic

definitions in schema would be simply illustrated. A binary number “1xxx01xx” presented here could be viewed as one of the examples of schemas where the symbol ‘x’ means “Don’t care”. The values located in ‘x’ could be 0 or 1. Also, a binary number with or without ‘x’ is a schema. For a binary number “101”, thus, all possible eight schemas are {101, x01, 1x1, 10x, xx1, x0x, 1xx, xxx}. There may include 2^n schemas for an n bit binary number.

The operational theories in GAs are based upon the building blocks hypothesis, where the building blocks are the schemas applied for schema theorem. A good contour could be conserved in case of selections. A better result could be shown up in case of crossover. GAs could only find the optimization by getting rid of the local traps in case of mutation. From above of all, the schemas with the smaller order, the shorter defining length, and those beneficial to looking for the optimization would survive generation by generation.

3.4.3 CNN Template Design by GA

When GA is used as the training tool for CNN templates, a few conditions that ought to be met need to be taken into considerations. To make sure of the stability in CNN, the positive cell-linking [1] and symmetry [66] of the “A” template have to be verified. Fortunately, any template optimized by GAs must be stable because the lower fitness values always result from unstable trajectories in CNN. It ensures the stability in CNN if a higher fitness value could be reached, which is exactly what GA goes for. The well-known evaluation function is defined as (13) by the mean squared error:

$$g(j) = \sum_{i=1}^{k} (y_i^d - y_i^s)^2 \quad , \tag{13}$$

where j is the training template including A, B, and z, k is the number of cells which

depends on the size of templates, y_i^d is the desired output for each i pixel, and y_i^s is the output in steady state for each i pixel. Obviously, only minimizing $g(\bullet)$ can make the training templates become our desired templates, which just goes against our purpose of getting higher fitness values. Therefore, a modified mapping function is presented in Fig. 3.4.3_1 to transform the original cost function $g(\bullet)$ into the required fitness function $f(\bullet)$.

This mapping function mixes both advantages of windowing and linear scaling. Windowing is one sort of fitness functions that can assign a constant minimum fitness value to the chromosome in the worst condition. The increased fitness value makes the chromosomes with lower fitness values have chance to compete with others in some exceptional conditions. A linear mapping function with scaling is such a transforming function that the population with an average evaluation value can be mapped into the one with an average fitness value. The two properties amplify the difference between good populations and bad ones so that the survival chromosome could evolve more easily.

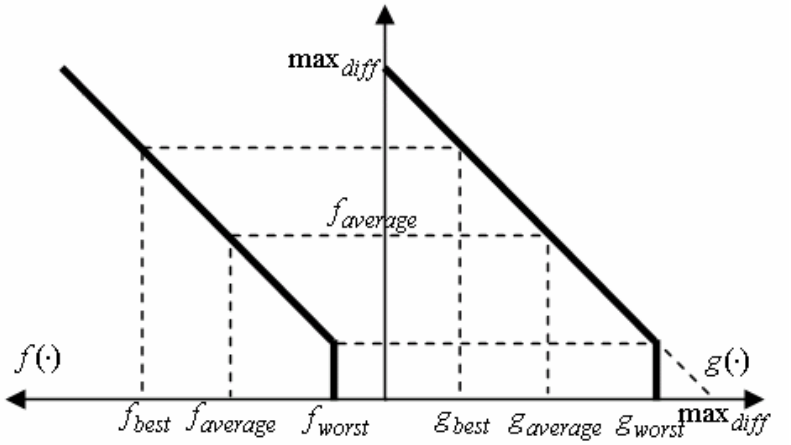


Fig. 3.4.3_1 Mapping function from the cost function *g(.)* to the fitness function *f*(.).

Beside of the transformation from the evaluation function to the fitness function,

some other necessary elements in GAs have to be considered as well. To be more systemized, the steps in the GA used in search of CNN's templates are arranged below.

(a) *Encoding process*: For the facility of using crossover and mutation, BGA (binary genetic algorithm) is frequently utilized in GA's realization. Representing a floating decimal template in terms of a binary string is necessary. In fact, CNN's hardware implementation would be the terminal goal in trying finding an appropriate template, so the values of parameters in that template need to be ranged from -5 to 5. There exist a few popular coding methods like standard coding, enhanced coding, inverse reordering, etc. Enhanced standard coding has improved the corresponding association in the standard coding methods. It is the right coding method to put the relevant digits corresponding to the same bit-location together for the effect that the familiarity in a binary string can be effortlessly found out without interferences. This encoding process of chromosomes is illustrated in Fig. 3.4.3_2. For the sake of explanations, a simplified template with most entries filled by zero would be exhibited in our encoding process. The resolution used in our template optimization need not be too high since the redundant digits in CNN increase the computing loadings only. We take the binary strings in 12 bits for convenience to represent the decimal digits in CNN's templates with the resolution less than 0.01; e.g., a binary string "111111111111" represents a floating decimal 5.00 in the template. But for our optimization process in the experiments, 5-7 bits for each weight in one population would be quite adequate and efficient to realize the encoding progress, which is ought to be accompanied by optimizing 5-10 weights within each of iterations in order to speed up the convergence in GA's evolution. In this manner, CNN's template decision could be well realized by the binary conversion in the first period of BGA.

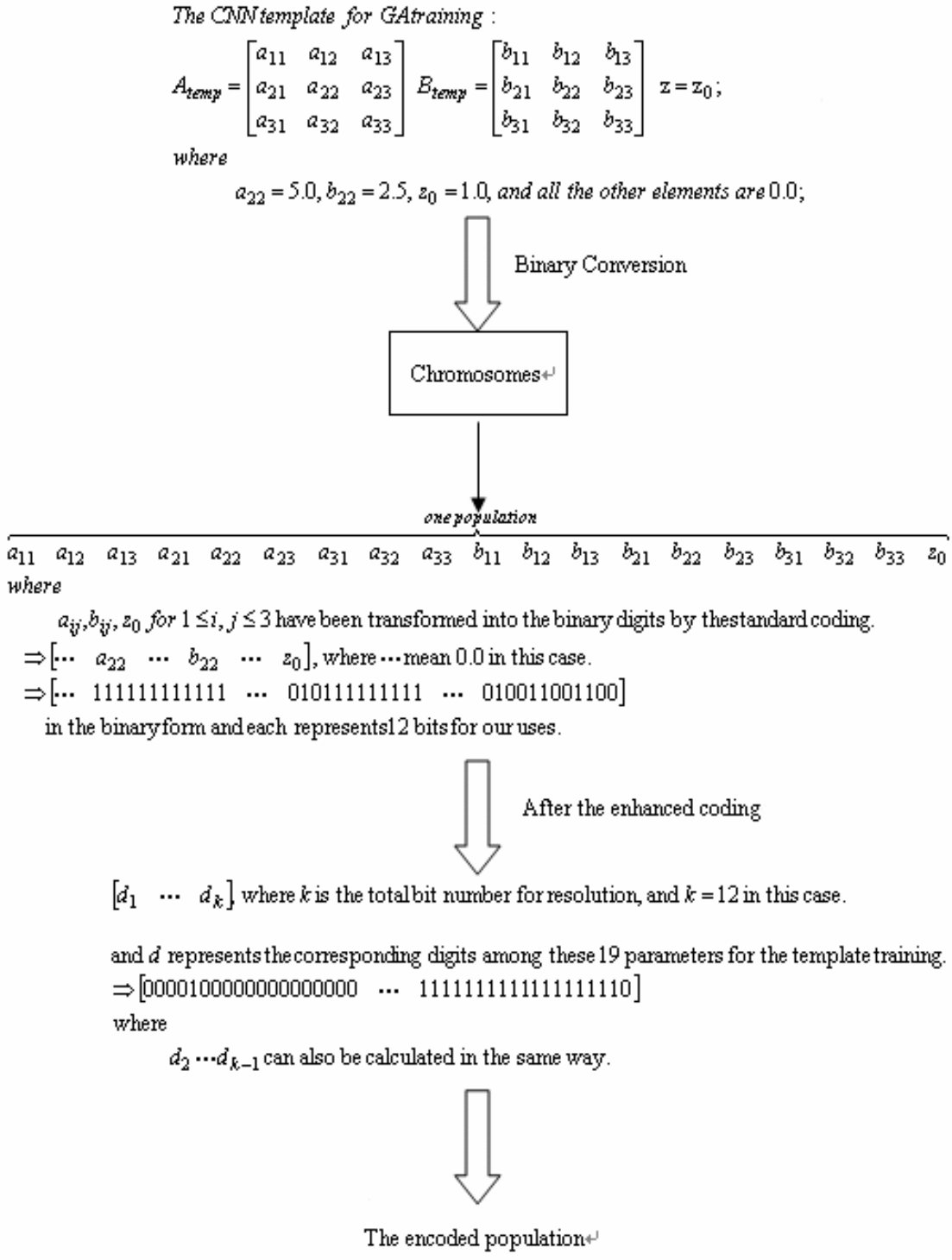


Fig. 3.4.3_2 The encoding process of chromosomes in the GA-CNN training phase.

(b) *Reproduction or selection*: In GAs, how to choose the chromosomes with higher fitness values from the candidate population is quite a decisive problem, since it influences the survival or decay of a competitive population. At this time, we

select the simplest and the most general selection method, Roulette wheel selection, to deal with this problem. At the point of adaptation functions, a higher fitness values will contribute to a bigger region area. A more contributed chromosome also would be liable to be picked out in the process of selection. The insufficient chromosomes will be generated by the recombination of the chosen populations through reproduction, crossover, or mutation operations mentioned below.

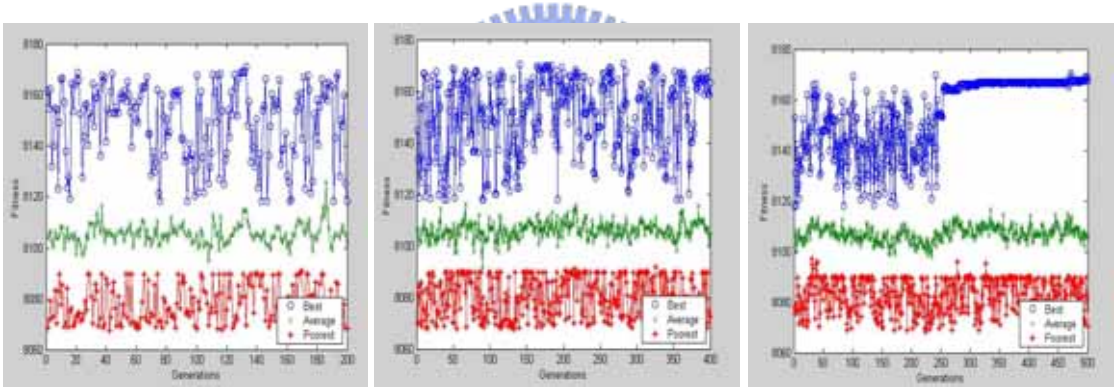
(c) *Crossover and Mutation*: Among various crossover operators, two-point crossover is adopted to alter the previous relationship between two crossing sites in the binary strings and its percentage value is chosen as 0.7 empirically. It signifies that all the bits between two different sites chosen randomly have to be exchanged. Two-point crossover ameliorates the drawbacks of one-point crossover with inability of linking certain schemas. Mutation, in the end, is just performed by flipping an arbitrary digit to increase the occurrence probability of the populations with low fitness values and varies from 5%-15% according to the present error in the fitness function. Besides, GAP is also set to be 0.65 for keeping track of the optimized chromosomes and making the vibration in GA's training slighter. Through these operators, CNN's template learning can be carried out by GA more successfully.

After the above GA optimization search, the designed CNN's templates can be rearranged as a triplet {A, B, z} and are listed in decimals in Fig. 3.4.3_3. On the GA training pass, we shall demonstrate the evolution processes by the curve of fitness functions (Fig. 3.4.3_4 (a)) and the corresponding simulated results through GA-CNN (Fig. 3.4.3_4 (c)). In Fig.3.4.3_4 (a), the fitness values of the best, average, and poorest populations are indicated for each generation in an amount of five hundred generations. Likewise, five consistent results simulated through GA-CNN for every hundred generations are also displayed in Fig. 3.4.3_4(c). It can be observed apparently from the distribution of fitness functions (in Fig.3.4.3_4 (a)) that the

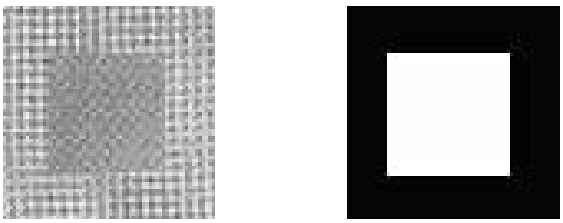
fitness values incline towards a steady state after an appropriate number of evolved generations. Also, in Fig. 3.4.3_5, the convergence curve reveals the variation in the mean squared errors between the desired output and the output acquired by GA-CNN. The distribution of fitness values and convergence curve both show the expected generation which results in the satisfactory classification consequences in experiments.

$$A=\begin{bmatrix} 0.81 & 0.78 & 3.63 \\ 2.04 & -4.36 & 2.75 \\ -1.32 & -2.42 & -1.01 \end{bmatrix} \quad B=\begin{bmatrix} 0.19 & -1.23 & 2.31 \\ -1.89 & 3.66 & -2.84 \\ 2.82 & 2.31 & 3.66 \end{bmatrix} \quad z=3.72$$

Fig. 3.4.3_3 The GA designed CNN's templates for screening textures classification.



(a)



(b)

(c)

Fig. 3.4.3_4 (a) The variation of the fitness values during the evolutions of GAs by 200, 400, and 500 generations, respectively. (From left to right) (b) The testing screening pattern for texture classification in gray scales and its binary desired output. (c) The simulated results after texture classification during the evolutions of GAs by 100, 200, 300, 400, and 500 generations, respectively. (From left to right)

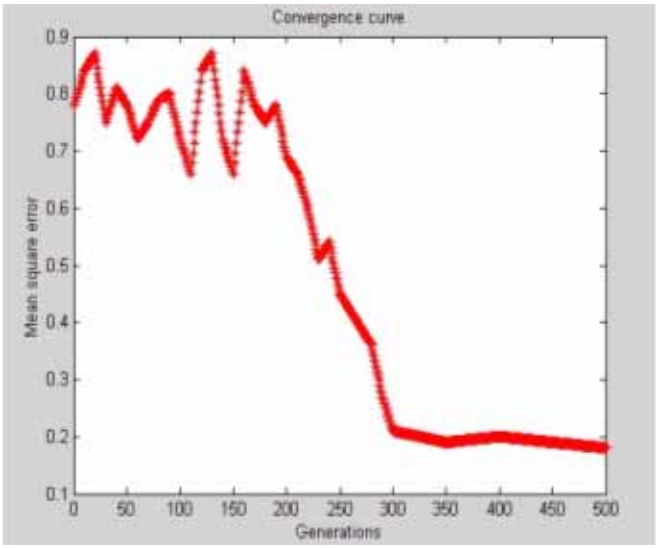


Fig. 3.4.3_5 The convergence curve for mean squared errors of the fitness function.

3.5 Selection of Descreening Filters

Numerous filtering approaches for image descreening have been proposed previously since it was believed that there must be a way to find out a qualified filter for screenings removal. But unfortunately, the outcomes of this concept only provoked over-blurred images or low descreening effect. In spite of the complex procedures, the descreening results are still restricted even if sharpening after descreening has been employed. At the same time, most of the existing literature also gives defense to their proposed filters against the common descreening filters like

Gaussian or median filters. As a matter of fact, these popular filters still work quite well on condition that the suitable arguments are correctly decided. In this chapter, we shall show that the median and Gaussian filters are capable of coping with nearly all the screened images if the screening patterns are classified and the appropriate filter arguments are determined.

The Gaussian filter and median filter have complementary effects on image descreening. Although the Gaussian filter is often under the opinion that it has less effect on screenings removal, it can remove screenings without destroying the information of high frequencies in the original documental images if the screening degree would not be too high. Contrarily, the median filter has a stronger aptitude for screening removal yet resulting in an over descreening phenomenon; i.e., over blurredness. When the screening degree is high enough, the median filter will have a great efficacy on image descreening. Therefore, the correct selection among these two filters according to different screening textures can solve almost all the image screening problems. Moreover, with the fittest filter arguments chosen in the filters adaptively, the descreening outcomes can be further improved. The adaptive argument decision mechanism will be described in the next section.

According to the above discussion, as soon as the screening type in the screened image has been testified, either Gaussian or median filter will be used. The selection mechanism depends on the tendency strength of the screening type in the screened image. This selection does not indicate the exact screening pattern but an implication about which of the two filters can offer a better descreening consequence. The information of the binary output from the CNN texture classifier is sufficient to determine which filter has to be selected. To extract such information, a determinative parameter r_s is defined as (14).

$$r_s = a(s) \Big/ b(s) \tag{14}$$

where $a(s)$ and $b(s)$ both are the number of pixels with respect to the two screening types, respectively. The parameter r_s is the selection ratio for recommending an agreeable filter for image descreening. The ratio is prone to approach zero or infinity in an ideal condition if some screening pattern is more dominant. In reality, the ratio is centered on the ranges by two boundary values, 10 and 0.1. It means that this parameter is ranged in two intervals, [0, 0.1] and $[10, \infty]$. And, it can be observed that these two values are distinct enough to halve the defined screening patterns.

3.6 Adaptive Determination of Arguments in the Chosen Descreening Filter

Even if a desirable filter for descreening has been selected correctly, an arbitrary decision of arguments in this filter might give rise to inferior descreening results. An advisable filter with the adaptive decision of arguments can just straighten out the problems in discussions. Two simple but applicable screening estimates are introduced here to provide the screened images with a well defined Gaussian or median filter. The two screening estimates are obtained by CNN's binary output as defined below. First, the CNN is performed to classify the screened image according to the extracted screening pattern and give a binary output for texture classification. Next, both x and y projection functions can be sketched or analyzed with regard to the digits of the dominant screening type. Fig. 3.6_1 displays an x projection function of the binary output from the chosen screening pattern. The y projection function of the same screening pattern can be plotted in the same way as Fig. 3.6_1. The two screening estimates can then be calculated as (15).

$$s_1 = \frac{n_{\max} - n_{\min}}{x_{\max} - x_{\min}}, \quad s_2 = \frac{n_{\max} - n_{\min}}{y_{\max} - y_{\min}} \tag{15}$$

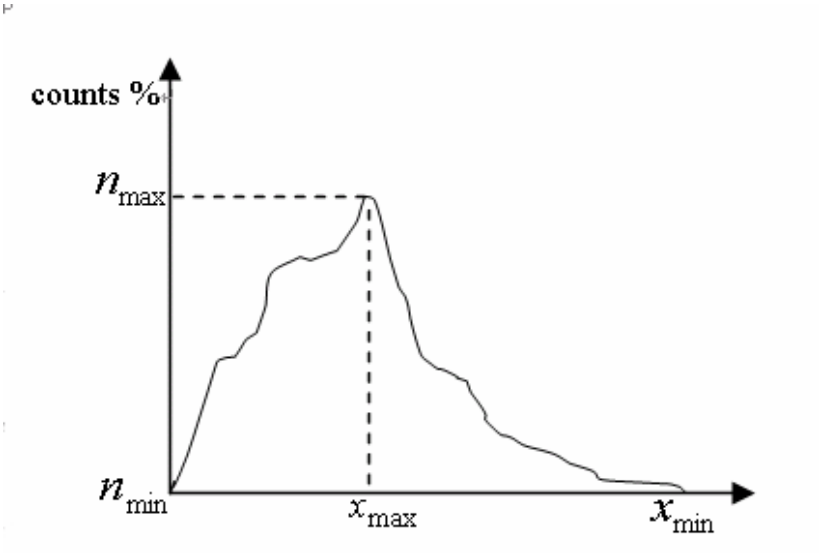


Fig. 3.6_1 An *x* projection function of the screening pattern

The CNN's binary output stands for the distribution of the two screening patterns and meanwhile reveals the screening degree in the original screened image. The above estimates point out the maximum variation in x and y directions, respectively; in other words, the distribution of screenings in that pattern could be also best described in two different directions. The inessential noises can be evitable since only the dominant screening type is considered. Also, with the concern in image descreening, the range of arguments in the Gaussian or median filter should be determined in the first place. For the Gaussian filter, we only adjust the variance in a Gaussian function from 1.5 to 9.0 with the precision up to 0.1. For the median filter, the size of the sliding block is the only argument value we have to determine in our scheme, which ranges from 3 to 15 in the width with the interval of one pixel. From any binary output obtained by the screening classification, both of the two estimates can be calculated. Naturally, these two estimates have to be normalized into its proper range in accordance with the selected filter. This working interval was encircled by the maximum and minimum estimates computed from the screening patterns in the

training phase. The maximal estimate is assigned to 9.0 as the minimal one is assigned to 1.5, and then the variance in Gaussian filter can be determined by interpolation. For the median filter, the width of sliding blocks can also be received in the same measure. Through the similar way in the training phase, we can inspect which direction is more decisive. In fact, we just emphasize the higher screening estimate in two directions now that the distribution of screenings in this direction will be evidently perceived.

3.7 Experimental Results

The experimental results reported in this section will focus on the final results after descreening. In our approach, all the experiments can be separated into two parts: the training and testing phases. The training phase includes the collection of two different types of screening patterns, CNN's template design, GA's optimization, and the acquirement in the available information of two screening estimates. The testing phase comprises the classification of screening types, the texture classification based on CNNs, filter selection, and the adaptive decision of arguments in the chosen filter. The most time-consuming step in our method is GA's optimization that determines CNN's template for texture classification. In the testing phase, fortunately, we only use the fixed CNN's templates, which have been optimized by GAs in the training phase. Actually, it takes almost no time in the stage of the screenings classification because only a small block (64*64 pixel^2) in gray scales is extracted for further analysis. As a whole, we spent about 3.3 or 0.8 seconds (by the computer, CPU: PIII, 800MHz) dealing with a document image in the size of 862 pixels by 768 pixels if a median or Gaussian filter is selected as the descreening filter, respectively. The detailed experimental results in these two phases will be given in the following two subsections.

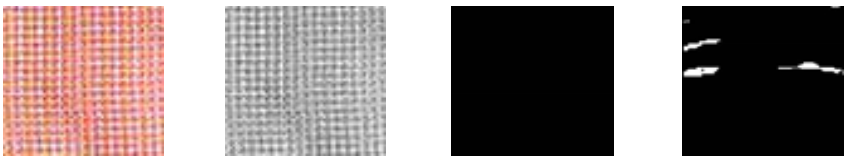
3.7.1 The Training Phase

The experimental results of using the GA for CNN's template design can be exhibited in the form of screenings classification. Fig. 3.7.1_1 to Fig. 3.7.1_3 show variously assorted consequences by means of CNN's texture classification based on GAs. The screening patterns in Fig. 3.7.1_1 and Fig. 3.7.1_2 are the ones from two defined distinct screenings. In Fig. 3.7.1_3, the screening is generated synthetically by mixing these predefined screening types. Fig. 3.7.1_4 presents the way to calculate the smooth indices of the screening pattern by the manual cropping in the training phase. Table 3.7.1_1 lists the data of the smooth indices with respect to two predefined screening types and a mixed screening pattern of those two screening types.

As Fig. 3.7.1_4 depicts, the screening pattern can be extracted accordingly. When the first screening is randomly selected, the corresponding smooth indices can be calculated simultaneously. If more than three indices are larger than 20% in those nine indices, the next screening pattern has to be randomly searched until that condition is satisfied. In this case, all the smooth indices of the third chosen screening pattern will meet this requirement. Consequently, this searching procedure can be terminated and the selected screening pattern can be applied for the screenings classification in the screened images.

As Table 3.7.1_1 indicates, the screening patterns (a) and (b) both have smaller index values than the screening pattern (c). It is obvious that smaller index values will bring about smoother blocks of images, which is able to provide an index to distinguish the smoothness of an image block. In comparison with what the previous works have been doing by using their features which were applied to texture classification and segmentation, the statistical arguments for screening pattern classification before image descreening that we proposed in this chapter have shown a better classification

performance and a lower misclassification rate by Table 3.7.1_2 in 200 documental samples with various screenings. Table 3.7.1_2 describes the classification error by several different indices, where one of them is the estimate that we made use of here, and the others are the common features like ROB and texture energy dealt with by half-toning or gray-scale outputs. It is easy to observe that the two estimates we proposed for image descreening indeed have more specific and distinguished characteristics toward the classification of screening patterns than what have been appeared in the past studies for aiming at the texture analysis only.

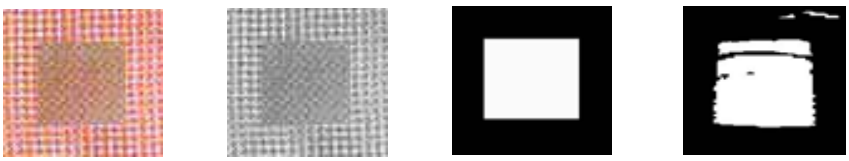


(a) (b) (c) (d)

Fig. 3.7.1_1 One defined screening type. (a) The original screening pattern. (b) The screening pattern in gray scales. (c) The desired classification result. (d) Our experimental classification result.

(a) (b) (c) (d)

Fig. 3.7.1_2 The other defined screening type. (a) The original screening pattern. (b) The screening pattern in gray scales. (c) The desired classification result. (d) Our experimental classification result.



(a) (b) (c) (d)

Fig. 3.7.1_3 The mixed screening type. (a) The original screening pattern. (b) The screening pattern in gray scales. (c) The desired classification result. (d) Our experimental classification result.

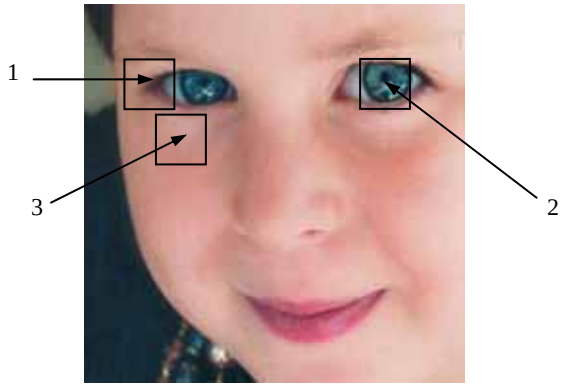


Fig. 3.7.1_4 An illustrative image for the extraction of screening patterns by smooth indices.

	Index1	Index2	Index3	Index4	Index5	Index6	Index7	Index8	Index9
(a)	8.1%	7.8%	6.9%	15.1%	5.1%	17.2%	6.6%	13.5%	7.2%
(b)	13.2%	5.4%	6.6%	8.5%	9.2%	9.6%	15.8%	11.2%	9.4%
(c)	11.5%	78.6%	27.6%	54.3%	43.2%	45.6%	33.1%	9.4%	19.2%

Table 3.7.1_1 Comparison of the smooth indices with respect to three screening patterns in distinct smoothness, (a), (b), and (c) accordingly.

Parameters	Output	Histogram		Average distributions of		Classification error
		Screening 1	Screening 2	Screening 1	Screening 2	
Our indices	B	1%	99%	0.6	75	2.4%
ROB	G	47%	53%	44%	49%	4.8%
TE	G	48%	52%	31%	33%	8.7%

Table 3.7.1_2 Comparison of the classification error for screening patterns by ROB, TE (or gray level average), and our introduced parameters (one determinative index and two screening estimates) in terms of different output formats (Binary/Gray scale).

3.7.2 The Testing (Descreening) Phase

In this subsection, we demonstrate some descreening outcomes of the proposed trained system on the screened images in the testing phase. The document images in Fig. 3.7.2_1 and Fig. 3.7.2_2 are processed by the Gaussian filter according to the screening classification result. The document image in Fig. 3.7.2_3 is processed by the selected median filter obtained in the same way. The Gaussian filter applied to Fig. 3.7.2_1 and Fig. 3.7.2_2 is facilitated with its adaptive argument decision, so the variance in the Gaussian function is set to be 2.4 for both of these two cases by the two introduced screening estimates. However, in Fig. 3.7.2_3, the screening type verifies the use of the median filter with the size of the sliding block by 6 pixels in width. We here make some comparisons to other traditional and famous methods in Fig. 3.7.2_4. Fig. 3.7.2_4 demonstrates the descreened images in various methods where (a) shows the descreened results based on our method while the others are based on three traditional methods. Among these three methods, one is the filtering approach based on wavelet analysis, another is the approach by Gaussian filtering only, and the other is the approach by medium filtering only. Apparently, our proposed method shows a better processed result even in the software-oriented exposition. No matter which filter is selected to do the image descreening by our proposed approach, the document images after processing are undoubtedly superior to those in the related studies. Again, this thesis additionally offers a chance to carry out in the design of analog circuit. CNN naturally plays a promising part in image documentation, and cannot be ignored in the field of applications of digital images.

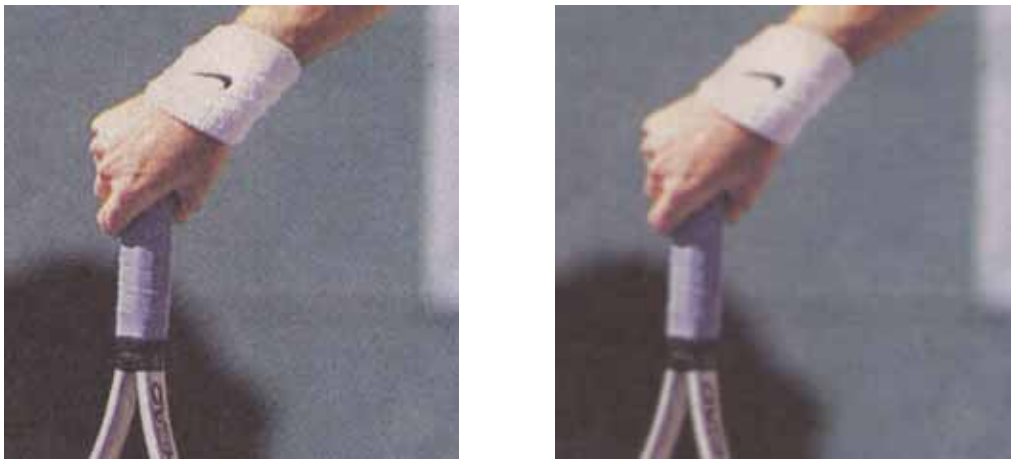
Since there is no standard method for comparison in image quality after image descreening processes, we devise an experimental scheme in the aspect of psychology by compiling and integrating the opinions of around 100 persons to indicate the acceptable degrees of the descreened images for every screening extent in various respects like descreening results, smooth, or sharp degrees at the point of human

perception. In Table 3.7.2_1, the evaluation range is halved by 5 where the interval below 5 has a higher tendency toward discredits and the interval above 5 inclines a higher confidence by credits. In the same way, a finer division by three intervals shows such different levels as high, medium, and low. All data in Table 3.7.2_1 represent every tester's point of view about the unprocessed and descreened images in an average scale. And Table 3.7.2_1 gives a kind of representative indices, perceived from human eyes, which clearly demonstrates the higher descreening performance processed by our method than that handled by some previous approaches which were extensively used in image descreening. From the experimental data in the descreening results, our method is actually superior to any other one in every respect of image qualities including smoothness, fineness, and edge information after descreening processes.

It is also proved that a simple filter still can work as long as the screening type could be classified first and an appropriate lowpass filter is chosen. In spite of the variety in the screening types, the proposed filter selection scheme can settle most of the descreening problems.

(a) (b)

Fig. 3.7.2_1 A testing image for descreening using the Gaussian filter. (a) The original image. (b) The descreened image.

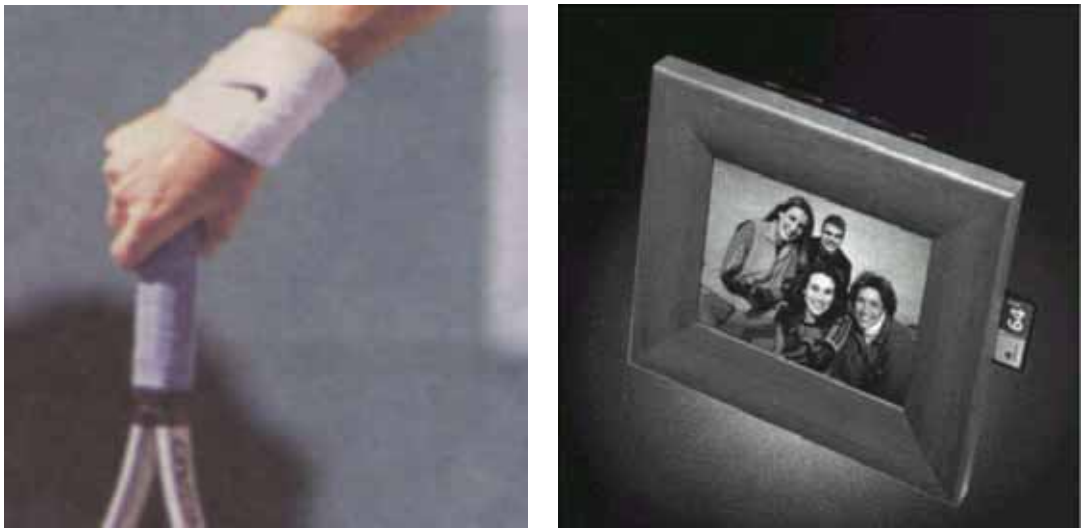


(a) (b)

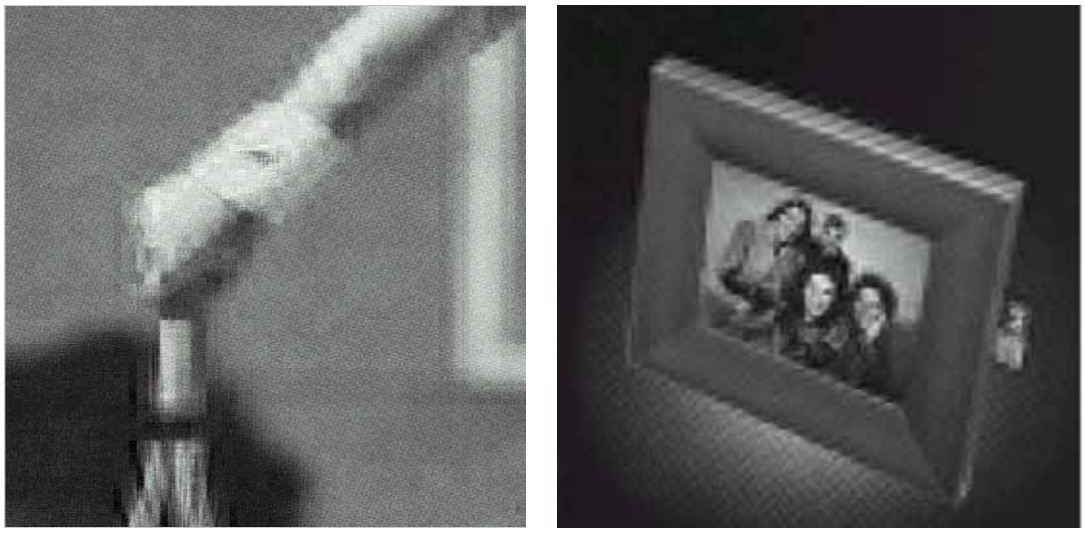
Fig. 3.7.2_2 A testing image for descreening using the Gaussian filter. (a) The original image. (b) The descreened image.

(a) (b)

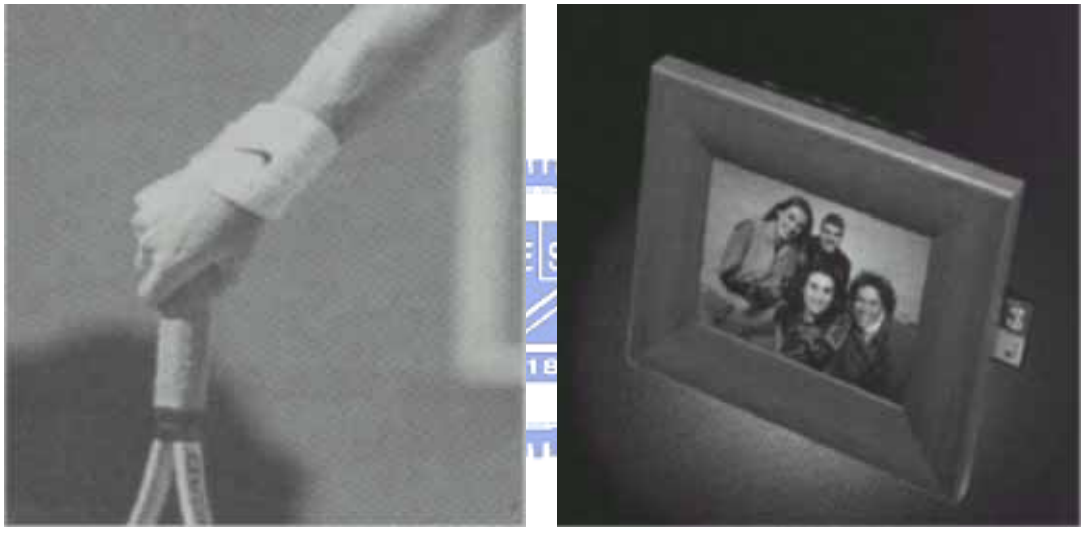
Fig. 3.7.2_3 A testing image for descreening using the median filter. (a) The original image. (b) The descreened image.



(a)



(b)



(c)

(d)

Fig. 3.7.2_4 The comparison to other famous methods. (a) The descreened images by our approach. (b) The descreened images by wavelet filtering method. (c) The descreened images by Gaussian filtering method. (d) The descreened images by Medium filtering method. (The images in each row represent various processed images in our database)

Screenings \ Images		Screening texture patterns		Original document images
		Screening 1	Screening 2	
Screening degrees	Granulation	8	3	7
	Regularity	7	5	6
	Coarseness	4	6	5
	Overall seriousness	9	7	8

De-screening \ Methods		Without classification of screenings				With classification of screenings
		Median filter	Gaussian filter	Specific Fixed filter	Wavelet transform	Our method
Descreening performance	Fineness	5	6	6	7	9
	Sharpness	3	7	4	7	7
	Smoothness	8	5	5	6	9
	Overall performance	6	5	5	7	9

Table 3.7.2_1 Comparison of descreening performance and screening extent in human's observation by credits and discredits halved in 5 (normally we have three different ranges: low for degree 1-3, medium for degree 4-7, high for degree 8-10).

3.8 Concluding Remarks

The work in this chapter was carried through with a creative idea in image descreening. An alternative way applied to the screening classification is the crucial part in making a correct choice of descreening filters for the removal of screenings. By the applications of CNN's texture classification, the predefined screening patterns can be successfully separated. By the uses of GAs, the decisive template in the functions of CNNs can also be well optimized. The classification based on CNNs can be accomplished in such an easier manner that we make the descreening results more acceptable. Also, we introduced the smooth indices to determine whether the

screening pattern in the testing image was smooth enough to be extracted. The defined screening types can be verified for the later filter selection according to this screening pattern. CNNs in this chapter played an influential part due to its stability and the association of neighboring cell linking. In particular, two suggested screening estimates in the training phase were taken to decide the range of the same ones in the testing phase, which helped to determine the arguments in the selected filter. In this adaptive way, an appropriate filter with the chosen arguments not only made the descreening processing easier but also rendered the processed images attractive ones to human perception.

Our experimental results indicated that the descreened document images by our scheme are more natural and acceptable to naked eyes, for both the high and low frequency part in a descreened image demonstrating the smoothness and sharpness in concord. With the selection of a proper filter and the decision of its arguments, the screenings have been removed away effectively, but the high frequency part in document images would still remain natural. It was shown that the descreening results were also pleasing or even better despite of the absence of sharpening processes in our approach. For more application in this field, the binary output of CNNs will be extended to the output in gray values or even in color scales with more channels.

4. Texture Discrimination based on GA-CNN proliferation structure

4.1 Introduction

In this chapter, we presents a novel approach to texture discrimination, realized by an innovative feature curve, TBS (Transition Bit String), extracted from the mapping of Cellular Neural Networks (CNN's) with optimized GA-CNN templates. This one dimensional feature curve TBS has not only been demonstrated valid and effective in the representation of various textures, it also leads to a better classification result for complicated texture patterns. What is more, TBS helps to carry out the GA-CNN based proliferation structure that always optimizes the number of templates for texture classification and representation. Also, the overview of diverse texture patterns from CNN's characterizes the use of CNN's and determines the discriminability between textures. With the new approach we're introducing here, we would be able to proliferate the CNN templates at a much wider flexibility simply by analyzing the feature curves of TBS for different textures. Meanwhile, we also provide a selective processing mechanism for distinct cases of textures in classification analysis, which takes into consideration the given conditions of texture patterns in question. Last but not least, since the proposed structure and the developed feature curve for texture classification show a satisfactory result even without given conditions about texture patterns and is still applicable in much more complicated texture patterns, we would be able to show the advantages of TBS in analyzing the representation of textures whereas presents the superiority it carries out in the analysis of texture classification.

This work is however significant since texture analysis could be applied to

numerous researches such like image documentation, image descreening, background removal, and many other digital image processing issues [67]. As to texture classification, GA would definitely be employed in the design of templates of CNN's. Nevertheless, most studies on texture classification based upon CNN's have put an excessive emphasis on the searching of templates in CNN's without really considering the wider varieties in texture patterns that make the problems even more complicated because the training of optimized templates is supposed to be time-consuming and inefficient. This chapter, therefore, aims at building a brand-new structure to effectively find the minimum number of CNN templates while retaining texture classification performance. To illustrate our approach more clearly, we have organized this chapter with the following sections: the proposed system architecture, Transition bit string (TBS), the characteristics of texture patterns from CNN's, design of characteristic templates optimized by genetic algorithms (GA's), texture classification mechanism, experimental results, and concluding remarks.

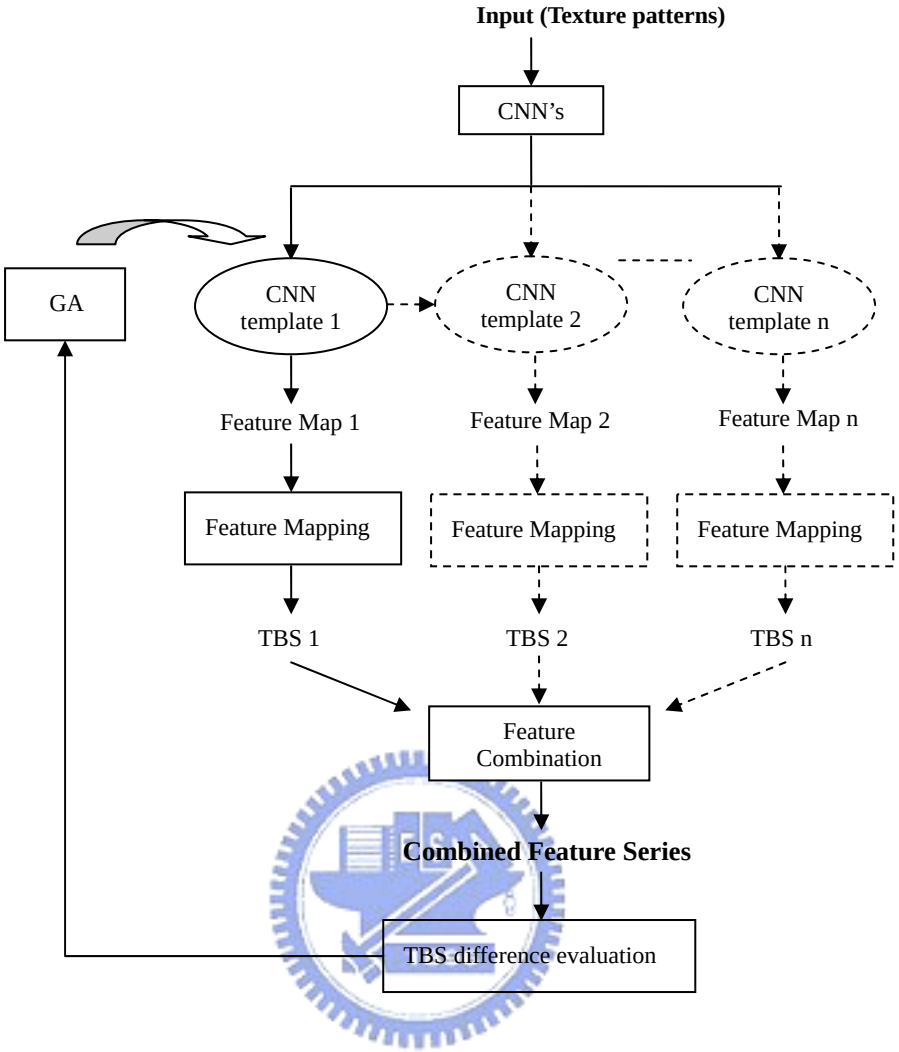
4.2 The Proposed System Architecture

A novel systematic architecture is proposed in this chapter, as shown in Fig. 4.2_1. This overall architecture describes the complete workflows for our classification problem and lists the main components under this structure. As we discussed earlier, CNN's plays a crucial role in overlooking the characteristics of various texture patterns in terms of optimized GA-CNN templates. The proposed structure here could process two distinct texture patterns defined by various ways to acquire training samples. By obtaining the feature maps from the CNN outputs, we use some special method for feature extraction to derive the curve of TBS. TBS then becomes the kernel part here since it could be taken as the tool for the representation of different textures and for the judgment foundation on differentiating different

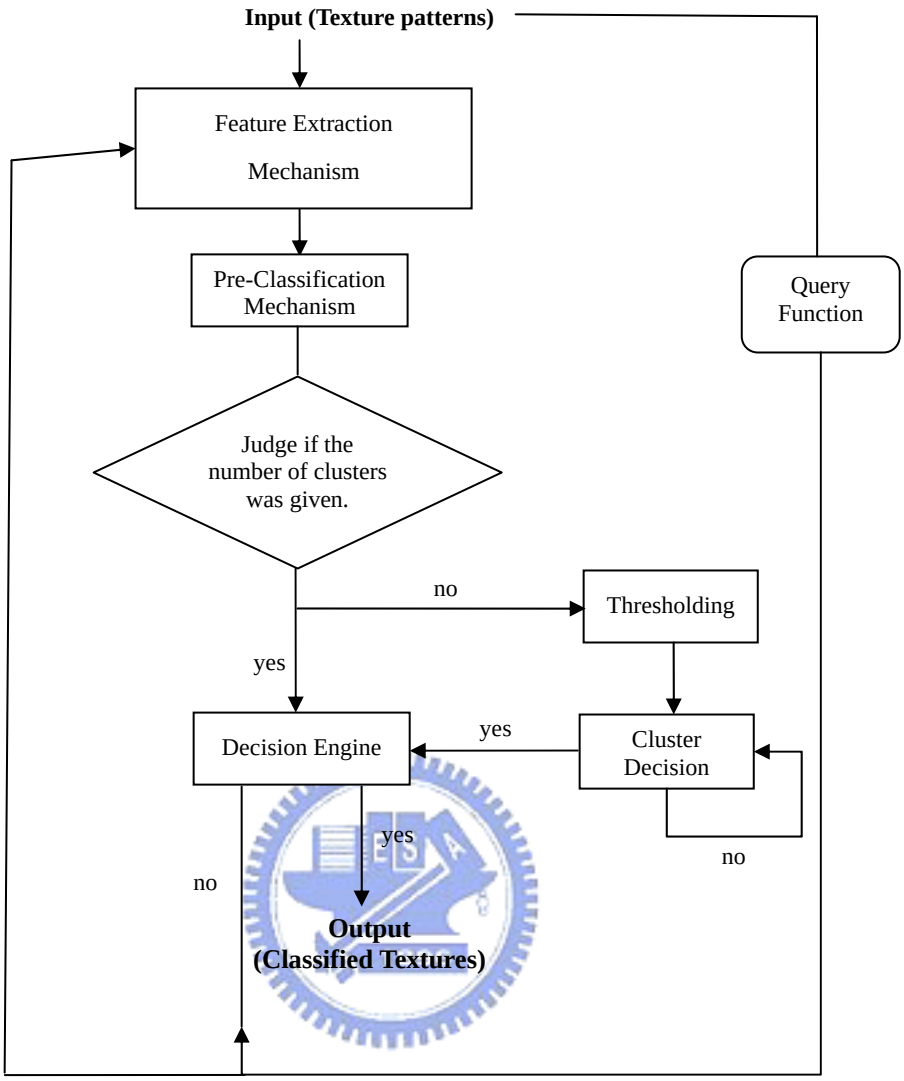
texture patterns. Finally, we apply TBS instead of the original image of texture patterns to the input of classifier for texture classification and use it to represent the texture patterns.

To explain more clearly, the feature combination in Fig. 4.2_1 (a) elaborates the proliferating process of CNN templates based on the decision and evaluation of TBS. The query function on the righthand side of Fig. 4.2_1 (b) provides a tradeoff choice for something that we might not have about the input of the classification issue. In order to avoid the unnecessary interferences and prepare a better processing strategy, the pre-classification mechanism has been put here to simplify the classification procedure. After classifying the combined feature curves for the first time, we would move on to the following decision engine if the number of clusters has been given. We would not take the other branch as indicated in Fig. 4.2_1 (b). The other branch includes thresholding and cluster decision, where thresholding helps remove any redundant information in the feature curve obtained from feature mappings and cluster decision makes the features extracted from the output of various CNN's classified in advance and decides the number of clusters. On the other hand, if the number of clusters has been determined on cluster decision, the same procedures would be executed as in the case of *a priori* clusters.

The classification algorithm has to keep running till the number of clusters is determined. And decision engine judges if the difference between any two clusters is greater than some threshold. If so, the classification process would be terminated and the outputs are the classified texture patterns. If not, the classification process would go on and one more CNN template would be proliferated to generate more TBS features in representing distinct textures. Our experimental results eventually demonstrate both the capability of our introduced feature curve and the pleasing classification outcome based on this method.


Input (Texture patterns)
CNN's
GA
CNN template 1
CNN template 2
CNN template n
Feature Map 1
Feature Map 2
Feature Map n
Feature Mapping
Feature Mapping
Feature Mapping
TBS 1
TBS 2
TBS n
Feature Combination
Combined Feature Series
TBS difference evaluation

(a)



(b)

Fig. 4.2_1 The GA-CNN based proliferating system (a) Feature extraction phase (b) The recognition phase

4.3 Transition Bit String (TBS)

For any applications in digital image processing, we offer a very useful feature index extracted from two-dimensional signals, making it the desired descriptions of texture patterns. This defined function TBS, can be taken as a projection mapping from a 2-d input signal to a 1-d output signal. For any different kinds of combinations of feature series, the characteristic function TBS could be defined as (16-1) or (16-2)

depending on the type of input signals. Equation (16) can also be expressed in terms of (17) and (18).

$$TBS = \sum_k h_k(x^k) \qquad for\ 1 \le k \le n_f \tag{16-1}$$

or

$$TBS = \bigcup_k h_k(x^k) \qquad for\ 1 \le k \le n_f \tag{16-2}$$

where

$$h_k(x^k) = \frac{n_k(x^k)}{n} \qquad for\ 1 \le k \le n_f \tag{17}$$

and

$$\begin{cases} n_k(x^k) = n_k(x^k) + 1, & if\ (x^k(y_\infty)^{(i)} - x^k(y_\infty)^{(i-1)}) < \varepsilon \\ x^k = x^k + 1, & otherwise \end{cases} \quad for\ 1 \le i \le n\ and\ 1 \le k \le n_f$$

…….....(18)

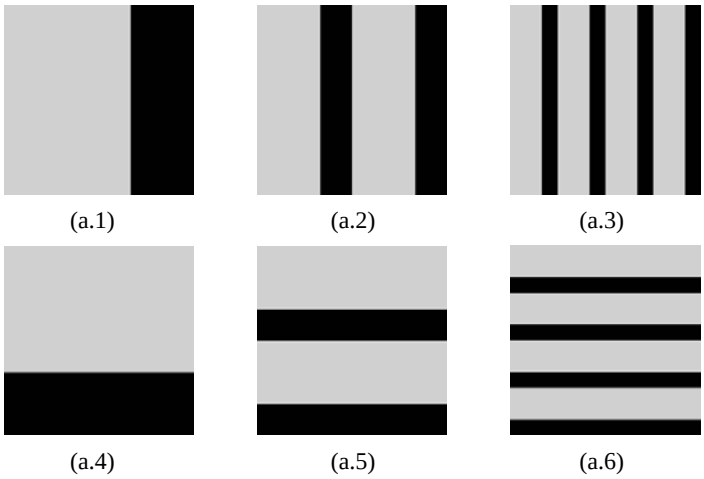
where TBS indicates the sum or union of differently-combined feature curves after scanning the original 2-d input signals in some orientation *k* over its axis x^k, while n_f is the total number of feature curves. Here we use the superscript *k* to avoid confusion with the overall domain of axis *x*. The domain of TBS should be the unions of x^k with the numerical value *k* from 1 to n_f. If the summation in (16-1) is replaced by the "OR" operation, TBS stands for the union of feature series in some sense and (16-2) will replace (16-1). Whether to select (16-1) or (16-2) is usually determined by the sensibility in orientations of texture patterns and output signals. Apparently, (16-2) would be more applicable to the applications of CNN's since we simulate the CNN outputs in a more logical form, which also keeps more information from the extracted feature curve. The derivative equation (17) kicks off the normalization of histogram by dividing the value *n*, the total number pixels in the images of CNN outputs, to keep from the disturbance of various image sizes. Also, the maximum value x^k in the

orientation k indicates the total number of continuous strings listed in the above-mentioned equations and here a string is referred to as a digital image with its corresponding pixels.

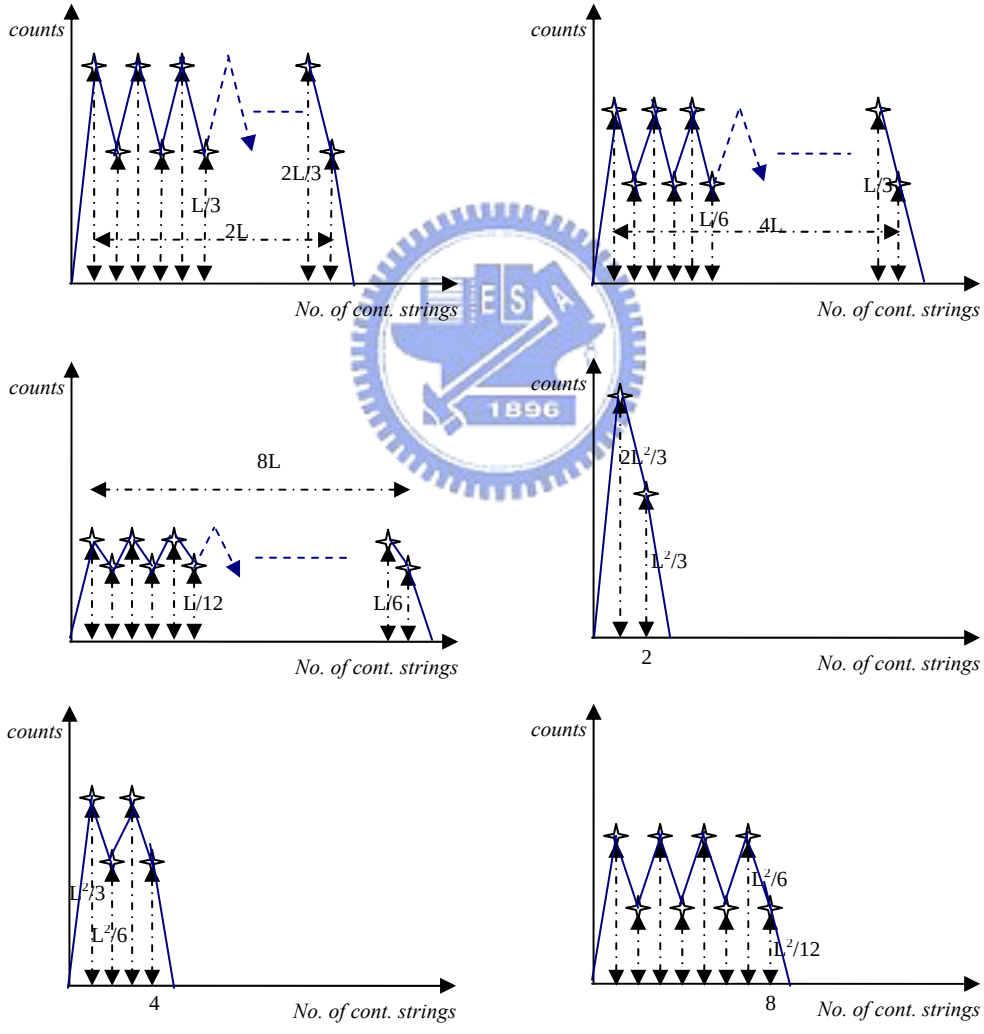
In addition, equation (18) describes the kernel part of this characteristic function where $n_k(x^k)$ represents the length of the current continuous string for the steady state signals (y_∞). And $x^k(y_\infty)^{(i)}$ means the transformed signals in orientation k from y_∞ in location i. In our application, we use the output of CNN's for this kind of input signals. The numerical value ($n_k(x^k)$) hence indicates the length of the x^kth continuous string for the kth feature curve in the orientation k. This value will be accumulated only when the difference ratio between the current string (x_i^k) and the previous one (x_{i-1}^k) is lower than some predefined threshold (ε). This threshold depends on the types of image and how we tolerate noises of the processed images. This explains the difference ratio is higher for more noisy textures. In our case, ε is set to be 1 for the CNN binary output. Our featured function regarding x^k projects the original 2-d signal (CNN outputs here) onto a 1-d curve so as to re-classify the binary strings according to our requirement. Finally, the defined characteristic function TBS can be determined after analyzing the combination of feature curves in all considered orientations. As we mentioned, the pre-defined threshold is frequently set to remove more noises while deciding the current continuous string. This featured function can be more flexible by selectively determining or adjusting this threshold.

Later in our experimental results, the feature curves in two dominant orientations (horizontal and vertical) are well-functioning enough to classify up to sixteen texture patterns in our database. This method is still applicable to the feature curves in more orientations if the texture patterns are distributed in some particular orientation. We named this feature curve TBS (Transition Bit String) since its horizontal axis

represents the times of transitions or the number of total continuous strings and the vertical axis shows the length of some specific string. To illustrate the use of TBS, we have some regular texture patterns in the ideal binary level at different frequencies and orientations as shown in Fig. 4.3_1 (a), where the proportion of whiter area in (a.1) is 2/3 distributed horizontally and each texture image is assumed in LxL size for simpler explanations. The proportion of whiter area in (a.2) is one half of the same area in (a.1), and (a.3) is again one half of (a.2). As to Fig. 4.3_1 (a.4) ~ (a.6), we have the same proportion between two different bright blocks while they are distributed vertically. The corresponding TBS in the horizontal orientation could be shown in Fig. 4.3_1 (b). We define the x-axis of Fig. 4.3_1 (b) as the number of continuous strings and y-axis as the length of each continuous string (denoted by counts). In Fig. 4.3_1, the numerical values (like 2L/3, L/3, etc…) represent the corresponding coordinate values of x-axis or y-axis on its feature curve. From the observations we make here, horizontal TBS plots indicate the same distributions for the same textured patterns regardless of frequencies and orientations; they also at the same time reveal different lengths and widths for those texture patterns. On the other hand, vertical TBS plots demonstrate the opposite conditions against the horizontal ones. In our case, our TBS combines the feature curves in two distinct orientations (horizontal and vertical) by (16-2). Hence, TBS in truth shows the capability in discriminating texture patterns at different scales or frequencies, and we could use TBS to classify and represent texture patterns of natural images as long as the images of texture patterns are spaced clearly by nature or by appropriate transformation. As a result, we may successfully incorporate the structure of CNN's in the sense of the ability in generating distinct binary outputs for digital images to obtain the unique projected feature maps with respect to various textures. The reason of introducing CNN's in this chapter will be given in the following section.



(a)



(b)

Fig. 4.3_1 Illustrations for TBS plots (a) The ideal texture patterns in various frequencies and orientations (b) The corresponding TBS plots orderly arranged from left to right, and top to bottom (corresponding to (a.1) ~ (a.6)).

4.4 The Characteristics of Texture Patterns from CNN's

When it comes to texture discrimination analysis, we all aim at finding the characteristics of each specific texture pattern. How to describe or represent the distribution in texture patterns would thus be significant and how to extract the representative features from various texture patterns would be also crucial to the final performance in the classification mechanism. As we know, textures can be viewed as complex visual patterns composed of entities, or sub-patterns, that have characteristic brightness, color, slope, size, etc. [44]. Generally speaking, four kinds of popular approaches nearly dominate all the researches in the field of texture analysis: structural, statistical, model-based, and transform methods. Structural approaches [44], [45] represent texture by well-defined primitives, indicated as micro textures, and a hierarchy of spatial arrangements, indicated as macro textures, of those primitives. As to model-based texture analysis [47] ~ [52], it uses fractal and stochastic models, and attempts to interpret an image texture by using generative image model and stochastic model, respectively. More clearly, [47] describes the natural scenes by fractal dimension estimation and takes advantage of iterations between parameters to improve the performance in one parameter adjustment. Using the set of Markov parameters and the correlation coefficients of textures [48] makes the classification problem more challenging. But the limitation of [48] is that the calculated arguments from texture patterns can not distinguish these textures since they share the same power spectra but different phase information. The approach proposed in [49] uses the Gibbs distribution for noisy textured images and it shares the parameter estimation as in [47]. For the unknown number of clusters among textures, [50] proposed an unsupervised texture segmentation method to model the textured images that can be divided into non-overlapping regions, from which the information of Gauss Markov

random field (GMRF) can be extracted. Besides, transform methods for texture analysis, such as Fourier [51], Gabor [52], [53] and wavelet transforms [54] ~ [56] represent an image in a space whose co-ordinate system has a strong understanding that is closely related to the characteristics of a texture, like scales or frequency. For instance, [53] interprets image textures with 2-D Gabor filtering by its orientation characteristics, and [54] describes the basis of image representation based on multi-frequency bands. For multi-band analysis, [55] uses wavelet package to represent textured images by entropy and energy of wavelet coefficients. Again, for the unsupervised case, a simple structure based on wavelet analysis for unknown number of textures was introduced in [56]. From the observations above, it is clear that our proposed texture analysis scheme keeps the advantages of structural methods while retaining those of statistical ones. The texture patterns can be regarded as primitives in structural methods while the specific CNN template could be viewed as the elements of primitives for reflection of some specific textures. In addition, like the analytical procedures in the statistical approach, our introduced TBS plays an essential part in describing the distribution in various textures in the aspect of feature extraction through different CNN templates. In the following section, we will have a brief introduction to the basic structure of CNN's for those who might not be familiar with CNN's. After that, a brand-new idea based on CNN's for texture discrimination will be given in section 4.4.2.

4.4.1 The Settings of CNN Templates

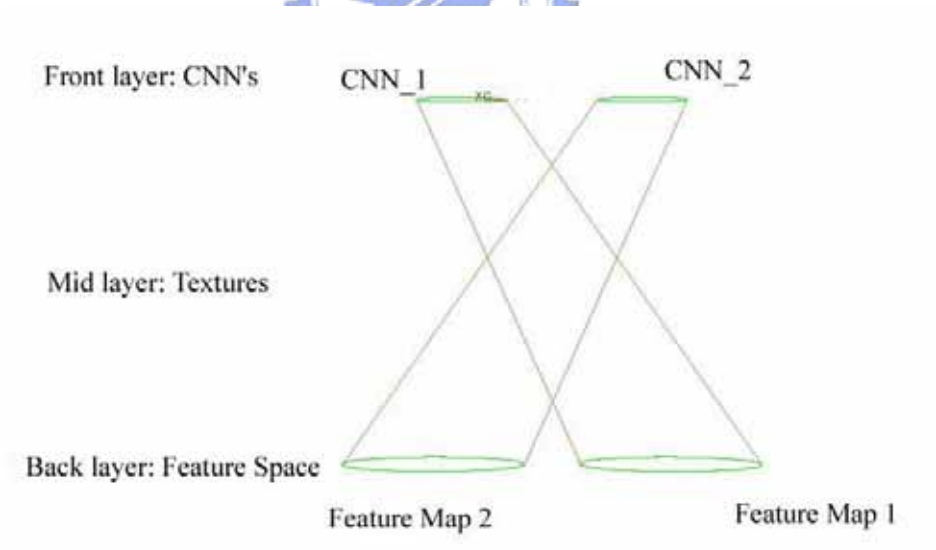
As what we have addressed in the chapter 3, the state equation and output function of CNN's for the image applications could be described as (11) and (12), respectively. Likewise, the CNN template contains the combination of a triplet {A, B, I} for the template learning, where A consists of all the arguments a_{kl}, and B represents the values b_{kl}. In the structure of CNN, A and B both are 3*3 or 5*5

matrices. The size of matrices depends on the functions of CNN. The threshold I is a one-dimensional scalar. In our system, we apply 3*3 matrices to both A and B in the consideration of functional performance for the texture classification. The constant input is chosen to be the original image; the initial state in CNN for the texture classification can also be set as the original texture pattern due to the ability of convergence. The good thing is not every element in templates has to be trained since our defined templates are sensible enough to describe texture patterns of any kind. These types of templates can represent the characteristics of textures in our innovative ideas, which will be conceptualized and illustrated as below and more details will come later in section 4.5.

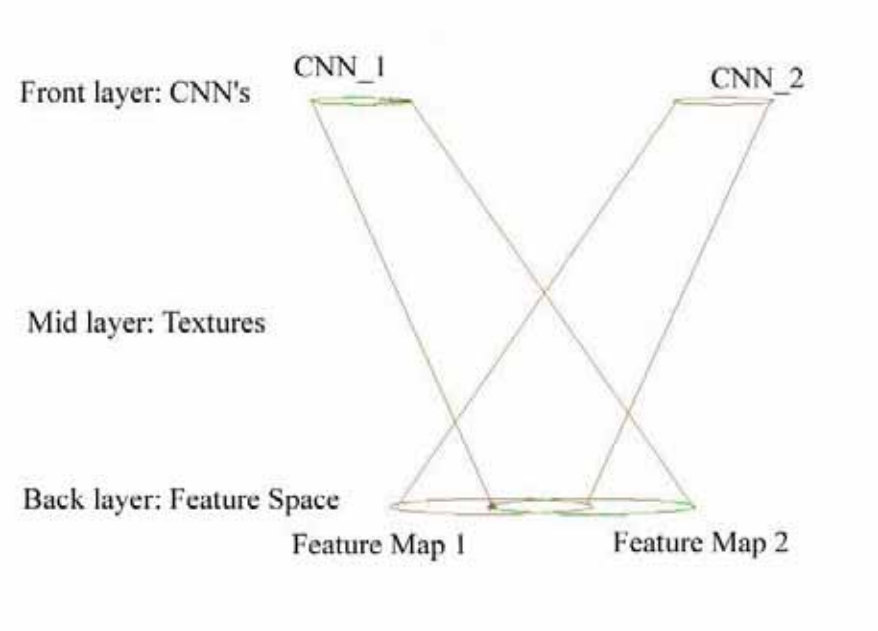
4.4.2 The Overview of Texture Patterns Based on CNN's

Unlike the traditional approaches using CNN's, we take advantage of the characteristics of templates in CNN's to simulate the distribution among texture patterns instead of looking for a specific template for some specific applications. The characteristics of different kinds of texture patterns could be generalized from CNN's mapping, as shown in Fig. 4.4.2_1 (a) and (b). One can understand why we try to describe the properties in texture patterns for classification problems from the perspective of feature mapping based on various templates of CNN's. As a result, it is similar to an outlook from various CNN templates (the front layer) to feature space (the back layer) through many kinds of texture patterns (the mid-layer). From Fig. 4.4.2_1 (a) and (b), different templates in CNN's correspond to different regions of feature space which might be projected from various texture patterns. It could also be observed that Fig. 4.4.2_1 (a) and (b) indicate two diverse conditions that might result from the singular and nonsingular feature projection based on the same CNN template. For illustrations of more details, Fig. 4.4.2_2 describes the feature space in a more definite fashion for both ways in bifurcating feature mappings seen through a

characteristic CNN template. With a suitable selection in CNN templates which can be found in advance or trained by GA's, textures would be projected by these templates into ranges of feature space different from one another. On the other hand, the more overlapped condition in the feature space projected from a specific CNN template would be undesirable in this chapter. As what has been described above, we try to find the applicable CNN templates that can best describe the texture patterns in order to represent or classify these textures in the consideration of improving the performance of our system. Naturally, CNN's could generate this kind of feature maps by optimizing templates based on GA, which would be elucidated at the design guidelines in the following section.



(a)



(b)

Fig. 4.4.2_1 Illustration Figure for Feature Mapping based on CNN's by (a) Non-overlapping condition and (b) Overlapping condition.

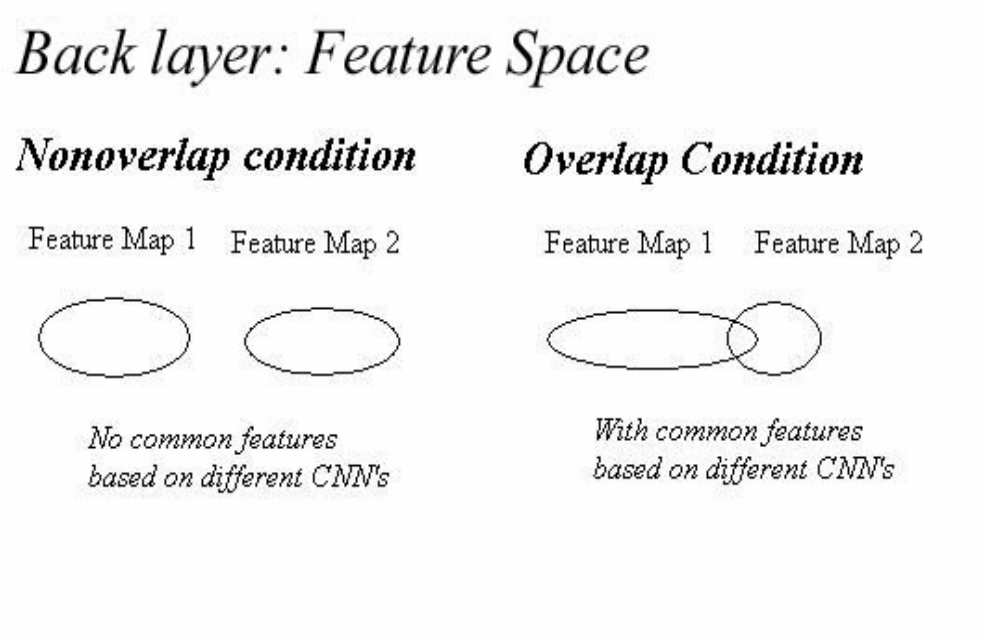


Fig. 4.4.2_2 Illustration Figure for the distribution in features projected from different CNN templates

4.5 Design of Characteristic Templates Optimized by Genetic Algorithms (GA's)

In the past studies, the most important issue in any applications related to CNN's lies in the search of CNN templates. However, looking for appropriate templates is sometimes time-consuming and difficult, which in turn results in a standard format of the system structure and the constraints of the performance of that system. Therefore, efficient template finding could be decisive and may also better off the final results. To discriminate texture patterns more clearly, we use genetic algorithms to find the specific CNN templates with the goal of generating distinct feature maps for any two different textures. The design of CNN templates has been prevailingly illuminated by GA since the convenience and robustness of GA have extended the capability of CNN's applications. A strategic approach has been proposed to provide a simple but comprehensive method for texture classification and segmentation [43]. The feature mentioned in this reference, Rob (ratio of black pixels) and average gray-levels, in fact give a very helpful tool to classify texture patterns after a series of processing units or analytic steps. But these features confine the applications in image processing by CNN's since we can not leave aside the practical problem of whether the hardware is implementable for complicated texture patterns. In this regard, we propose a one dimensional feature curve in this chapter to highlight the more complicated issues of image processing by CNN's, so that hardware of the applications alike in higher level image processing can be implemented more easily. In addition, this feature curve plays an important role in the design of CNN templates based on GA by incorporating with the evaluation of fitness function and the simplification of templates in the optimizing process of GA.

4.5.1 The Design Rules on GA by TBS

A genetic algorithm (GA) is a kind of methodology, which can accomplish the specific task or processing by a series of rigorously mathematical operations and logical decisions. With the ability of selection, crossover, mutation, and reproduction in genes, the superior offspring will survive while the inferior one being excluded through competition. When GA is used as the training tool for CNN templates, all the elements in the templates of CNN's have to be verified. As far as texture analysis is concerned, the CNN templates which possess the ability to simulate or differentiate texture patterns belong to a special kind of filter for extracting features in the original textures. Therefore, we would manage to modulate this specific form of CNN templates in stead of looking for a new CNN template just between any two specific texture patterns. The reason is simple: the latter gives rise to the decrease of the trained CNN templates and truly reflects the non-overlapped feature mappings from those templates. In fact, there exist some types of templates in generating distinct feature mappings among different textures for enhancing the differentiation functions of the classifier. Thus, the goal here turns to predefine this special form of templates to differentiate texture patterns. This kind of CNN templates could be determined and optimized by the adjustment of TBS difference when the following criterion is satisfied.

$$\frac{abs(TBS_k^i - TBS_k^j)}{\max(TBS_k^i, TBS_k^j)} \geq \eta_k \tag{19}$$

where TBS_k^i and TBS_k^j are the *k*th index of our featured curves calculated from the CNN outputs for any two different textures, texture *i* and texture *j*. The subscript *k* represents the *k*th location in the calculated TBS. The η_k for the *k*th index represents the threshold of difference ratio between texture patterns, whose value, between 0 and 1, can be selected beforehand for various sets of texture patterns. Also η_k approaching 1 implies a higher difference ratio between texture *i* and texture *j*. We set

the same η_k for all indices for convenience and a bigger η_k for higher discriminative power between textures. For this requirement, we choose the error function in GA as follows.

$$g(l) = \sum_{k=1}^{n} abs(y_k^i - y_k^j) \tag{20}$$

where *l* is the training template including A, B, and I. *n* is the number of cells that depends on the size of templates; y_k^i is the desired output for the target texture pattern and y_k^j is the steady state output of the discriminative texture pattern for the *k*th cell. Here the definition of (20) just expresses the relationship between the current optimizing templates, and the error function should be related to our TBS difference. So the error function in GA was defined as the reciprocal of TBS difference. Finding a lower difference between y_i^d and y_s is equivalent to finding a higher TBS difference. As what we intend to do in this chapter, the fitness function given above is set to be the difference of the defined feature curve, TBS, between different texture patterns. Besides, any template optimized by GA must be stable because the lower fitness values always result from unstable trajectories in CNN. It ensures the stability in CNN if a higher fitness value is reached, which is exactly why GA is used for. Since these two functions in this issue would be running after the same goal, we do not need to define an additional error function or to project it on the fitness function by any transforming function, like most approaches would do. The other related adjustments about GA on the search of CNN templates such as the encoding process, the way of adjusting reproduction and selection, crossover and mutation rate, etc, can also be referred to the previous chapter for more details.

In order to optimize CNN templates in pursuit of generating different feature maps for various texture sets more efficiently, we predefine three types of uncoupled

CNN template which take the distribution of different texture patterns into account with regard to its regularity, uniformity, and symmetry. Hence, we have

Type 1

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & a & 0\\ 0 & 0 & 0\end{bmatrix}\quad B=\begin{bmatrix}b_2 & 0 & b_2\\ 0 & b_1 & 0\\ b_2 & 0 & b_2\end{bmatrix}\quad I=c.$$

Type 2

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & a & 0\\ 0 & 0 & 0\end{bmatrix}\quad B=\begin{bmatrix}0 & b_2 & 0\\ b_2 & b_1 & b_2\\ 0 & b_2 & 0\end{bmatrix}\quad I=c.$$

Type 3

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & a & 0\\ 0 & 0 & 0\end{bmatrix}\quad B=\begin{bmatrix}b_2 & b_2 & b_2\\ b_2 & b_1 & b_2\\ b_2 & b_2 & b_2\end{bmatrix}\quad I=c.$$

All the three types of CNN templates are defined according to the arrangement of every cell coupling with its texture pattern, and only three conditions are considered due to the applicability in various texture patterns. For each training process of GA, only one of these three templates would be met to form the discriminative feature map for any pair of texture sets. Later we will show the faster convergence condition based upon our GA-defined template in the up-coming experimental results. The three types of CNN templates describe different ranges of feature space, and our featured curves could decrease the number of templates that have to be optimized. It is interesting to observe that CNN edge template in the library happens to be one of the defined templates since the former describes the texture pattern in the high frequency band. Thus it becomes the first template to represent textures before deciding whether or not the proliferation of CNN templates is necessary. Meanwhile, TBS helps provide an evaluation index in the optimizing

process of GA to determine the extent of discriminations between textures.

To acquire the feature maps based on different CNN's, we have to optimize our defined templates. And the optimized results after GA search are shown in Fig. 4.5.1_1. For any two texture patterns (Fig. 4.5.1_2 (a)) which can not be classified correctly by the first template, a new CNN template will be optimized as stated in our approach. The evolution processes is demonstrated by the corresponding feature maps (Fig. 4.5.1_2 (b)) through GA-CNN and the corresponding fitness functions (Fig. 4.5.1_2 (c)) during GA optimization. The first and second row in Fig. 4.5.1_2 (b) and (c) correspond to the left and right texture pattern in Fig. 4.5.1_2 (a), respectively. From the steady state feature maps shown in the rightmost figure of Fig. 4.5.1_2 (b), we know that the original texture patterns have been well described by the new feature maps, which show the diverse distributions in the horizontal and vertical directions for those two patterns. And in Fig. 4.5.1_2 (c), the fitness values of the best, average, and poorest populations are indicated for each generation in an amount of two hundred generations. It can be observed apparently that the fitness values incline towards a steady state after an appropriate number of evolved generations. Fig. 4.5.1_3 describes the convergence curve by the variation of error functions in the training process of GAs between our target texture patterns. We define the error function as the difference of parameters between the current and desired patterns as in (20) in order to observe the convergence condition during the evolution of optimized CNN templates based on GA. The label of x-axis in Fig. 4.5.1_3 is denoted by epochs since the generated populations based on GA should be corresponding to the training process of neural networks. Finally, the distribution of fitness values and convergence curve both show the expected generation which results in the satisfactory classification consequences in experiments.

Template 1

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 2.01 & 0\\ 0 & 0 & 0\end{bmatrix}, B=\begin{bmatrix}-0.24 & -0.24 & -0.24\\ -0.24 & 2.01 & -0.24\\ -0.24 & -0.24 & -0.24\end{bmatrix}, I=-1.5$$

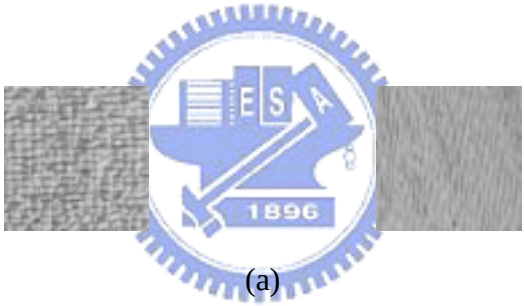
Template 2

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 0.97 & 0\\ 0 & 0 & 0\end{bmatrix}, B=\begin{bmatrix}-0.76 & -0.76 & -0.76\\ -0.76 & 4.92 & -0.76\\ -0.76 & -0.76 & -0.76\end{bmatrix}, I=-4.09$$

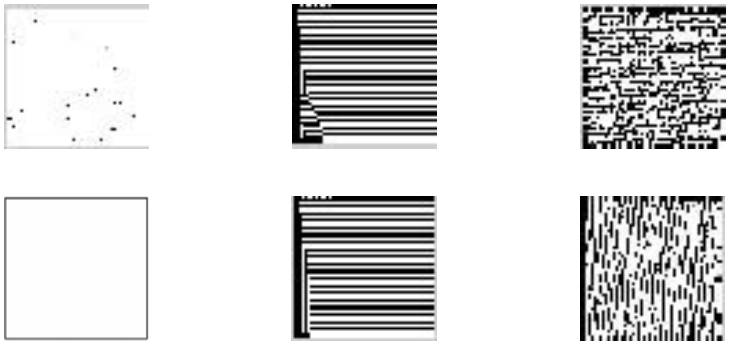
Template 3

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 2.01 & 0\\ 0 & 0 & 0\end{bmatrix}, B=\begin{bmatrix}-0.96 & 0 & -0.96\\ 0 & 2.01 & 0\\ -0.96 & 0 & -0.96\end{bmatrix}, I=-2.5$$

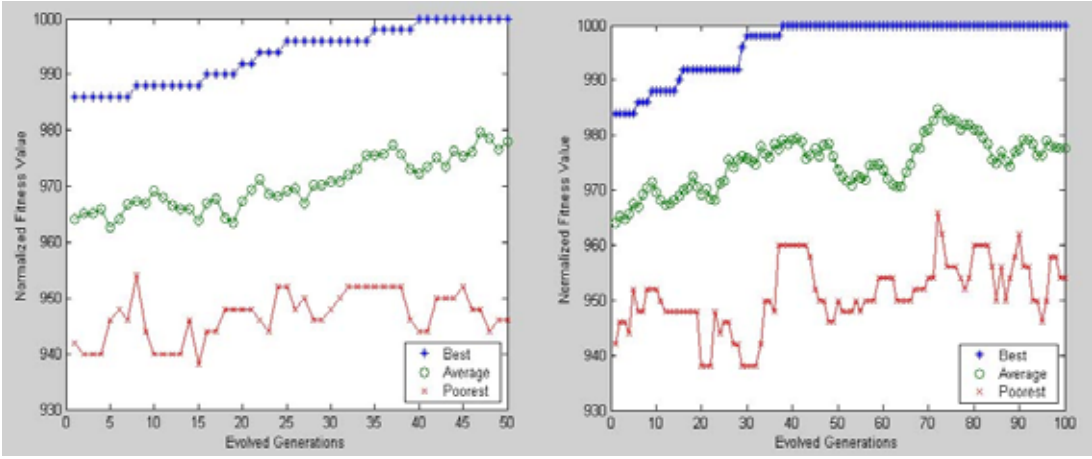
Fig. 4.5.1_1 Optimized CNN template set for all sixteen texture patterns

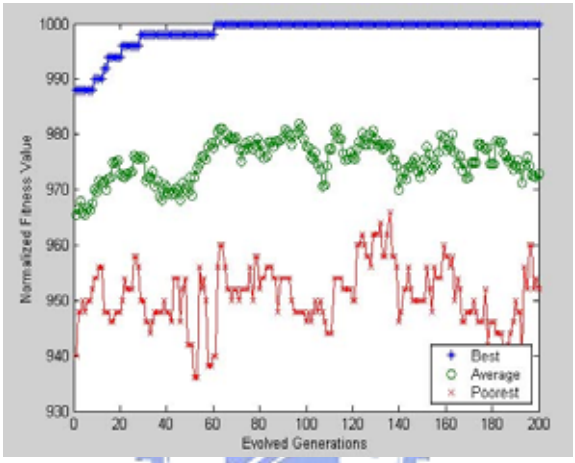


(a)



(b)





(c)

Fig. 4.5.1_2 GA training process for our defined CNN template (a) The misclassified texture patterns in the first run (b) The evolved feature maps during the training phase by GA (c) The corresponding fitness function for the best, average, and poorest populations after 50, 100, and 200 generations accordingly.

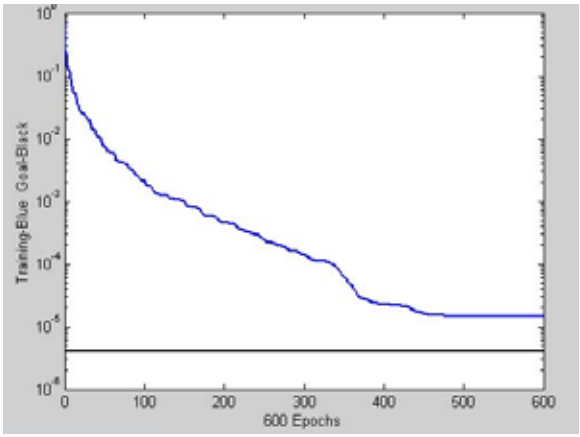


Fig. 4.5.1_3 The convergence curve by GA trainings

4.6 Texture Classification Mechanism

So far it is obvious that we have put forth a new structure in generating CNN templates in a more flexible fashion. This structure also deals with the classification issues in supervised and unsupervised conditions for texture patterns. Here we addressed on how our texture classification mechanism can work for this function. Our entire classification mechanism emphasizes on the decision engine and the classification algorithm besides the searching processes for CNN templates based on GA. As indicated formerly, CNN template set ensures the existence of feature maps for various texture patterns. TBS gives an evaluation standard for determining whether the classified outputs are desired ones and whether TBS series can describe the difference between texture patterns. Therefore, for the decision engine, TBS difference between textures decides whether or not the CNN template has to be proliferated. If TBS difference is greater than some threshold, this TBS series is distinct enough to represent different textures and the classification process is terminated. Otherwise, we must generate more CNN templates to organize a new TBS series for differentiating textures. Also, whether the number of clusters is a *priori* is taken into consideration for different processing purposes. If the number of clusters is known, the classification process would be much easier by applying the pre-classifier and decision engine. Since the features (TBS series) extracted from CNN outputs are quite representative for various texture patterns, a complex or time-consuming classifier is no longer required for pre-classification mechanism. Instead, a simple classifier would work on the classification for features extracted from textures. If the number of clusters is not given, we would use the classification algorithm in the first place to find the number of clusters that will be brought in later. There is no need to apply any existed unsupervised classifier that might be difficult and inefficient for

texture classification. To sum up, we have the classification algorithm on the foundation of TBS series as shown below.

Classification Algorithm

1. *Extract the dominant features from the TBS series.*
2. *Set* $n_{iteration}$ *=1 where* $n_{iteration}$ *is the number of iterations, and calculate the average of all texture patterns.*
   i.e. Calculate $t_c^i = \frac{(t_1^i + t_2^i + \cdots + t_n^i)}{n}$, *for* $1 \le i \le m$ *where* t_c^i *means the centre tuple in the ith dimension, m is the total dimension of each tuple, and n is the total number of tuples (texture patterns).*
3. *Calculate the distance from each tuple to the centre tuple.*
   i.e. $d_c^j = \left\| (t_j^i - t_c^i) \right\|$, *for* $1 \le j \le n$ *where* d_c^j *is defined as the 2-norm distance from the tuple j to the centre tuple in the sum of every dimension i in tuple j.*
4. *Find the maximum distance* $d_{\max}$ *such that* $d_{\max} = \max_{1 \le j \le n} \{d_c^j\}$.
5. *Divide* d_c^j *by* $d_{\max}$ *for* $1 \le j \le n$ *to obtain the normalized distance* d_n^j.
6. *Find the tuple j such that* $d_n^j \ge \eta$ *for* $1 \le j \le n$. *where* η *is the predefined difference ratio between texture patterns and* $0 \le \eta \le 1$. *Let k be the total number of tuples whose j satisfies the above constraint.*
7. *Make the centers of new clusters be* t_j *where* $1 \le j \le k$. *Now the number of clusters will be k+1 (i.e.* $n_{cluster}^r = k+1, and\ r = n_{iteration}$*) since* t_c *should also be one of the centers of clusters.*
8. *Assign the remained unclassified members* t_j *where* $1 \le j \le n - n_{cluster}^r + 1$ *to the cluster whose* d_n^i *is the minimum among all i (*$1 \le i \le n_{cluster}^r$*).*
9. *Recalculate the centers of new clusters to form* c_j *where* $1 \le j \le n_{cluster}^r$.
10. *Evaluate the difference ratio between clusters* r_{ij}.
    where $r_{ij} = \left\| c_i - c_j \right\| / \max_{1 \le i \le k+1} \{\left\| c_i - c_j \right\|\}$ *for* $1 \le i \le k+1$ *and* $1 \le j \le k+1$.
    Find cluster i and cluster j such that r_{ij} *is minimum.*
    If $r_{ij}^{\min} \ge \eta$ *then*
    Stop the algorithm and $n_{cluster} = n_{cluster}^r$ *where* $r = n_{iteration}$.
    Else
    Merge cluster i and cluster j, set $n_{iteration} = n_{iteration} + 1$, *and go back to step 9 with* $n_{cluster}^r = n_{cluster}^{r-1} - 1$ *where* $r = n_{iteration}$.

Like the first step illustrated in this algorithm, extracting the dominant features from TBS series is trying to simplify the analysis of feature curves and enable the

convergence of classification. These dominant features used in this chapter are average height, width, peak value, and the position of the peak value calculated from the feature curve TBS. The classification algorithm helps to find the number of clusters in an efficient way from how clearly we expect the textures to be distinguished. Naturally, the classification mechanism would go back to the proliferating system if the current TBS series are still insufficient to distinguish texture patterns. This then becomes precisely the case where the number of clusters has been given. And the processing procedures would be more lucid after the number of clusters is *a priori*. We will look into the outcomes for different given conditions in the following section to show how both cases demonstrate satisfactory results.

4.7 Experimental Results

This section mainly focuses on the experimental results of classification outcomes. Besides, we would also show the performance of our feature curves presented previously to demonstrate how they may represent various texture patterns in practical use. All the texture patterns we use in this chapter are obtained from Brodatz texture database, and we have sixteen texture patterns for the following experiments. For both experimental and optimized parts, the standard size of texture patterns in this chapter is 64x64 pixels. First, the capability of the feature curve, TBS, in one direction for four texture patterns cropped in different orientations and regularity are shown in Fig. 4.7_1 (a)-(c). Fig. 4.7_1 (a) represents the original four texture patterns in different orientations and regularity with each row representing the same texture patterns we have collected in five different orientations and regularity. Fig. 4.7_1 (b) and (c) indicate the feature maps obtained from the operation of the first template in CNN's and the corresponding feature curves scanning in the horizontal direction. Fig. 4.7_1 (c) figures out the different distributions among these

four texture patterns. Using it as an example in these four textures, the feature curves obtained from the given CNN templates indeed describe the discriminability between different texture patterns.

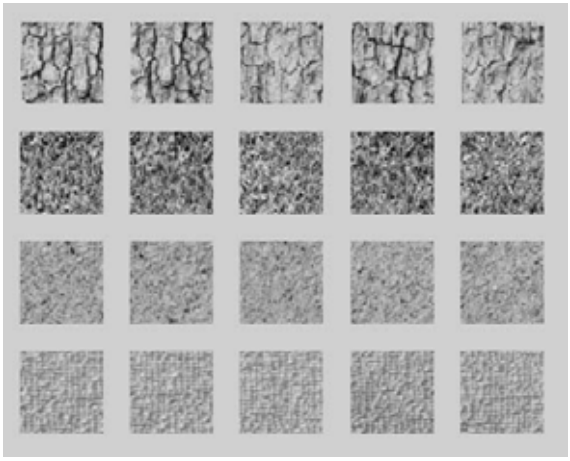
In a more difficult case, the original CNN template is not sufficient to discriminate those eight texture patterns interpreted in Fig. 4.7_2 (a)-(c). Fig. 4.7_2 (a) shows the original eight texture patterns, and Fig. 4.7_2 (b) and (c) both present the same thing as in Fig. 4.7_1 excluding the additional texture patterns. As what our approach highlights, a new CNN template has to be optimized then in order to differentiate the current texture patterns (Fig. 4.7_2 (a)) by the combined TBS series extracted from this new optimized CNN template. With TBS difference ratio, we can foresee whether the texture patterns will lead to the similar or different distribution of TBS. In this way, we need to train one more CNN template for distinct feature maps as shown in Fig. 4.7_3 (a), whereas Fig. 4.7_3 (b)-(c) show the different distribution of TBS in this case. Fig. 4.7_3 (b) and (c) indicate the diverse TBS in the horizontal and vertical directions, which causes the differentiation in some other texture patterns. More specifically, from the 4^{th} row (texture 4) to the 8^{th} row (texture 8) in Fig.4.7_2 (a), (texture 4 and 5) and (texture 7 and 8) in particular, the original feature maps (Fig. 4.7_2 (b)) and TBS (Fig. 4.7_2 (c)) which can not represent or differentiate these textures will be representative enough to discriminate these patterns by incorporating this new TBS (Fig. 4.7_3 (b) and (c)) to form a complete TBS series. Therefore, the experimental results (Fig. 4.7_2 and 4.7_3) show that the proliferated CNN templates accompanied by the original ones give a better representative discriminability for more texture patterns. In order to elucidate the functions of TBS series for texture discrimination, we set up a verifying experiment for numerical comparisons based on these figures (Fig. 4.7_2 and 4.7_3) in Table 4.7_1. The numerical value on each line represents the discrimination index of corresponding textures in Table 4.7_1 (a). The

discrimination index denoted by $\eta(i, j)$ implies the discriminative ability between texture i and texture j (the row number of Fig 4.7_2 or 4.7_3). We only show the smallest top 5 discrimination indices for each figure from left to right. We can observe that all the numbers in Table 4.7_1 (a) are smaller than 0.5, which speaks of the difficulty in separating these textures. Therefore, we demonstrate the functions of TBS series (a combination of more features extracted from more CNN templates) in Table 4.7_1 (b). We would only show the textures which result in the lowest five discrimination indices for comparison. As Table 4.7_1 (b) indicates, the combined TBS series have enhanced the discriminative ability among textures since all the discrimination indices in Table 4.7_1 (b) are higher than those in Table 4.7_1 (a). The quantitative experiments indeed prove that features from the proliferated templates can make texture patterns more discriminative.

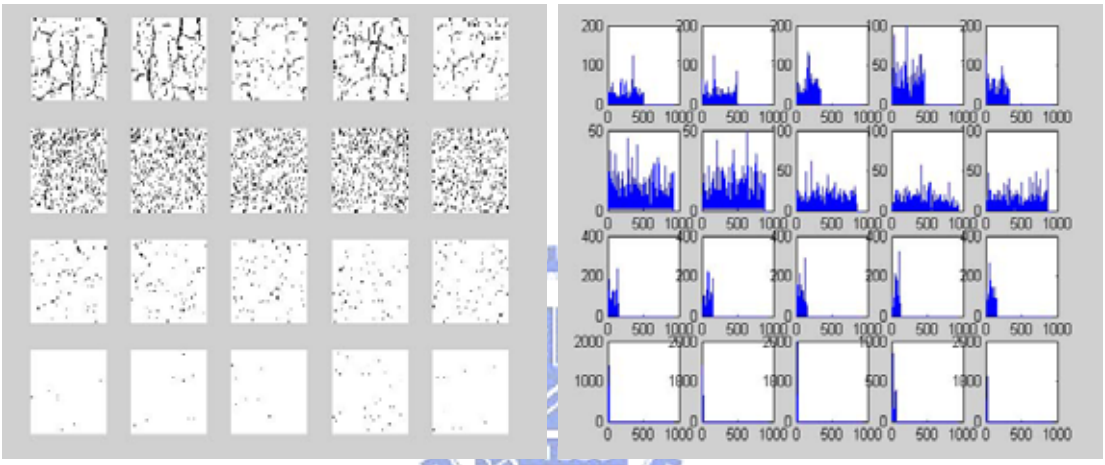
By the same token of satisfactory experimental results, we also demonstrate how effective and prevailing our defined CNN template can be. Our structure is by far better than some other traditional method that takes it as nothing more than a normal template that has nineteen parameters to be trained. Fig. 4.7_4 reveals the convergence condition by the fitness functions of GA's from the best, average, and poorest populations after the same generations for the general CNN template (Fig. 4.7_4 (a)) and our predefined CNN template (Fig. 4.7_4 (b)). Unlike the generally-adopted CNN templates based on GA that often encounter difficulty in the convergence of obtaining a proper template, our results show a much faster convergence speed in the predefined case. In addition, our experiments show that at least three optimized templates would be sufficient enough to classify up to sixteen kinds of texture patterns by using TBS.

To classify less than eight textures in obviously distinct orientations, one out of two templates that is proliferated would be enough to differentiate different texture

patterns. As Table 4.7_2 shows, our introduced feature curve, TBS, does provide a better classification outcome than other features based on CNN's in the previous studies will do. Table 4.7_3 and 4.7_4 indicate respectively the classification results for different rotations and sampling sizes of textures. This suggests the stability and invariance of our approach when applied to different situations. Also, Table 4.7_5 compares the experimental results between different feature curves, and there seems to be no difference even if only one orientation of TBS has been used. Since GA is a training process with optimization, we need to have some discussions about texture data sets on training/testing separation. We use the tenfold cross-validation testing model to make our experimental results more persuasive. Hence Table 4.7_6 indicates the individual and average classification error for a fixed textured data set of eight texture patterns in the standard size (64x64). We can see that the average classification error based on our introduced feature series is much lower than that of other traditional features based on CNN's. Eventually, if the number of clusters is not given, we have the classification results in Table 4.7_7. Table 4.7_7 shows that the number of clusters should increase if the difference ratio between textures (η) decreases and vice versa. In this way, we can decide the number of clusters that we wish to classify the texture patterns in the database by adjusting η. As the classification outcome shows, the error percentage of our introduced system structure is still acceptable even if the number of clusters is not known beforehand. In Table 4.7_7, the error percentage of the fixed η is not given for unsupervised case since it is meaningless to calculate the classification error percentage on the different bases of the real situation. All the experiments indicate that our developed texture classification mechanism with the introduced TBS needs fewer templates for optimization, which for sure result in time-efficiency in the optimizing process of GA and the satisfactory results in texture discrimination as well.

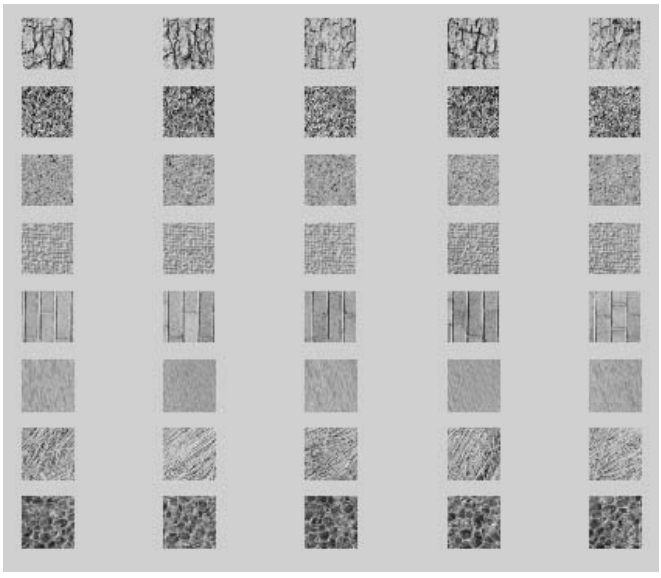


(a)

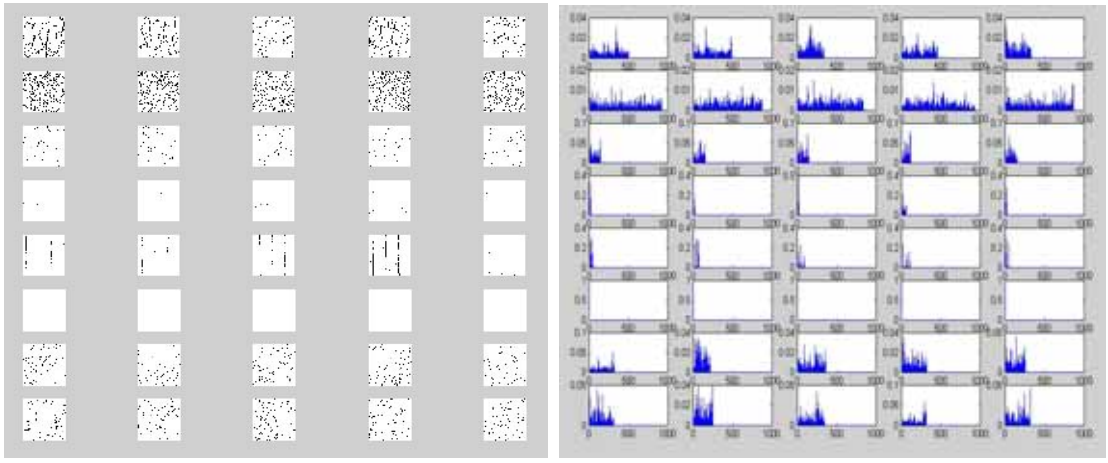


(b) (c)

Fig. 4.7_1 Texture representation for four texture case (a) the original texture patterns (b) feature maps (c) TBS.

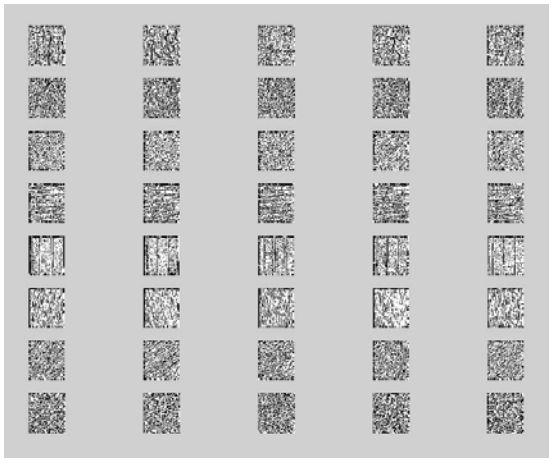


(a)

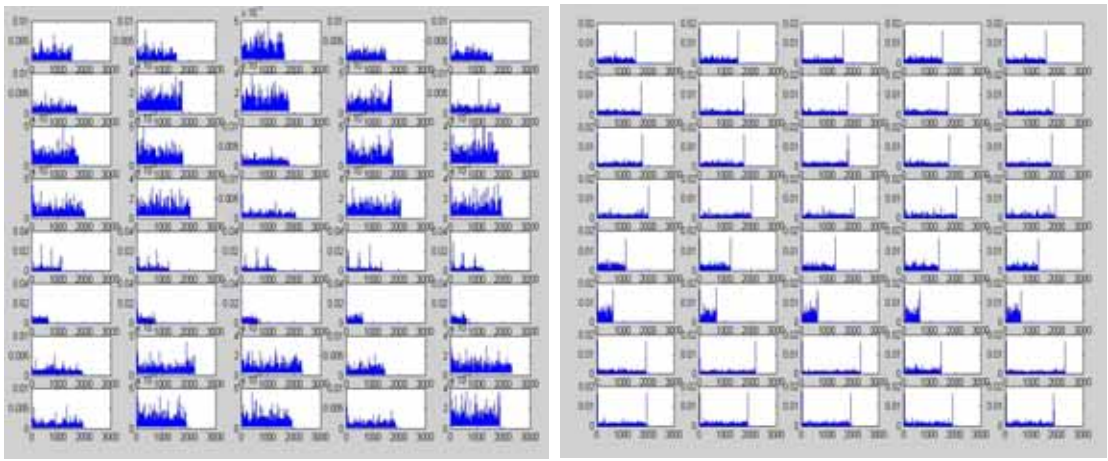


(b) (c)

Fig. 4.7_2 Texture representation for eight texture case (a) the original texture patterns (b) feature maps (c) TBS.

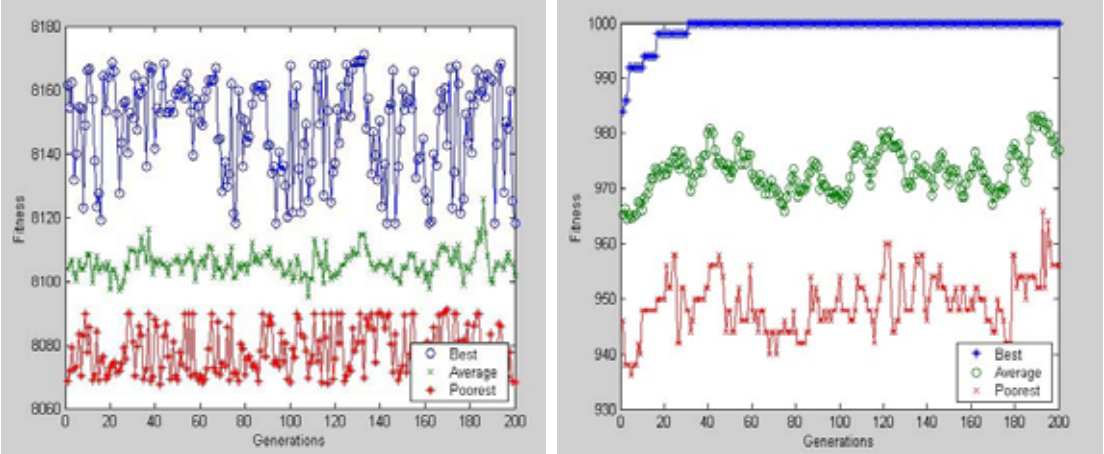


(a)



(b) (c)

Fig. 4.7_3 Texture representation for eight texture case based on another CNN template illustrated by (a) feature maps (b) TBS in the horizontal direction (c) TBS in the vertical direction.



(a) (b)

Fig. 4.7_4 The convergence condition by GA for (a) The general CNN template (19 optimized parameters) (b) Our predefined CNN template

Table 4.7_1 Numerical Comparison for texture discrimination ability based on Fig.4.7_2 and 4.7_3

a. For individual feature curve (by only one CNN template)

Discrimination Index / Subjects	1	2	3	4	5
Fig.4.7_2	η (4,5)=0.12	H(4,6)=0.15	η (5,6)=0.17	η (7,8)=0.22	η (3,7)=0.27
Fig.4.7_3 (b)	η (1,2)=0.13	H(3,4)=0.14	η (2,8)=0.16	η (2,7)=0.21	η (7,8)=0.27
Fig.4.7_3 (c)	η (2,3)=0.10	H(2,4)=0.14	η (1,3)=0.18	η (4,7)=0.19	η (2,8)=0.22

b. Comparison of discrimination index by one feature curve and combined feature curves (TBS series)

Discrimination Index / Subjects	η (2,3)	η (4,5)	η (1,2)	η (2,4)	η (3,4)
One CNN template	0.10	0.12	0.13	0.14	0.14
TBS series	0.51	0.58	0.60	0.55	0.52

Table 4.7_2 Comparison in the classification outcome based on various features

Parameters	Output	Number of templates				Classification error			
		4 textures	8 textures	12 textures	16 textures	4 textures	8 textures	12 textures	16 textures
TBS	Binary	1	2	2	3	0.4 %	3.6 %	4.8%	5.8%
Rob	Binary	3	4	4	5	2.4 %	4.5 %	5.1 %	6.7%
TE	Gray	3	4	4	5	4.5 %	6.3%	7.7 %	8.5%

Table 4.7_3 Experimental results for different rotations of texture patterns

Number of textures	Classification error for different rotations of texture patterns			
	Original (0)	90 degree	180 degree	270 degree
4 textures	0.4%	0.7%	0.5%	0.6%
8 textures	3.6%	3.8%	3.5%	3.7%
12 textures	4.8%	5.4%	5.1%	5.5%
16 textures	5.8%	6.0%	6.2%	5.9%

Table 4.7_4 Experimental results for different sizes of texture patterns

Number of textures	Classification error for different sizes of texture patterns			
	64x64	32x32	16x16	100x100
4 textures	0.4%	1.2%	3.5%	0.3%
8 textures	3.6%	3.8%	4.3%	3.3%
12 textures	4.8%	5.2%	6.5%	4.7%
16 textures	5.8%	6.2%	7.2%	5.8%

Table 4.7_5 Experimental results for different TBS (vertical and horizontal)

Number of textures	Classification error for different TBS	
	Horizontal only	Vertical only
4 textures	0.5%	0.5%
8 textures	4.2%	4.7%
12 textures	5.2%	5.8%
16 textures	6.5%	6.7%

Table 4.7_6 Experimental results for tenfold cross-validation testing model

Features	Average Classification error
TBS	13.5%
Rob	17.8%
TE	25.2%

Table 4.7_7 Experimental results for the case when the number of clusters is not known

Number of textures	Clusters	H	Error (%)	Clusters	η	Error (%)
4 textures	3	0.5	NA	4	0.60	2
8 textures	10	0.5	NA	8	0.55	4
12 textures	8	0.5	NA	12	0.45	5
16 textures	20	0.5	NA	16	0.36	8

4.8 Concluding Remarks

In this chapter, we have brought up a brand-new methodology, where a GA-CNN proliferating system is presented to deal with texture patterns in different given conditions. Here CNN's inevitably take up a crucial part in signal analysis from a perspective of various templates through many kinds of texture patterns. Also, GA is used to optimize our defined templates of CNN's which might project the texture patterns in different feature maps. Most significantly, the feature curve (TBS) which has been shown valid in the representation of texture patterns is introduced to classify textures in the most efficient way by determining if one more CNN template should be proliferated. TBS series could help to determine the number of clusters if it is not given in advance so that the classification process would be much simpler and more flexible. Furthermore, TBS could be applied to any binary images in the distinct feature maps if there is some other analytic tool to generate such feature maps like

CNN’s. Finally, the experimental results corresponding to the newly-introduced approach in this chapter are proven superior to those of other methods. It is therefore surely foreseeable that our structure will be of great help in applications of higher order image processing in the future.

5. Conclusions and Perspectives

In this thesis, we have addressed on the applications of cellular neural networks (CNN's) in high-order image processing. We start with introducing the fundamentals and design guidelines of CNN and GA to deduce the later architecture that we proposed in this thesis. For using CNN's in some specific image application, we use genetic algorithm (GA) to optimize a better CNN template. The major reason that we used GA as an analytical training tool for our issuing problems lies in the prevailing applied field and evolution mechanism of GA. As we know, search of CNN templates could be sometimes too complicated to follow the rules in higher-order image processing. Thus, we introduced some basic but significant CNN template examples at the first place so as to inspire more applications based on CNN's by observing the nature of images. Most significantly, we make use of CNN's to deal with some difficult issues of image processing, image descreening in particular, which has been regarded as a tough question since no single solution can be applied to all cases. However in this thesis, we classify the original screening images in two classes beforehand by our defined index, and a different simple filter would be applied to each of these two classes. In this application, CNN plays an important role in classification of images by looking upon screening patterns as different textures. The descreened images hence could be more satisfactory and acceptable to human perception.

What's more, we focus on texture discrimination and representation for more image applications related to texture analysis. In this aspect, we proposed a brand-new structure to proliferate CNN templates, which makes decision of CNN templates more flexible. We do not have to assume how many templates to be optimized at the beginning of texture classification, and instead we adaptively determine the number of

optimized templates for the characteristics of the texture set. CNN can be no doubt used here on account of its filtering capability and attraction of hardware implementability. To characterized CNN more sensitively in various kinds of texture patterns, we also presented a series of features, Transition Bit String (TBS) defined in this thesis, for more practical uses. Also, our feature series could not be necessarily carried out by CNN. To be more organized, we have the following major contributions in this thesis.

A. We propose a systematical method for image descreening.
B. We inspire an idea of screening pattern classification in advance for image descreening, which makes image documentation simpler and more impressive.
C. We present a proliferation structure to determine CNN templates in a more flexible fashion.
D. We innovate a kind of characterizing features for texture analysis.
E. We give more possibilities of hardware implementations in high-order image processing.

To sum up, this thesis not only provides an approach in the recent state-of-the-art but also gives more chances and perspectives for hardware implemented in high-order image applications. In addition, the indices and features proposed in this thesis would bring about more solutions in the complicated problems of image processing. We shall at this place list some researching fields that might be carried out in the future.

A. Hardware implementation

The main content of this thesis would like to apply the CNN structure to more applications in pursuit of future hardware implementation or chip design. Therefore, this future work might be the most important and ultimate issue among many kinds of applications inspired by this thesis.

B. Augment of more CNN channels

CNN used to apply to applications of image processing only in gray-scale, which might cause the loss of some important information of original images. It is certainly believed that more channels of CNN implementation could do a great help for some specific applications of images that should depend on the color information of original images. So the implementation of more CNN channels might provide more solutions for more complicated problems of image processing.

C. Feature selection of CNN proliferation structure

In this thesis, we only take into consideration the generative way of extracted features for our CNN proliferation structure due to the augmented properties of TBS feature series. Of course, for the concept of feature selection, how to reduce the number of features might also be useful in pattern recognition. The consideration of how to select useful features could be put forth in the future.

D. More applications or generalization of TBS

We have presented the basic definition and formula of TBS in this thesis, and TBS series have also been proven to be useful in representation of various texture patterns. As what we have concluded earlier, TBS series would not necessarily be applied to CNN outputs, and these feature curves could be quite useful for analyzing image data in software-oriented applications. More specifically, TBS series could be expressed in various cases of requirements of images. For example, the setting of threshold, the scanning directions, the way of counting pixels, and so on would provide more flexible tuning parameters for more versatile image applications.

E. More applications of image processing by CNN

No doubt, more applications of image processing are the major goals that this

thesis pursues. This thesis has offered more way of using CNN output and illustrated the relationship between CNN structure and image data. It is observed that the results convolved from CNN outputs could also be expressed in terms of various sorts of definitions for images. More complicated applications of image processing could be consequently carried out in the future by our proposed feature series as well as the constructed structure.

In this section, we have concluded the major contributions and the possible future works in details. Hence, we would be dedicated to solving more and more issues of digital image processing by our proposed strategies using CNN in the future.

Bibliography

[1] L. O. Chua and T. Roska, “Stability of a class of nonreciprocal cellular neural networks,” *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1520-1527, 1990.

[2] F. Zou and J. A. Nossek, “Stability of cellular neural networks with opposite-sign templates,” *IEEE Trans. on Circuits Syst.*, vol. 38,pp. 675–677, June 1991.

[3] I. Fajfar and F. Bratkovi¢c, “Design of monotonic binary-valued cellular neural networks,” in Proc. IEEE Int.Workshop Cellular Neural Networks Applications, Sevilla, Spain, pp. 321–326, June 1996.

[4] B. Mirzai, D. *Lẏm*, and G. S. Moschytz, “On the robust design of uncoupled CNN’s,” in Proc. European Symp. Artificial Neural Networks, Bruges, Belgium, pp. 297–302, Apr. 1998.

[5] J. A. Nossek, “Design and learning with cellular neural networks,” in Proc. IEEE Int.Workshop Cellular Neural Networks Applications, Rome, Italy, pp. 137–146, Dec. 1994.

[6] C. *G̈* uzelis and S. Karamahmut, “Recurrent perceptron learning algorithm for completely stable cellular neural networks,” in Proc. IEEE Int. Workshop Cellular Neural Networks Applications, Rome, Italy, pp. 177–182, Dec. 1994.

[7] T. Kozek, T. Roska, and L. O. Chua, “Genetic algorithm for CNN template learning,” *IEEE Trans. Circuits Syst. I*, vol. 40, pp. 392–402, June 1993.

[8] B. Chandler, C. Rekeczky, Y. Nishio, and A. Ushida, “Using adaptive simulated annealing in CNN template learning—A powerful alternative to genetic algorithms,” in Proc. European Conf. Circuit Theory Design, Budapest, Hungary, pp. 655–660, Sept. 1997.

[9] G. Seiler, A. J. Schuler, and J. A. Nossek, “Design of robust cellular neural networks,” *IEEE Trans. Circuits Syst. I*, vol. 40, pp. 358–364, Mar. 1993.

[10] I. Fajfar and F. Bratkovi˘c, “Optimizing cellular neural networks for robustness and speed,” in Proc. European Conf. Circuit Theory Design, Istanbul, Turkey, Aug. pp. 781–784, 1995.

[11] M. H¨anggi and G. S. Moschytz, “Genetic optimization of cellular neural networks,” in Proc. IEEE Int. Conf. Evolutionary Computation, Anchorage, AK, pp. 381–386, 1998.

[12] Hanggi, M. and Moschytz, G.S., “Stochastic and hybrid approaches toward robust templates,” in Proc. IEEE Int. Workshop Cellular Neural Networks Applications, London, U.K., Apr. pp. 366–371, 1998.

[13] L. O. Chua and L. Yang, “Cellular neural networks: Applications,” *IEEE Trans. Circuits Syst. I*, vol. 35, pp. 1273–1290, Oct. 1988.

[14] J. K. Hawkins, “Textural properties for pattern recognition,” *Picture Procesping and Psychopictorics, Bernic Sacks Lipkm and Azriel Rosenfeld (EDS)*. New York: Academic Press, 1969.

[15] J. Serra, “Theoretical bases of the Leitz texture analyses system,” *Leifz Sci. Tech. Inform.*, Supplement 1, 4, pp. 125-136, Apr. 1974 (Wetzlar, Germany).

[16] G. Matheron, “Elements Pour Une Theorie des Milieux Poreu,” Paris, France: Masson, 1967.

[17] J. Weszka, C. Dyer, and A. Rosenfeld, “A comparative study of texture measures for terrain classification,” *IEEE Trans. on Sys., Man , and Cyber.*, vol. SMC-6, no. 4, pp. 269-285, Apr. 1976.

[18] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Mass., 1989.

[19] L. Davis, ed., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[20] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolutionary Programs*, Springer-Verlag, Berlin, 1992.

[21] J.H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, Ann Arbor, Mich., 1975.

[22] R. R. Brooks, Lynne Grewe and S. S. Iyengar, “Recognition in the wavelet domain: A survey”, *Journal of Electronic Imaging*, pp. 757-784, July 2001.

[23] L.-K. Shark and C. Yu, “Denoising by optimal fuzzy thresholding in wavelet domain,” *Electronics Letters*, Vol. 36, pp. 581 –582, March 2000.

[24] R.D. Nowak and R.G. Baraniuk, “Wavelet-domain filtering for photon imaging systems,” *IEEE Trans. on Image Processing*, Vol. 8, pp. 666 –678, May 1999.

[25] D. Margulis, *Mathematics, Morie, and the Artist*, CATFWorld, Sep/Oct, 1997.

[26] R. Ulichney, *Digitial halftoning* , MIT press, Cambridge, Massachusetts, 1987.

[27] P. W. Wong, “Inverse halftoning and kernel estimation for error diffusion,” *IEEE Trans. Image Processing*, vol. 4, pp. 486–498, Apr. 1995.

[28] S. Hein and A. Zakhor, “Halftone to continuous-tone conversion of error-diffusion coded images,” *IEEE Trans. Image Processing*, vol. 4, pp. 208–216, Feb. 1995.

[29] D. C. Youla and H. Webb, “Image restoration by the method of convex projections—Part I: Theory,” *IEEE Trans. Med. Imag.* vol. MI-1, pp. 81–94, Oct. 1982.

[30] D. F. Dunn, T. P. Weldon and W. E. Higgins, “Extracting halftones from printed documents using texture analysis,” *Optical Engineering*, vol 36, no 4, pp. 1044-1052, 1997.

[31] A. Jaimes, F. Mintzer, A.R. Rao, and G. Thompson, " Segmentaion and automatic descreening of scanned documents," *Spie*, vol. 3648, pp. 517-528, 1999.

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

[33] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Bio. Cybern.*, vol. 52, pp. 141-152, 1985.

[34] J. J. Hopfield and D. W. Tank, "Computing with neural circuits: a model," *Science (USA)*, vol. 233, no. 4764, pp. 625-633, 1986.

[35] D. W. Tank and J. J. Hopfield, "Simple 'neuron' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Tran. Circuits Syst.*, CAS-33, pp. 533-541, 1986.

[36] F. Zou, S. Schwarz, and J. A. Nossek, "Cellular neural network design using a learning algorithm," in Proc. IEEE Int. Workshop on Cellular Neural Networks and Their Applications, pp. 73-81, 1990.

[37] S. Schwarz and W. Mathis, "A design algorithm for cellular neural networks," in Proc. 2nd Int. Conf. Microelectronics for neural network, pp. 53-59, 1991.

[38] H. Harrer. "Multiple layer discrete time cellular neural networks using time variant templates," *IEEE TCAS-II*, vol 40, n.3, pp. 191-199, 1993.

[39] P. Lopez, D.L. Vilarino and D. Cabello, "Design of multilayer discrete time cellular neural networks for image processing tasks based on genetic algorithms," ISCAS 2000, n.4, pp. 133-136, 2000.

[40] P. Lopez, M. Balsi, D.L. Vilarino and D. Cabello, "Design and training of multilayer discrete time cellular neural networks for antipersonnel mine detection using genetic algorithms," in Proc. 6th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, pp. 363-368, 2000.

[41] D.L. Vilarino, D. Cabello, M. Balsi and V. M. Brea, "Image segmentation based on active contours using discrete time cellular neural networks," in Proc. fifth IEEE Int. Workshop on Cellular Neural Networks and Their Applications, pp. 331-336, 1998.

[42] P. Lopez, D.L. Vilarino and D. Cabello, "Genetic algorithm based training for multilayer discrete time cellular neural networks," IWANN'99. Lecture Notes in Computer Science, vol. 1607, pp. 467-476, Springer-Verlag, 1999.

[43] Tamas Sziranyi and Marton Csapodi, "Texture classification and segmentation by cellular neural networks using genetic algorithms," *Computer Vision and Image Understanding*, vol. 71, No. 3, Sep, pp. 255-270, 1998.

[44] M. Levine, *Vision in Man and Machine*, McGraw-Hill, 1985.

[45] R. Haralick, "Statistical and structural approaches to texture," Proc. IEEE, vol.

67, no. 5, pp.786- 804, 1979.

[46] G. Cross and A. Jain, "Markov random field texture models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 5, no. 1, pp. 25-39, 1983.

[47] A. Pentland, "Fractal-based description of natural scenes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 661-674, 1984.

[48] R. Chellappa and S. Chatterjee, "Classification of textures using gaussian markov random fields," *IEEE Trans. Acoustic, Speech and Signal Processing*, vol. 33, no. 4, pp. 959-963, 1985.

[49] H. Derin and H. Elliot, "Modeling and segmentation of noisy and textured images using Gibbs random fields," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, no. 1, pp. 39-55, 1987.

[50] B. Manjunath and R. Chellappa, "Unsupervised texture segmentation using markov random fields," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 478-482, 1991.

[51] A. Rosenfeld and J. Weszka, "Picture recognition" in *Digital Pattern Recognition*, K. Fu (Ed.), Springer-Verlag, pp. 135-166, 1980.

[52] J. Daugman, "Uncertainty relation for resolution in space, spatial frequency and orientation optimised by two-dimensional visual cortical filters," *Journal of the Optical Society of America*, vol. 2, pp. 1160-1169, 1985.

[53] A. Bovik, M. Clark, and W. Giesler, "Multichannel texture analysis using localised spatial filters," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, pp. 55-73, 1990.

[54] S. Mallat, "Multifrequency channel decomposition of images and wavelet models," *IEEE Trans. Acoustic, Speech and Signal Processing*, vol. 37, no. 12, pp. 2091-2110, 1989.

[55] A. Laine and J. Fan, "Texture classification by wavelet packet signatures," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1186-1191, 1993.

[56] C. Lu, P. Chung, and C. Chen, "Unsupervised texture segmentation via wavelet transform," *Pattern Recognition*, vol. 30, no. 5, pp. 729-742, 1997.

[57] B. Mirzai, Z. Cheng, and G. S. Moschytz, "Learning algorithms for cellular neural networks," in Proc. IEEE Int. Symp. Circuits Systems, Monterey, CA, vol. 3, pp. 159–162, June 1998.

[58] L. O. Chua and P. Thiran, "An analytic method for designing simple cellular neural networks," *IEEE Trans. Circuits Syst. I*, vol. 38, pp. 1332–1341, Nov. 1991.

[59] M. Hanggi and G. S. Moschytz, "An exact and direct analytical method for the design of optimally robust CNN templates," *IEEE Trans Circuits Syst. I*, vol. 46,

no. 2, pp. 304-311, Feb. 1999.

[60] Tamas Sziranyi and Attila Hanis, “Sub-pattern texture recognition using intelligent focal-plane imaging sensor of small window-size,” *Pattern Recognition Letters*, pp. 1133-1139, 1999.

[61] Tamas Sziranyi and Josiane Zerubia, “Markov random field image segmentation using cellular neural network,” *IEEE CAS-I: Fundamental Theory and Applications*, vol. 44, no.1, 1997.

[62] Tamas Sziranyi, “Texture segmentation by the 64X64 CNN chip,” in Proc. IEEE Int.Workshop Cellular Neural Networks Applications, pp. 547-554, 2000.

[63] Tamas Sziranyi and Jozsef Csicsvari, “High-speed character recognition using a dual cellular neural network architecture (CNND),” *IEEE CAS-II: Analog and Digital Signal Processing*, vol. 40, no. 3, March 1993.

[64] John P. Miller, Tamas Roska, Tamas Sziranyi, Kenneth R. Crounse, Leon O. Chua, and Laszlo Nemes, “Deblurring of images by cellular neural networks with applications to microscopy,” in Proc. Third IEEE Int.Workshop Cellular Neural Networks Applications, pp. 237-242, 1994.

[65] Tamas Sziranyi, “Texture recognition using a superfast cellular neural network VLSI chip in a real experimental environment,” *Pattern Recognition Letters*, pp. 1329-1334, 1997.

[66] L. O. Chua and L. Yang, “Cellular neural networks: Theory,” *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1257-1272, 1988.

[67] Y.-W. Shou and C. T. Lin, “Image descreening by GA-CNN-Based Texture Classification,” *IEEE Trans. on circuits and system I*, vol. 51, no.11, pp. 2287-2299, 2004.

[68] Leon O. Chua, *CNN: A paradigm for complexity*, World scientific, 1998.