Modeling and characterizing novel pulsed neural networks for real-time applications

Zhuang, Hualiang

2011

Zhuang, H. (2011). Modeling and characterizing novel pulsed neural networks for real-time applications. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/46323>

<https://doi.org/10.32657/10356/46323>

**MODELING AND CHARACTERIZING
NOVEL PULSED NEURAL NETWORKS
FOR REAL-TIME APPLICATIONS**

ZHUANG HUALIANG

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2011

Acknowledgments

It is my great pleasure to express gratitude to Associate Professor Low Kay Soon, my supervisor, for his helpful advice, guidance, encouragement and patience throughout the research work leading to this thesis. His knowledge and enthusiasm in his work have greatly motivated me to challenge many issues that have arisen during this project. I am also indebted to Dr. Yau Wei Yun for his valuable support and instructive suggestion. Many thanks are given to the School of Electrical and Electronic Engineering for the provision of environment and equipments to the research project.

Finally, I would like to express my appreciation to my family for the love and support.

Summary

Biologically inspired artificial neural networks have been well researched and found in many applications. In contrast to the conventional artificial neural network models, pulsed neural models can explain some phenomena of information coding with more realistic biological interpretation. In this thesis, we propose and investigate some new models of pulsed neural networks, including pulse-based radial basis function (RBF) network and multi-channel pulse coupled neural network (MPCNN). The proposed neural computation is suitable for full parallelism implementation, which is a salient merit that can be exploited for a multitude of real-time applications.

Different from conventional RBF networks, the proposed pulse-based RBF network utilizes the synchronism of temporal pulses generated by model neurons to perform RBF computations. This bio-inspired design is highly advantageous due to its parallelism characteristics. Based on a novel winner neuron selection algorithm, which is designed by employing the rank order of neurons' output pulses, it gives comparable performance when benchmarked with the conventional counterparts in clustering and nonlinear function approximation applications. The rank order based paradigm is also extended to deal with some complex clustering problems such as processing of meshed clusters which is difficult if conventional methods such as k -means and self-organizing map (SOM) are used. As a 2-dimensional (2-D) variant of pulsed neural networks, the proposed MPCNN overcomes the shortcoming of the conventional scalar-based PCNN for processing the multi-spectral images. MPCNN is a multi-spectral image segmentation approach that uses the timing of individual pulse produced by the neurons to determine the distances between pixels' feature vectors and

their rank order respectively. Consequently, the fast links can be established among neurons with respect to the spectral feature vectors and spatial proximity of mapped pixels. Furthermore, the image segmentation algorithm is designed to enable neurons' connectivity in the 2-D space through intra/inter-segment linking. Utilizing temporal pulse behavior, these novel parallel neural circuits can be designed with more compactness for implementation on hardware platform such as FPGA. It also overcomes the bottleneck of computational burden in contrast to its sequential-code-based counterparts.

The new pulsed neural networks have been validated experimentally for several real-time applications, including satellite image processing, color image segmentation for object detection, real-time object-tracking in video recording, and friction estimation of a precision linear stage. The analysis on computational complexity and experimental evaluation on performance show that the proposed approaches outperform their conventional and state-of-the-art counterparts.

Table of Contents

Acknowledgements	i
Summary	ii
Table of Contents	iv
List of Figures	viii
List of Tables	xi
List of Abbreviations	xii
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	4
1.3 Major Contributions of the Thesis	6
1.4 Organization of the Thesis	8
2 Literature Review	10
2.1 Introduction	10
2.2 Integrate-and-Fire Model	10
2.3 Temporal Coding	14
2.4 Network Learning	15
2.5 Applications and Hardware Implementations	18
2.6 Conclusions	19
3 Pulse-Based RBF Neural Networks	22
3.1 Introduction	22

3.2	Design of Pulse-Based Model Neuron	24
3.2.1	Basic model	24
3.2.2	Discussion on alternative distance measures	29
3.3	Network Architecture	32
3.4	Unsupervised Clustering Algorithm	32
3.4.1	Learning algorithm	32
3.4.2	Discussion about the convergency	35
3.5	Basic Clustering Examples	36
3.5.1	Cluster radius	36
3.5.2	Noisy case	38
3.5.3	Cluster merging	39
3.5.4	Clustering of realistic data	40
3.6	Complex Clusters	41
3.7	Supervised Learning for Nonlinear Function Approximation	45
3.7.1	Network model	45
3.7.1.1	Winner-takes-all	46
3.7.1.2	Optimal interpolation	47
3.7.2	Gradient descent learning	49
3.7.3	Example of nonlinear function approximation	54
3.8	Conclusions	56

4 On-Chip Implementations of the Pulsed RBF Neural Circuits and Their Applications	58
4.1 Introduction	58
4.2 Implementation of Pulse-Based RBF Model with Unsupervised Learning	60
4.2.1 Synapse unit	62
4.2.2 Integrate-and-fire circuits	64
4.2.3 Circuits for network learning	65
4.3 Application to Friction Estimation Based on Nonlinear Look-up Table	68
4.3.1 Pulsed RBF neural circuit as a nonlinear look-up table for friction estimation	68
4.3.2 Experimental results	71
4.4 Application to Object-Tracking Based on Clustering	74
4.4.1 Designs of neural circuits and algorithms	76
4.4.2 Experimental results	80
4.5 Application to Analysis of Satellite Image by Combining with GA	84
4.5.1 Enhanced GA for training of pulsed neural net	84
4.5.1.1 Representations	86
4.5.1.2 Fitness function evaluation	87
4.5.1.3 Selection and mating	88
4.5.1.4 Finalization of GA	89
4.5.2 Hardware platform	91
4.5.3 Experimental results for satellite image processing	92
4.5.3.1 Data and parameters	92

4.5.3.2	Performance evaluation of the GA-PNN	94
4.6	Conclusions	98
5	Multi-Channel Pulse Coupled Neural Networks	100
5.1	Introduction	100
5.2	Multi-Channel PCNN for Color Image Segmentation	104
5.2.1	Multi-channel PCNN	104
5.2.2	Algorithm for color image segmentation	107
5.3	Evaluation on Segmentation Performance	113
5.4	Application to Object Detection	117
5.4.1	Grouping segments	117
5.4.2	Shadow-based object detection	120
5.5	Experimental Results	121
5.5.1	Evaluation on computational complexity	122
5.5.2	Performance of object detection	124
5.6	Conclusions	127
6	Conclusions and Future Work	129
6.1	Conclusions	129
6.2	Recommendation for Future Research	131
	Author's Publications	134
	Bibliography	136

List of Figures

2.1	Typical shape of response functions of post-synapses. (a) EPSP; (b) IPSP	11
2.2	A typical IAF model	12
2.3	Typical shape of the threshold function of IAF model	13
3.1	Pulsed neural network with RBF units. Dashed lines denote the synaptic delays c_{ij}	24
3.2	Waveforms of the dynamics of the neural model shown in Fig. 3.1	26
3.3	(a) Misclassification rates against data dimension; (b) Relative variance values against data dimension	31
3.4	Result of clustering for different values of α : (a) $\alpha = 0.7$; (b) $\alpha = 0.35$	37
3.5	Relationship between α and the number of clusters in $\mathbf{R}^2$ example	38
3.6	Noisy case with $\alpha = 0.35$. (a) result with $\bar{\zeta}_j = 0$; (b) result with $\bar{\zeta}_j = 5$	38
3.7	Redundant cluster centers. Circles denote the cluster centers	39
3.8	Incorrect clustering result for concave clusters. Circles denote the cluster centers	42
3.9	Network with lateral connections	43
3.10	Clustering results using the network with lateral connections	44
3.11	Pulsed neural network for function approximation	45
3.12	Relationship between Δt_y and Δy	51

3.13 Result of nonlinear function approximation. (a) winner-takes-all scheme; (b) optimal interpolation scheme	55
3.14 Mean absolute error (MAE) during gradient descent learning	56
4.1 Block diagram of the hardware system	60
4.2 Circuits for synapse unit, integrate-and-fire and register of ax_i	62
4.3 Signal waveforms of synaptic counter	63
4.4 Masking/unmasking circuit and comparison circuit of α and firing time of the winner neuron	64
4.5 Circuit diagram for updating the synaptic delays	67
4.6 Experimental linear stage system	68
4.7 Friction-speed curve from offline experiments	69
4.8 Block diagram of the linear motor system	70
4.9 Acquired data of velocity and motor current	72
4.10 Instantaneous relationship between the velocity and the calculated friction	73
4.11 Distribution of the RBF neurons after network training based on the training data set	73
4.12 Block diagram of the object-tracking system	75
4.13 Experimental FPGA system	81
4.14 Results of identifying and locating different objects using clustering algorithm	82
4.15 Relationship between number of objects and average lags using different methods	83
4.16 Model of the pulsed neural network	85
4.17 Architecture of the GA-PNN	86
4.18 Learning-based classification	93

4.19	Effect of α to the classification accuracy	95
4.20	Effect of M_p to the classification accuracy	96
4.21	Accuracy of GA-PNN and BP net with various epoch numbers	96
4.22	Comparison of classification results	97
5.1	Model neuron of Eckhorn's PCNN	101
5.2	Multi-channel PCNN with 2D array of neurons	103
5.3	Pulse based RBF model	106
5.4	Observation before and after region-merging operation	111
5.5	Impact of parameters for region merging	112
5.6	Segmentation results of examples	114
5.7	Segmentation results in pseudo-color for images polluted by salt-and-pepper noise	116
5.8	Comparison of F -measure versus different noise density	117
5.9	Potential objects grouped from segmentation result in (b)	121
5.10	Block diagram of the hardware system	122
5.11	Comparison of processing time between SRG and MPCNN with respect to various numbers of resultant segments	123
5.12	Results of various cases where suspected shadows are highlighted in white color	126
5.13	PCA based target recognition	127
5.14	Detection accuracy for various objects.	127

List of Tables

2.1	Feature comparison of various pulsed neural networks	21
3.1	Clustering of Fisher's Iris Dataset. ($k=3$ for the k -means; 3 Kohonen neurons for SOM.)	41
3.2	RBF neurons	55
3.3	Weight values w_j	55
3.4	MAE for various function approximation methods	55
4.1	Linear motor's specifications	71
4.2	Results for friction estimation using different methods and parameters	74
4.3	Error rates of object tracking	83
4.4	Spectral bands	94
4.5	Hardware resource utilization	98
5.1	Comparison of segmentation results among various methods	115
5.2	Summary of impact on segmentation & object detection for various cases in Fig 5.12.	125

List of Abbreviations

AER	Addressed event representation
ART	Adaptive resonance theory
BP	Back propagation
CCD	Charge-coupled device
CMOS	Complementary metal oxide semiconductor
EEG	Electroencephalogram
EPSP	Excitatory post-synaptic potential
FPGA	Field programmable gate array
GA	Genetic algorithm
GPU	Graphics processing unit
IAF	Integrate-and-fire
IPSP	Inhibitory post-synaptic potential
ISI	Inter-spike interval
MAE	Mean absolute error
MAPE	Mean absolute percentage error
MLP	Multi-layer perceptrons
MPCNN	Multi-channel pulse coupled neural network
OPB	On-chip peripheral bus
PCA	Principle component analysis

PCNN	Pulse coupled neural network
PHY	Physical layer of the open systems interconnection model
PLB	Processor local bus
PNN	Pulsed neural network
PSP	Post-synaptic potential
PWM	Pulse-width modulation
RBF	Radial basis function
RNG	Random number generators
SNN	Spiking neural network
SOM	Self-organizing map
VLSI	Very-large-scale integration

Chapter 1

Introduction

1.1 Motivation

From the rudimentary model of perceptrons [1, 2] to various state-of-the-art artificial neural networks, most of the innovations in this field are inspired by the biological knowledge. One of the well-known neurobiological evidences is that neurons produce all-or-none action potentials (i.e. spikes or pulses) with certain electrochemical activity. Moreover, the firing rate of these pulses will increase with increasing stimulus intensity [3]. This phenomenon leads to a general belief that biological neural systems use these pulses (or pulse trains) to encode and transmit information. In most mathematical neural models, a common hypothesis is that a continuous variable can be interpreted as an average or instantaneous firing rate of pulses. For example, analog values in the popular sigmoidal gates are interpreted as average firing rate of biological neurons [4]. This is based on the knowledge that the neurons fire at various intermediate frequencies between certain lower and upper limits.

Can we conclude such interpretation of instantaneous or average firing rate completely capture the reality of biological neural coding and computation? Doubts arise at least from the phenomena of fast analog computations by networks of neurons in the visual cortices of

primates. A number of researchers [4, 5] have demonstrated that the primate brain can analyze and classify visual pattern in about 100msec, in spite of the fact that it involves at least 10 synaptic stages (i.e. layers) from the retina to the responsible cortical neurons. This means that each individual processing stage would operate in less than 10msec. On the other hand, the firing rates of neurons involved in these computations are usually below 100Hz due to the fact that a biological neuron has refractory periods that prevent it from generating a large number of pulses in a short period. This implies that such processing can be accomplished under the condition where each individual neuron only gets to fire either none or one spike. In this vision case, it appears questionable to use the instantaneous firing rate of a single neuron, i.e. the interval between two sequential spikes, as a way of information coding. This time constraint could become even more severe if much faster processing in other sensory pathways is considered. For example, neurons in the bat auditory cortex can response in 8 msec through layers of processing after the stimulus onset [5].

Consider the population rate coding [6], which encodes and transmits a signal using the average firing rate of a population of multiple neurons. It seems that this method could overcome the issue of instantaneous firing rate of single neuron because one can obtain whatever number of pulses in a time slot as long as there are enough independent neurons acting in that time slot. Nonetheless, the reality of biological nervous system in turn will yield bandwidth constraint. The pulse generation process can be described as a Poisson process [4]. For example, we assume the neuron can be observed emitting one pulse during a 10msec observation of Poisson window. Considering a Poisson process with 100Hz rate value, there will be at least 281 independent neurons, i.e. 281 Poisson processes, that are

required to obtain a precision of $100 \pm 10\text{Hz}$ in the 90% confidence interval. Otherwise, the precision of the population rate coding would be too low for realistic perception, such as primate vision. Neurobiological evidence also shows that only a few hundred thousand nerve fibers are effective for transmitting visual signals [7]. If 281 of such fibers are used to transmit the intensity of each pixel in an image, then only about 30×30 pixels can be sampled in a visual channel. This seems incompatible with the fact that visual system of primate can perceive highly detailed information. Some problems about firing rate coding have also been reported in [8, 9]. These constraints indicate that the conventional notion whereby a continuous variable can be interpreted as an average or instantaneous firing rate of pulses may not precisely reflect the reality of biological neurons. Neglecting these evidences might lead to missing an opportunity to discover useful characteristics of neural networks.

A number of experimental results indicate that biological neural systems use the timing of individual pulse, correlation between pulse trains or other time-domain characteristics of pulses produced by related neurons to encode information [10-12]. Inspired by these biological phenomena, researchers began to investigate a new generation of model neurons and related neural networks [14-17], which employ temporal pulse based neurons as computational units. In their works, detailed temporal activities of pulses, including pulse-forming mechanism, dynamics of transmission and reception, and interactions based on the phase properties of the pulse trains, are taken into account for neural computing. These neural network models generally can be referred to as pulsed neural networks [18].

In contrast to the conventional neural network models (based on sigmoidal gates etc.), pulsed neural models not only can explain some phenomena of information processing in the brain with more realistic biological interpretation. Moreover, they have also proven to yield competitive computation performance [14, 15]. These neural models have hardware implementation advantage as the temporal pulse based coding provides potential compactness. It is also more robust than the analogue implementation. In addition, these research works can motivate further exploration on the unclear computation mechanism of brain, and inspire the development of new artificial neural networks.

1.2 Objectives

The core objective of this thesis is to investigate the characteristics of pulsed neural networks in contrast to conventional neural network models. Based on novel proposed pulsed neural networks, the research will be conducted to evaluate their property including computational complexity and performance. The detailed objectives are listed below.

- A new pulse-based RBF neural model with highly parallelism characteristics will be proposed. It will be extended to network structures for classification and nonlinear approximation. Learning algorithms based on the concepts of ART (adaptive resonance theory) [19] and GA (genetic algorithm) [13] will be developed to provide computations compatible with characteristics of the pulsed RBF model.
- Evaluation of various aspects of the proposed pulsed RBF neural model and algorithms will be conducted. This includes the comparison with the conventional counterparts in

clustering and nonlinear function approximation applications. The clustering for concave data set, which cannot be handled by conventional k -means and SOM, will be tested using the rank order based paradigm of the pulsed neural network. The performance of nonlinear function approximation will also be compared with other pulsed neural network counterpart.

- Utilizing temporal pulse behavior, neural circuits will be designed for both forward computation and feedback learning with more compactness for hardware implementation in massive parallelism. This design will be tested by running on programmable platform, such as FPGA. The scheme will be validated experimentally for practical applications.
- A 2-D variant of pulsed neural networks, multi-channel PCNN (MPCNN), will be proposed. The design will be catered to color or multi-spectral image processing. The corresponding image segmentation algorithm utilizing temporal pulse behavior of the multi-channel PCNN will also be developed so that it can be implemented in full parallelism in 2-D space.
- The proposed multi-channel PCNN will be evaluated for color image segmentation. Using common quantitative measures, the segmentation performance of the proposed MPCNN will be compared with conventional and state-of-the-art methods for segmentation of noisy images while its improvement on computational complexity are also analyzed in contrast to the sequential-code-based counterparts.
- We will develop the pulsed neural network based systems for real-time applications, including satellite image processing, color image segmentation for object detection, real-time object-tracking in video recording, and friction estimation of a precision linear stage.

1.3 Major Contributions of the Thesis

The major contributions of this thesis are summarized as follows:

- We propose a novel pulse-based RBF neural network in Chapter 3. Different from the conventional RBF networks, the proposed pulse-based RBF network utilizes the synchronism of temporal pulses generated by the neurons to perform RBF computations. This bio-inspired design is highly advantageous due to its parallelism characteristics. Based on a novel winner neuron selection algorithm, which is designed by employing the rank order of neurons' output pulses, it has comparable performance when benchmark with the conventional counterparts in clustering and nonlinear function approximation applications. The rank order based paradigm is also extended to deal with some complex clustering problems such as processing of meshed clusters which is difficult if conventional methods such as k -means and self-organizing map (SOM) are used. Its capability of noise-resistance is satisfactory. The effects of various kernel functions are also analyzed to guide the selection of RBF kernel considering both the performance requirement and the convenience for hardware implementation. These research works have been reported in our publications [20, 21].
- A new multi-channel pulse coupled neural network (MPCNN) [22] is proposed for real-time processing of multi-spectral images in Chapter 5. As a 2-D variant of pulsed neural networks, MPCNN overcomes the shortcoming of the conventional scalar-based PCNN. In this research, MPCNN is designed as a multi-spectral image segmentation approach that uses the timing of individual pulse produced by the neurons to determine the distances between pixels' feature vectors and their rank order respectively. Consequently, the fast links can be established among neurons with respect to the spectral feature vectors and spatial proximity

of mapped pixels. Furthermore, the image segmentation algorithm is designed by utilizing the 2-D structure of PCNN, where neurons' connectivity may transfer in parallel in the 2-D space to establish intra/inter-segment linking. Analysis shows that its sensitivity to the algorithm parameters are significantly reduced as compared with the conventional segmentation counterparts such as seeded region growing algorithm. Using common quantitative measures, the segmentation performance of the proposed MPCNN is comparable to conventional and state-of-the-art methods for segmentation of noisy images while its full parallelism can improve the processing speed by up to hundreds times as compared with the sequential-code-based counterparts. In contrast to conventional coded neural network models, another salient point of this pulsed neural network is that it can produce satisfactory results without iterative learning. This effectively avoids the potential issues of convergency and time complexity.

- The characteristics of the pulse based computing are utilized to design the hardware neural circuits for forward computation, feedback learning and simultaneous 2-D linking on programmable platform, such as FPGA. Using the temporal pulse behavior, the neural circuits can be built compactly in massive parallelism. More specifically, the firing time and rank order of pulses excited by input signals are employed to compute the vector distance and select the winner neuron in the competitive learning. ART (adaptive resonance theory) and Genetic algorithm are modified to seamlessly merge into pulsed neural circuits based on their suitability to digital computation [20, 23]. These designs provide the methodology for overcoming the bottleneck of computational burden of conventional neural networks. Chapter 4 reports the detailed schemes and experimental results. The research work

contributes a practical basis for further biologically inspired methods, which require real-time computing.

- The new pulsed neural networks have been validated experimentally for several real-time applications [24, 25], including satellite image processing, color image segmentation for object detection, real-time object-tracking in video recording, and friction estimation of a precision linear stage. The analyses on computational complexity and experimental evaluation on performance show that the proposed approaches outperform their conventional and state-of-the-art counterparts.

1.4 Organization of the Thesis

This thesis consists of six chapters. Chapter 1 states the motivations, objectives and major contributions of the thesis. In Chapter 2, we give a survey on the research works of the pulsed neural networks in various aspects, including basic neuron models, temporal coding of signals, network structures, learning algorithms and hardware implementations. Chapter 3 proposes a model of pulsed RBF neural network. An unsupervised learning algorithm is proposed for this pulsed RBF neural network to perform clustering. The proposed scheme is extended to deal with some complex clustering problems such as processing of meshed clusters. A supervised learning algorithm is also designed for the enhanced neural network to operate nonlinear function approximation. Comparisons and benchmarking with state-of-the-art counterparts are discussed. In Chapter 4, we design hardware circuits for the proposed pulsed neural network with learning algorithm to be implemented on FPGA. A GA based supervised clustering algorithm is also proposed and implemented. Furthermore, these pulse-

based neural circuits are validated experimentally for several real-time applications, including friction estimation of a precision linear stage, object-tracking in video recording, and satellite image processing. In Chapter 5, a multi-channel PCNN with segmentation algorithm for color image processing is proposed. This investigation has an intrinsic connection to the pulsed RBF model developed in Chapter 3. Firstly, both pulsed RBF and MPCNN are pulse-based neural model. They utilize the time-domain characteristics of pulses to implement the neural computations, which are different from concept of conventional neural networks. Secondly, the designs of pulsed neural networks in both Chapter 3 and Chapter 5 are appropriate to massive parallel computing platform. This point overcomes the computation bottleneck of conventional neural networks. Finally, the pulsed RBF model is a general classifier based on integrate-and-fire model. This classifier is applied to MPCNN as a similarity computing unit. In addition to the 2D linking phase of MPCNN in chapter 5, the pulse-based design and parallelism-oriented algorithm of these two phases are merged seamlessly. Also, chapter 3 can be considered as basis and tool for an extension and specific application in chapter 5. The detailed research of Chapter 5 is organized as follows. Besides the design of MPCNN, the selection of the algorithm parameters is discussed. Its segmentation performances for noisy images are evaluated using quantitative measures, including *Precision*, *Recall* and *F-measure*. The scheme has been applied to some applications such as military object and runway detections. Chapter 6 concludes the thesis and recommends some future research directions.

Chapter 2

Literature Review

2.1 Introduction

The motivation of research in pulsed neural networks was presented in the previous chapter. The research can be traced back to early works of decades ago [12, 26, 27]. In recent years, pulsed neural networks attracted more attentions due to the development in theories and applications. The state-of-the-art platforms of software and hardware provide the possibility for implementing new bio-inspired designs based on neuromorphics and digital embedded techniques. In this chapter, we present a survey on the research works of the pulsed neural networks covering basic neuron models, temporal coding of signals, network structures, learning algorithms and hardware implementations. While presenting these peers' literatures, relevant aspects of our proposed research works are also described for comparison.

2.2 Integrate-and-Fire Model

Integrate-and-Fire (IAF) models are widely used to describe the dynamics of neural models, which is the fundamental of pulse-based neural models. IAF models can be traced back to the early research by Lapique [26, 27]. There exist a number of IAF models with slight variations. Maass proposed a formal description in the form of network [28] and named it as spiking neural network (SNN). We consider the SNN as a typical IAF model and review it in the following.

Let the network consists of a finite set V of spiking (or pulsed) neurons, a set $E \subseteq V \times V$ of synapses, a weight $w_{u,v} \geq 0$ and a response function $\varepsilon_{u,v} : \mathbf{R}^+ \rightarrow \mathbf{R}$ for each synapse $\langle u, v \rangle \in E$, and a threshold function $\Theta_v : \mathbf{R}^+ \rightarrow \mathbf{R}^+$ for each neuron $v \in V$. Here $\mathbf{R}^+$ is defined as $\mathbf{R}^+ := \{x \in \mathbf{R} : x \geq 0\}$. The postsynaptic response function $\varepsilon_{u,v} : \mathbf{R}^+ \rightarrow \mathbf{R}$ can be selected in different forms to cater for the respective requirement, for example, linear, leaky integrator or various piecewise functions. $\varepsilon_{u,v}$ can be either excitatory or inhibitory modulation to the postsynaptic potential (EPSP or IPSP) as shown in Fig. 2.1[14].

Fig. 2.1. Typical shape of response functions of post-synapses. (a) Excitatory postsynaptic potential (EPSP); (b) inhibitory postsynaptic potential IPSP.

If $F_u \subseteq \mathbf{R}^+$ is the set of firing times of a neuron $u \in V$, then the potential at the trigger zone of neuron v at time t is given by

$$S_v(t) := \sum_{u: \langle u, v \rangle \in E} \sum_{s \in F_u: s < t} w_{u,v} \cdot \varepsilon_{u,v}(t-s), \quad (2.1)$$

where $w_{u,v} \geq 0$ is the weight of the synapse from neuron u to neuron v [28], and c denotes the synaptic delay.

Fig. 2.2. A typical IAF model.

Fig. 2.2 illustrates a typical connectionist of the IAF model. Each dashed segment respectively denotes the synaptic delay c_i of synapse $\langle u_i, v_j \rangle$, $i = 1, \dots, N_i$. N_i is the number of input channels. An input temporal pulse from neuron u_i reaches the post-synaptic terminal (the node at the end of the dashed segment) and excites the post-synaptic potential (PSP) $\varepsilon_{u,v}(t - c_i)$ after a latency c_i . In a noise-free IAF model, a neuron v_j fires at time t_j once the time varying integrated potential $S_j(t)$ reaches the threshold $\Theta_v(t - t')$, where t' is the time of the most recent firing of v_j . Thus firing time t represents the output value of the IAF neuron.

To reflect the biological reality, a refractory period of the neuron is considered as follows: if a neuron v has fired at time t' , it will not fire again for some time (e.g. a few msec). This refractory effect can be modeled by a suitable threshold function $\Theta_v(t-t')$ [14] as shown in Fig. 2.3. One can observe the difference between this temporal pulse-based IAF model and the well-known sigmoidal gate, whose output value is thought to represent the firing rate of the neuron.

Fig. 2.3. Typical shape of the threshold function of IAF model.

In some literatures, variants of IAF models with noise models have also been reported [29, 30]. Based on these models, some researchers investigated in theoretical analyses. Gerstner analyzed the dynamics of integrate-and-fire neurons with various noise models [29]. The stability conditions for the procedure of spike firing are investigated [18]. In the theory reported in [14], Maass analyzed the computational capability of spiking neurons with regard to the necessary number of neurons. He proved that spiking neurons have intrinsic advantage in contrast to sigmoidal gates for the computation of Boolean functions, while their capability for the computation of real-value functions is comparable to sigmoidal gates. This advantage is valid only for the computation of elementary model neurons. For structural models such as

multi-layer networks, the tedious neural dynamics barricades the functional development of the networks. In application oriented research, more efficient variants have been investigated [20, 22, 31].

2.3 Temporal Coding

Based on the hypothesis that the firing times of neurons or the correlations between pulse trains in a biological neural system could encode relevant information, various coding schemes for temporal structure based neural computation have been proposed [4, 10-12].

One efficient way of encoding analog information is through differences in the firing times of different neurons [14, 15]. It is sometimes referred to as delay coding. For example, a vector $\langle x_1, \dots, x_n \rangle \in [0, 1]^n$ can be encoded by the firing times $T - a \cdot x_i$ of n neurons, where T is a reference time and a is a constant.

Maass *et al* [14, 30] gave theoretical analyses on computation power of the spiking neural networks with delay coding. They concluded that the computation power of a spiking neural network is in general superior to a feed-forward sigmoidal neural network in equivalent scale. In addition, they discussed neural coding through correlation in firing times in the light of biological evidence that correlations between firing times of neurons are relevant for neural coding and computation. They also investigated in a temporal coding scheme based on inter-spike intervals (ISI) pattern of a pulse train, and illustrated its capability as pattern detector [30].

Different from Maass' pattern detector, Hopfield's research [15] focuses on pulsed neurons that could store multi-dimensional pattern with suitable synaptic delay pattern. This neural model can solve analogue pattern recognition problem if the pulsed neurons work as coincidence detectors. In contrast, a sigmoid gate that can only operate on linear separation [32] would have difficulties to solve the same problem. Hopfield indicated that this model neuron in fact worked as a radial basis function unit. However, he did not discuss the implication of various aspects of this neural model, such as the mathematical representation of the model, the activation-function dependent kernel function and implementation-oriented learning algorithm etc. In this research, we investigate the issues with regards to delay coding and propose related pulse-based RBF neural circuits [20].

As an extension of delay coding method, Thorpe et al [33] proposed a "rank order code" by using the firing rank order of a neuron population to quantify analog value. This can speed up the pulse based coding procedure so that the pulsed neural network can be applied to rapid image processing. In [34], another coding method based on the population of pulsed neurons was reported. It employs a set of neurons and encodes real-valued variables to temporal pulses by using graded and overlapping activation functions (e.g. Gaussian) of these neurons [34]. Bohte et al [34] proposed this encoding method for transforming continuous data into spike-time patterns and applied it to unsupervised clustering.

2.4 Network Learning

Most pulsed neural networks take the synaptic weight as a type of plasticity. Popular learning methods such as Hebbian learning and gradient descent rule have been employed for

updating the synaptic weights in some applications [34-37]. Ruf [38] proposed a mechanism for unsupervised learning based on self-organization map using temporal coding. Bohte et al [39] derived an error-back-propagation algorithm (BP) for multi-layer pulsed neural networks, analogous to the derivation by Rumelhart et al [40]. As a variant, Storck et al investigated the synaptic delay adaptation [41]. They proposed a dynamic synaptic model with an adaptable parameter that determines the time of the maximum excitatory postsynaptic potential (EPSP). Supervised learning algorithm was also designed for speech recognition [42].

In [34, 39, 41], the synaptic delay is treated as the average delay of a “sub-synapse” population. While this architecture reflects the biological realities, it increases the computation of the model and confines itself from the more flexible supervised and unsupervised learning algorithms. In our pulse-based RBF neural network [20], we abstract the synaptic delay as a directly adjustable variable in this neuron model for ease of computation and circuit realization. A flexible ART-like (ART is the acronym of Adaptive Resonance Theory [19]) learning algorithm is proposed for this pulsed RBF neural network to implement clustering. Unlike the Hebbian learning algorithm that needs to preset the number of RBF neurons and the initial features [35, 37], the proposed scheme generates the RBF neurons depending on the feature of input data during the learning process. Moreover, it also avoids the intrinsic local- minimum issue of typical BP algorithms [39].

Pulse coupled neural network (PCNN) [43, 44], which is proposed based on the linking field model of Eckhorn [16], is a special category of pulsed neural networks. PCNN can be considered as a 2-D variant of pulsed neural networks. In a typical PCNN, neurons are

constellated as a 2-dimensional (2-D) array in a single layer so that pixels of an input image can be corresponded to the neurons by one-to-one mapping for neural processing. Inter-neuron linking is established based on the dynamics of neurons to simulate and explain the experimentally observed feature-dependent activity of cortical neurons in the visual cortices of mammals such as cats and monkeys [13, 45]. If the receptive field for perceptual signals or signals from foregoing layers in multi-layer networks is named as the primary feeding receptive field, PCNN's central new concept is the introduction of a secondary receptive field, i.e. the linking field. The linking field integrated input from intra-layer neighboring neurons modulates the primary feeding receptive field input. This coupling action provides an effective way for synchronous pulse dynamics in 2-D networks [43, 46-48]. In the theoretical research, Johnson analyzed the properties of PCNN, and proved that PCNN had translation, rotation, scale, distortion, and intensity signal invariance for image processing [17]. PCNN algorithm has quite a few of parameters. To clarify the impacts of the parameters, Kuntimad and Ranganath proposed analytical guidelines for the algorithm parameters, which can yield robust performance [31]. Its application includes scene segmentation [31, 44, 49-52], image smoothing [53], image fusion and decomposition [54, 55], path optimization [56] and image feature extraction etc [57- 59].

The typical PCNN model is confined to the processing of grayscale images. One of the reasons is its inter-neuron coupling is defined in terms of products of scalars. For pixels featured with color vectors, the relationship between the timing of a pulse and a pixel's feature will no longer be scalar-to-scalar. Another reason is the computational burden of the algorithm. A typical PCNN is a 2-D array of neurons with one-to-one pixel to neuron correspondence. Including its complex dynamics and multiple inter-neuron coupling, the

computation will be undoubtedly time-consuming for color images if executed in sequential codes. Recently, some researchers investigated in applying PCNN to multi-channel images. In [54], a model named m-PCNN is reported to perform medical image fusion, which focuses on merging multiple images into a single image to improve the quality of the resulting image. The aforementioned issues are still remained as it directly augments the original single-channel PCNN by extending the number of input channels. Its computation complexity is as high as the conventional PCNN.

In this thesis, we propose a multi-channel pulse coupled neural network (MPCNN) [22] to overcome the shortcoming of the conventional scalar-based PCNN. MPCNN is a multi-spectral image segmentation approach that uses the timing of individual pulse produced by the neurons to determine the distances between pixels' feature vectors and their rank order respectively. Consequently, the fast links can be established among neurons with respect to the spectral feature vectors and spatial proximity of mapped pixels so that color image segmentation can be implemented by intra/inter-segment linking in the 2-D space. Another salient point of this pulsed neural network is that it can produce satisfactory results without iterative learning in contrast to conventional neural networks. This effectively avoids the potential issues of convergence and time complexity.

2.5 Applications and Hardware Implementations

Inspired by biological phenomena, pulsed neural networks have found their potential applications in various fields, such as pattern recognition [34, 41], nonlinear function approximation [36, 60], vision and image processing [33, 48, 54], path optimization [56], auditory modeling [61, 62] and speech recognition [42, 59] etc.

Most of the applications of pulsed and conventional neural networks are carried out using software simulator. This limits one of the most important advantages of neural networks, namely the massive parallelism. Hardware implementation is the direct way for fully parallel computation, and VLSI designers and neuromorphics researchers have put a lot of effort [63-65]. There are two options, namely the analogue and digital techniques, for neural VLSI designers. Analogue technique is attractive for reasons of compactness, potential speed and low power. But its robustness to noise and design flexibility are far from satisfaction. Its counterpart, digital technique, can overcome those disadvantages but it is usually area and power-hungry because of its bit-wise approach. Pulsed neural model provides the possibility for combining the advantage of both analogue and digital techniques [18]. For example, pulsed neural circuits require only one line to transmit one channel signal coded by temporal pulse train. This makes it to be comparable in compactness to analogue circuits while the pulsed signals can be designed to be more robust than the analogue signals. Research works based on customized VLSI have been reported [63-65]. To improve the design flexibility, FPGA systems, which can be designed to simulate pulsed computation, are employed as a moderate option in some works [66, 67].

2.6 Conclusions

In this chapter, the literature review starts from original integrate-and-fire neural models, which are motivated by the neurobiological evidence. Various signal-coding methods based on temporal pulses, which are expected to reflect the reality of biological neural systems, are surveyed. Related learning algorithms and parallelism-oriented schemes for hardware

implementation are also reviewed. Table 2.1 summarizes the features of various pulsed neural networks. Amongst multi-layer networks (MLP), BP is the most frequently reported learning algorithm. This may be due to its popularity in the field of conventional neural networks. However, it is prone to result in local minimum. This intrinsic weakness limited it from further development. Single channel PCNN is another attractive direction in the area of pulsed neural network owing to its capability on image processing. But it is desired to be improved to deal with multi-spectral images and overcome the bottleneck of computation burden. For RBF networks, most of the reported works employ the Hebbian algorithm for network training because they focus on the details of biological phenomena. In contrast, our works are implementation-oriented schemes. We proposed pulse-based RBF with GA and ART-like learning algorithm, and multi-channel PCNN on FPGA [20-25] for the practical applications of the pulsed neural networks. In summary, as compared with the traditional neural networks, pulsed neural networks could provide an insight into mechanism of information transmission and computation in the biological neural system. The research in this direction hopefully could discover further novel approach to the problems of neural computation, and could in the long term overcome the limitations of state-of-the-art technology.

Table 2.1 Feature comparison of various pulsed neural networks.

topology/ learning algorithm	signal coding		pulse interval	rank order	population coding	advantage	disadvantage
	timing of individual pulse						
MLP net	Gradient decent/BP	[14,30]	[66, 67]	[4, 33]	[39]	differentiable activation function; fewer neurons.	local minimum; slow.
	biologically plausible	[35]			[41, 42]	bio-inspired	Heuristic; case dependent
	SOM	[38]				unsupervised; fast.	inapplicable to supervised case
RBF net	Hebbian	[15]			[34, 37]	bio-inspired	slow; case dependent
	ART-like	[20, 21, 24]				flexible; hardware implementation.	moderate performance
	GA	[23]				parallel; global convergence; hardware implementation	moderate performance
single-channel PCNN	non-iterative	[17], [43-59]				2-D structure; without iterative learning.	intensive computing
multi-channel PCNN	non-iterative	[22, 54]		[22, 25]		multi-channel 2-D structure; without iterative learning; parallel hardware implementation.	intensive computing with sequential codes.
advantage		novel bio-inspired operation; hardware implementation.	hardware implementation.	novel bio-inspired operation; fast; hardware implementation	bio-inspired		
disadvantage		moderate performance	operation constraints	moderate performance	intensive computing		

Chapter 3

Pulse-Based RBF Neural Networks

3.1 Introduction

In Chapters 1 and 2, various aspects of pulsed neural networks have been introduced. Besides explain some phenomena of information processing in the brain with more realistic biological interpretation, they also yield competitive computation performance in contrast to the conventional neural models. Hence they have attracted much attention from researchers in the field of neural computation in recent years [14-67].

Amongst various pulsed neural models, Hopfield's model [15] is designed as a set of coincidence detectors for multi-channel input pulses. It can be used to solve analogue pattern recognition problem naturally. Hopfield indicated that these neuron models in fact worked as radial basis function (RBF) units. However, he did not discuss the detailed implication of various aspects of this neural model, such as the mathematical representation of the model, the activation-function dependent kernel function and implementation-oriented learning algorithm etc. This original idea has subsequently been extended to a number of sophisticated schemes. Natschläger and Ruf [37] propose a RBF network using spiking neurons, which encode the input pattern in the synaptic delays. Each of the delays is treated as the average

delay of a “sub-synapse” population, and is adjusted indirectly by adjusting efficacies of the “sub-synapse” population. The firing time of the spiking neuron has a regular dependency on the Euclidean distance between input vector and the center vector reflected in the synaptic delays. Bohte et al [34] combine the architecture of this spiking neural network (SNN) with a temporal coding method that uses overlapping Gaussian receptive fields to translate analog input values to spike-times. A Hebbian learning algorithm is designed for this spiking neural network (SNN) to implement unsupervised clustering.

In both [34] and [37], the synaptic delay is treated as the average delay of a “sub-synapse” population. While this architecture reflects the biological realities, it increases the computation of the model and confines itself from the more flexible unsupervised learning algorithm. In this research, we propose a new model of pulsed RBF neural network based on Hopfield’s original idea. This model is designed such that the firing times of its output pulses depend on the synchronism and the occurring time of the synchrony of the input pulses. The synchronism in turns can be optimized by coordinating the relationship between the pattern of the input pulses and the synaptic delay pattern. For ease of computation and circuit realization, we abstract the synaptic delay as a directly adjustable variable in this neuron model. A flexible ART-like learning algorithm is proposed for this pulsed RBF neural network to implement clustering, where the rank order of neurons’ output pulses is employed for winner neuron selection algorithm. Unlike the Hebbian learning algorithm that needs to preset the number of RBF neurons and the initial features, the proposed scheme generates the RBF neurons dynamically depending on the feature of input data during the learning process. The rank order based paradigm is also extended to deal with some complex clustering problems such as processing of meshed clusters which is difficult if conventional methods

such as k -means and self-organizing map (SOM) are used. The network is also extended to operate nonlinear function approximation. Comparisons with established approaches will be discussed in this chapter.

Fig. 3.1. Pulsed neural network with RBF units. Dashed lines denote the synaptic delays c_{ij} .

3.2 Design of Pulse-Based Model Neuron

3.2.1 Basic model

The proposed model of the pulsed neural network is shown in Fig. 3.1, which is an extension of Fig. 2.2. In contrast to Fig. 2.2, multiple neurons v_j are constellated to form a layer J with full connection to the neurons u_i in layer I. The neurons in layer J receive input pulses through connections from input neurons $u_1, \dots, u_i, \dots, u_{N_i}$. For each neuron v_j , $j = 1, \dots, N_j$, it will function as a RBF unit. The dashed segment between every input neuron to the corresponding synaptic terminal (the node at the end of the dashed segment) denotes the

delay, which is defined as the difference between the firing time of pre-synaptic neuron and the time the post-synaptic terminal is activated.

In this study, the input neurons are assumed to receive real-valued inputs. The information is encoded to the corresponding firing times of input pulses by the following rules [14],

$$t_i = T_{in} - a \cdot x_i, \quad i = 1, \dots, N_i, \quad (3.1)$$

where t_i is the firing time of the i^{th} input neuron; x_i is the sampled value of i^{th} component of the input vector; T_{in} is a reference time; a is a constant. This is similar to the concept of PWM if t_i is regarded as the time of the rising edge of a PWM pulse. The model for a pulse-based neuron in layer J is basically of the integrate-and-fire (IAF) type as follows

$$S_j(t) = f\left(\sum_{i=1}^{N_i} \varepsilon(t - t_i - c_{ij})\right), \quad (3.2)$$

where $S_j(t)$ is the integrated membrane potential of neuron v_j ; c_{ij} is the delay from the i^{th} input neuron to the j^{th} post-synaptic terminal; N_i is the number of input neurons; $\varepsilon(\cdot)$ is the response function, which describes the postsynaptic potential (PSP) excited by a single input pulse, and it is defined as

$$\varepsilon(u) = \begin{cases} r + r_0 & 0 \leq r \leq \tau \\ 0 & \text{else} \end{cases}, \quad (3.3)$$

where r_0 is a bias and $[0, \tau]$ is the region of effective response. $f(\cdot)$ in (3.2) is an activation function which is defined in this research based on a linear function $f(s) = s$.

With the model (3.1)-(3.3), the integrated potential $S_j(t)$ will be excited to the maximal response if all excitatory pulses reach their post-synapses synchronously. The worse synchronism they have, the weaker the integrated response will be generated. In the following, we show that this pulsed neural model works as a RBF unit.

Fig. 3.2. Waveforms of the dynamics of the neural model shown in Fig. 3.1.

Fig. 3.2 shows the response waveform of the pulsed neural network shown in Fig. 3.1. The neural computation is conducted based on periodic cycles. In each cycle time T_k , $k = 1, 2, \dots$, the network is designed to complete one input-output action. Each cycle consists of three stages. The duration of each stage is T_m . For the input layer, all input neurons must fire

pulses before T_{in} . This time segment is referred to as input stage. These pulses arrive at their post-synaptic terminals through different delay paths and activate their post-synaptic terminals at time $t_i + c_{ij}$, $i = 1, \dots, N_i$, respectively. The integrated potential of neuron v_j is computed by (3.2). In this model, it is set to be refractory before $2T_{in}$, i.e. the threshold is assumed as ∞ during this time. In the time region $[2T_{in}, 3T_{in}]$, which is referred to as output stage or trigger zone, the threshold is set as $\theta = N_i T_{in}$ and the neuron v_j will fire an output pulse at time

$$t_j = \arg_t(S_j(t) = \theta). \quad (3.4)$$

where $\arg_t(\cdot)$ denotes the solution of argument t . Once it fires, the neuron enters into the refractory status again.

A typical RBF is usually selected as a function of Euclidean distance between a center and the data point. To be consistent with this essential fact, the synaptic delays c_{ij} are selected such that their patterns reflect the center of input data points. The output of neuron v_j is a pulse fired at time t_j when the integrated potential $S_j(t)$ reaches the threshold θ . We define the bias r_0 in (3.3) as follows so that t_j is an exact radial basis function of the Euclidean distance between the center vector $\mathbf{c}_j = [c_{1j}, \dots, c_{ij}, \dots, c_{N_i j}]^T$ and input vector $\mathbf{aX}_{in} = [ax_1, \dots, ax_i, \dots, ax_{N_i}]^T$,

$$r_0 = (t_i + c_{ij} - T_{in}) - (t_i + c_{ij} - T_{in})^2, \quad i = 1, \dots, N_i. \quad (3.5)$$

In (3.5), $t_i + c_{ij}$ is the time when the input pulse of the i^{th} channel reaches the terminal node.

Combining (3.2), (3.3) and (3.5), we have

$$S_j(t) = \sum_{i=1}^{N_i} ((t - t_i - c_{ij}) + (t_i + c_{ij} - T_{in}) - (t_i + c_{ij} - T_{in})^2). \quad (3.6)$$

As a quantitative form of (3.4), the firing time t_j of the neuron v_j can be derived by substituting (3.1) and the threshold θ into (3.6),

$$t_j = \frac{\theta}{N_i} + T_{in} + \frac{1}{N_i} \sum_{i=1}^{N_i} (c_{ij} - a \cdot x_i)^2. \quad (3.7)$$

Equation (3.7) reveals the explicit RBF relation between t_j and the Euclidean distance between the vectors $\mathbf{c}_j$ and $a\mathbf{X}_{in}$. The shorter the Euclidean distance will lead to earlier firing of the neuron v_j .

If we assume the synaptic delays are plastic, the pattern of these delays can be coordinated properly so that an input vector with related pattern would excite the output pulse as required. Furthermore, this plasticity allows the delay pattern to be trained by customized algorithm with the training dataset.

The definition of r_0 in (3.5) makes the pulsed neural model acts as a Euclidean distance dependent radial basis function. Besides Euclidean distance, function of a general measure of distance can also be used to compute the kernel function of the pulse-based RBF model. The general RBF form of (3.7) can be written as

$$t_j = f(\|c_j - aX_{in}\|^p). \quad (3.8)$$

Other examples of popular distance metrics includes L_1 -norm and L_∞ -norm, i.e. $p=1$ and $p=\infty$ in (3.8) respectively. The L_∞ -norm distance will be explained in the next section. Here we take the L_1 -norm distance as an example. To compute the L_1 -norm distance, r_0 is set as

$$r_0 = (t_i + c_{ij} - T_{in}) - |t_i + c_{ij} - T_{in}|. \quad (3.9)$$

Similar to the derivation of (3.6), the integrated potential is obtained as

$$S_j(t) = \sum_{i=1}^{N_i} ((t - t_i - c_{ij}) + (t_i + c_{ij} - T_{in}) - |t_i + c_{ij} - T_{in}|). \quad (3.10)$$

Then t_j in (3.8) is formulated as

$$\begin{aligned} t_j &= f(\|c_j - aX_{in}\|^1) \\ &= \frac{\theta}{N_i} + T_{in} + \frac{1}{N_i} \sum_{i=1}^{N_i} |c_{ij} - a \cdot x_i|. \end{aligned} \quad (3.11)$$

This distance metric is also known as Manhattan distance, which is popular in some fields such as image processing.

3.2.2 Discussion on alternative distance measures

When the RBF net is utilized for clustering, Euclidean (L_2 -norm) distance is usually used in (3.8) because of its advantages such as rotation invariance and direct relation to within-class

variance. However, its computation is more complex than the alternatives such as L_1 -norm and L_∞ -norm distance. L_1 -norm has been considered in section 3.2.1. For the L_∞ -norm distance, it is the maximum of the absolute value of the coordinate difference of the two vectors in (3.8). The computing of these two distance metrics will be faster than the L_2 -norm distance. The impact to the performance of clustering will be discussed in the sequel.

Here the Euclidean metric is selected as the benchmark. The impact of the distance metrics to the quality of clustering depends on a number of factors. This includes the number of clusters, the dimension of the space, and the nature of the data in the space etc. To focus on the effect of distance metrics, we consider a two-cluster case, where three points are randomly placed in a N_i -dimensional space. Two of the points are randomly taken as the cluster centers, and the third as data point. The clustering result determined by the Euclidean distances between the data point and the two centers are assumed to be the correct result. Based on 10^4 trials, we compute the statistics produced by the clustering based on L_1 -norm and L_∞ -norm distance metrics benchmarked to the L_2 -norm metric. Misclassification rate and relative within-class variance are illustrated in Fig. 3.3. Misclassification rate is the fraction of ‘incorrect’ clustering trials from alternative metrics against the results by using Euclidean distance. The relative within-class variance is the ratio of within-class variance for the cluster assignments given by the alternative distance metrics, divided by the variance from the clustering result obtained by using Euclidean distance.

From Fig. 3.3, we observe that the misclassification rate is monotonically increasing with the increasing data dimension. For the 1-norm distance, the misclassification rate saturates at about 10%. For the L_∞ -norm distance, the error rate approaches to 50% with the increase in data dimension. This phenomenon indicates that L_∞ -norm is not better than just randomly assigning points to clusters for very large data dimension. For relative variance, both cases of L_1 -norm and L_∞ -norm metrics appear approaching to 1 with an increase in data dimension. But the deviation of L_1 -norm metric is better than L_∞ -norm metric for low dimension cases. Thus L_1 -norm metric is selected in (3.8) for practical implementation in this research.

Fig. 3.3. (a) Misclassification rates against data dimension; (b) Relative variance values against data dimension. Circles denote the values from L_1 -norm metric; diamonds denote the values from L_∞ -norm metric.

3.3 Network Architecture

The proposed neural network is constellated with fully-connected multiple layers of neurons as shown in Fig. 3.1. The network is now applied to pattern clustering and recognition. Given a dataset, each data-point will be encoded by using (3.1) to a multidimensional spike-time vector as the network input. This can be taken as a mapping from dataset space to the space spanned by spike-time vectors. The delay pattern of synapses to each neuron v_j is assigned as a reflection of the corresponding cluster center. Equation (3.8) shows that the firing time of v_j reflects a functional relationship (RBF) of the distance between the input vector and cluster center reflected in the delay pattern. Here, the neuron v_j can be considered as an RBF unit or RBF neuron. Furthermore, these synaptic delays can be adjusted so that a neuron in layer J fires earlier than others for data-points from a single cluster. That means this neuron matches the center of the cluster. Winner-takes-all competition can be employed for the final clustering. An ART-like learning algorithm is proposed for this pulsed neural network to implement unsupervised clustering in the ensuing subsection.

3.4 Unsupervised Clustering Algorithm

3.4.1 Learning algorithm

In this section, we propose an unsupervised learning algorithm for adjusting the synaptic delays to match the optimal cluster centers. Assume a dataset is given as

$$\mathbf{X} = \{ \mathbf{X}_k : \mathbf{X}_k = [x_1^k, x_2^k, \dots, x_{N_i}^k]^T, \mathbf{X}_k \in [0,1]^{N_i}, k = 1, 2, \dots, M \}, \quad (3.12)$$

where N_i is the dimension of data-point; M is the size of the dataset. The learning algorithm is listed in the following procedures.

Algorithm 3.1

Step 1. With $k=1$, the network and parameters are initialized as follows:

$T_{in} = 1$; $a=1$; The number of neurons in layer I is N_i ; the initial number of neurons in layer J is $N_j = 1$, the synaptic delays from neurons of layer I is set as

$$c_{iN_j} = T_{in} - (T_{in} - ax_i^k) = ax_i^k, \quad i = 1, \dots, N_i. \quad (3.13)$$

To improve noise rejection, we introduce a score ζ_j . Set $\zeta_1 = 1$ for v_1 .

Step 2. Set $k=k+1$. Code X_k to input spike-time vector using (3.1), and compute the firing times t_j of neuron v_j , $j = 1, \dots, N_j$.

Let α be a preset constant as a firing time boundary. If $\min_{j=1, \dots, N_j} (t_j) > \alpha$, set $N_j = N_j + 1$ and

add a new neuron v_{N_j} to layer J. The delays from layer I to this new neuron are assigned as

(3.13). Set $\zeta_{N_j} = 1$ for the new neuron v_{N_j} . If $\min_{j=1, \dots, N_j} (t_j) \leq \alpha$, adjust the delays to the neuron

v_m which fires at time $\min_{j=1, \dots, N_j} (t_j)$ as follows:

$$\Delta c_{im} = \gamma_0 ((T_{in} - (T_{in} - ax_i^k)) - c_{im}) = \gamma_0 (ax_i^k - c_{im}), \quad (3.14)$$

$$c_{im} = c_{im} + \Delta c_{im}, \quad (3.15)$$

where γ_0 is a preset constant learning rate.

Step 3. Calculate the score ζ_j of each v_j as follows:

If $j_k = \arg(\min_j (t_j))$, ζ_{j_k} is increased by one. Keep other ζ_j unchanged.

Step 4. If $k < M$, go to step 2. If $k = M$, delete those neurons v_j with $\zeta_j < \bar{\zeta}_j$. $\bar{\zeta}_j$ is a preset criteria. Perform cluster-merging (will be explained in section 3.5.3). Go to Step 5.

Step 5. If the number of learning epochs reaches a preset limit Π , the learning is terminated. If the plastic synaptic delays converge, the learning is ended. Otherwise, set $k=0$ and go to step 2.

After the optimal synaptic delay patterns are obtained, data points for retrieval process can be input to the network for clustering. A data point is classified to the cluster whose representative neuron in layer J fires earliest.

In this algorithm, the delay patterns linking to neurons in layer J reflect the centers of clusters. The firing time t_j of a neuron v_j , subject to an input data-point X_k , has been revealed as a measure of the distance between the data-point and v_j . The timing of the first-fired neuron, i.e. $\min_{j=1, \dots, N_j} (t_j)$ is utilized as a criterion for winner neuron selection. When $\min_{j=1, \dots, N_j} (t_j) > \alpha$, it means the distance between X_k and the nearest center is greater than the limit α . This is equivalent to a measure of the predefined radius of clusters, i.e. X_k is not within the boundary of any existing cluster. Thus a new RBF neuron is added with the linking delay pattern assigned as the reflection of the center of a new cluster. If $\min_{j=1, \dots, N_j} (t_j) \leq \alpha$, the

synaptic delay linking to neuron v_j is adjusted towards X_k . In this way, the expected cluster centers can be located automatically.

3.4.2 Discussion about the convergency

Consider Algorithm 3.1. The learning rule (3.14) for the within-cluster learning will be discussed in this subsection. For a cluster center represented by neuron v_j , (3.14) is rewritten as follows for discussing the convergence of the synaptic delays of v_j .

$$c_{ij}(h+1) - c_{ij}(h) = \gamma_0(ax_i(h+1) - c_{ij}(h)), \quad (3.16)$$

i.e.

$$c_{ij}(h+1) = (1 - \gamma_0)c_{ij}(h) + \gamma_0(ax_i(h+1)), \quad (3.17)$$

where h is the index of the iterative step; $x_i(h)$ is the i^{th} component of the input data of iteration h belonging to the cluster centered by neuron v_j .

Assume $c_{ij}(0)$ is given by the initialization. When $h=0$, we have

$$c_{ij}(1) = (1 - \gamma_0)c_{ij}(0) + \gamma_0(ax_i(1)). \quad (3.18)$$

By iterative substitutions, we can derive

$$c_{ij}(h) = (1 - \gamma_0)^h c_{ij}(0) + \gamma_0 \sum_{l=1}^h (1 - \gamma_0)^{l-1} (ax_i(l)). \quad (3.19)$$

When $h \rightarrow +\infty$, the first item of the right side of (3.19) approaches to 0 as $0 < \gamma_0 < 1$.

Consider the second item. It is not difficult to derive

$$\lim_{h \rightarrow +\infty} (\gamma_0 \sum_{l=1}^h (1 - \gamma_0)^{l-1}) = 1. \quad (3.20)$$

Therefore, the second item of the right side of (3.19) is equivalent to a weighted average value of $ax_i(l)$ for all l . That is to say the synaptic delay $c_{ij}(h)$ converges to this weighted average value.

3.5 Basic Clustering Examples

3.5.1. Cluster radius

In the proposed clustering algorithm, α is equivalent to a measure of the predefined radius of clusters. Because the firing time of a neuron v_j from to an input data-point X_k is a measure of the distance between the data-point and v_j , the criteria $\min_{j=1,\dots,N_j} (t_j) > \alpha$ for classifying new cluster means the minimum distance of the data-point X_k to all neurons v_j is greater than the cluster radius α , i.e. X_k is not within the boundary of any existing cluster. Thus a new RBF neuron is added as the center of a new cluster. The effect of α as a measure of cluster radius is illustrated in the following example.

Consider a set of $\mathbf{R}^2$ data points $X_{in} \in [0,1]^2$ visualized in Fig. 3.4. The proposed RBF network is trained to perform clustering using two different α values of 0.7 and 0.35. r_0 is defined as in (3.5). Other parameters for the learning algorithm are $N_i = 2$, $\gamma_0 = 1/128$. The data and results are shown in Fig. 3.4. With $\alpha = 0.7$, Fig. 3.4(a) shows that all the data points are clustered into two categories. This means that two RBF neurons have been generated in the layer J with the proposed algorithm. These two RBF neurons reflect the two cluster centers, which are located at (0.36, 0.33) and (0.86, 0.81) respectively in the $\mathbf{R}^2$ plane. They

are shown as circle markers in Fig. 3.4(a). Moreover, the distance between these two centers is 0.693. With $\alpha = 0.35$, the same set of data points result in four categories as shown in Fig. 3.4(b) due to a smaller cluster radius being used. In this case, there are four RBF neurons being generated to reflect the four cluster centers which are located at (0.10, 0.53), (0.22, 0.20), (0.54, 0.13) and (0.86, 0.80) respectively. Similarly, they are shown as circle markers in the figure. Amongst these four centers, the distance between the centers 2 and 3 is the smallest, i.e. 0.328. Since the distance between the cluster centers is a relative measure of the cluster radius, we can conclude that the value of α determines the cluster radius. Fig. 3.5 shows the relationship of α to the number of clusters. It is clear from the figure that the number of clusters reduces as the value of α increases.

Fig. 3.4. Result of clustering for different values of α : (a) $\alpha = 0.7$; (b) $\alpha = 0.35$. Color circles denote the cluster centers reflected in the delay patterns of neurons in layer J. The data points are labeled with specific markers for each cluster.

Fig. 3.5. Relationship between α and the number of clusters in $\mathbf{R}^2$ example.

Fig. 3.6. Noisy case with $\alpha = 0.35$. (a) result with $\bar{\zeta}_j = 0$; (b) result with $\bar{\zeta}_j = 5$. Red circles denote the cluster centers; stars, squares and diamonds denote the noisy points.

3.5.2. Noisy case

To study the impact of noisy data on the clustering, some ambiguous data are added to the dataset and shown in Fig. 3.6. In this example, the signal-to-noise ratio is 6.18, and the noise data are uniformly distributed in the data space. If a strong de-noise action is perform, e.g.

$\bar{\zeta}_j = (M/N_j) \cdot 30\%$ ($\bar{\zeta}_j$ is defined in step 4 of algorithm 3.1), those ambiguous data will be rejected as noise as shown in Fig. 3.6 (b). Those noisy data denoted as stars, squares and diamonds are distinguished from clustered markers. On the other hand, if a weak or none de-noise action is given, e.g. $\bar{\zeta}_j = 0$, these points lead to new clusters, and their cluster centers (red circle markers) are generated as shown in Fig. 3.6(a).

Fig. 3.7. Redundant cluster centers. Circles denote the cluster centers.

3.5.3. Cluster merging

In some cases, multiple neurons may approach to the same cluster. This may occur occasionally in the process of the neuron generation in step 2 of algorithm 3.1. In this case, one neuron is generated at the edge of a cluster while another is at the far-end of the same

cluster. During training, they are ‘attracted’ by the data points in the same cluster. As shown in Fig. 3.7, the redundant centers 4 and 5 cause the desired cluster being separated into multiple clusters. In this case, cluster-merging could be performed as follows: At the end of each epoch in algorithm 3.1, we use each of the synaptic delay $\mathbf{c}_j = [c_{1j}, \dots, c_{ij}, \dots, c_{Nij}]^T$, $j = 1, \dots, N_j$ as input data for the network. If $t_{j'} = f(\|\mathbf{c}_{j'} - \mathbf{c}_j\|^p) \leq \alpha$ for $j \neq j'$, either neuron v_j or $v_{j'}$ with smaller ζ_j will be pruned.

3.5.4. Clustering of realistic data

The Fisher’s Iris dataset [68] has been widely used for examples in cluster analysis. This is a problem with four-dimensional inputs and three output clusters. The sepal length, sepal width, petal length, and petal width are measured in millimeters on fifty iris specimens from each of the three species, namely iris setosa, iris versicolor, and iris virginica. In this section, we apply the settings of r_0 in both (3.5) and (3.9) for observation. For clarity, we call r_0 in (3.5) as L_2 -norm r_0 , and r_0 in (3.9) as L_1 -norm r_0 .

The input data should be normalized to satisfy (3.12). Parameters for the learning algorithm with L_2 -norm r_0 are heuristically selected as follows $N_i = 4$, $\alpha = 0.35$, $\gamma_0 = 1/128$, $\bar{\zeta}_j = 5$. For the case of L_1 -norm r_0 , these parameters are kept unchanged except $\alpha = 0.4$. The percentage of correct classification as compared with MATLAB Toolbox based k -means and real-valued self-organizing map (SOM) are given in Table 3.1. From the table, it is observed that the proposed pulsed-based RBF network yields 90.00% correct classification for both

cases with L_1 - and L_2 - norm r_0 . In addition, it is also observed that the k -means and SOM yield 88.67% and 87.33% correct classification respectively. Thus the performance of the proposed methods with either L_1 - or L_2 - norm r_0 is better than the conventional counterparts.

Table 3.1. Clustering of Fisher's Iris Dataset. ($k=3$ for the k -means; 3 Kohonen neurons for SOM.)

Method	Percentage of correct classification
Proposed method with L_1 - norm r_0	90.00% $\pm 0.67\%$
Proposed method with L_2 - norm r_0	90.00% $\pm 0.67\%$
k -means	88.67% $\pm 0.67\%$
SOM	87.33% $\pm 0.67\%$

3.6 Complex Clusters

The clustering methods based on L_1 - and L_2 - norm distance are limited to convex data set. From this point of view, the difference between the algorithm 3.1 and conventional methods such as k -means and SOM is not significant. One of the important advantages of the proposed neural model is that it can be extended to cope with concave data set. Let us observe a set of data points, which consists of two clusters as shown in Fig. 3.8. The figure

shows the clustering result from the pulsed neural network trained by the proposed algorithm 3.1. We observe that some of the data points (marked with '∇') belonging to the concave set are incorrectly clustered to the convex set. This is because the RBF neurons centering in the convex set are firing earlier than the RBF neuron centering in the concave set. In another words, these data are closer to the incorrect neuron than the correct neuron.

Fig. 3.8. Incorrect clustering result for concave clusters. Red circles denote the cluster centers. 'x' and '∇' markers denote different clusters.

Fig. 3.9. Network with lateral connections.

To overcome this problem, we use more RBF neurons with lateral connections, and decrease the radius of the clusters α . By using the algorithm 3.1, more RBF neurons will be generated and distributed properly to cover data points of all the clusters. The cluster centers stored in the pre-synaptic weights are considered as local centers instead of final cluster centers. Based on the assumption that components belonging to the same cluster have certain density of data points among each other, we establish and strengthen lateral connections between the RBF neurons as shown in Fig. 3.9 once data points between them appear. Given an input training data, the timing of first- and second-fired output neuron will determine the establishment of their lateral connection. For pulse based learning, the rank order of firing neurons v_j can be conveniently obtained by utilizing temporal difference of pulses. For instance, if the neurons v_1 and v_2 have established the lateral connection, their mutual lateral strengths L_{12} and L_{21} will be strengthened during learning by adding an increment ΔL . In the retrieval phase, their individual dynamic potentials can be augmented from (3.2) as

$$S_1(t) = \sum_{i=1}^{N_i} \varepsilon(t - t_i - c_{i1}) + L_{21} u(t - t_{v2}), \quad (3.21)$$

$$S_2(t) = \sum_{i=1}^{N_i} \varepsilon(t - t_i - c_{i2}) + L_{12} u(t - t_{v1}), \quad (3.22)$$

where L_{12} and L_{21} are lateral strengths between v_1 and v_2 ; t_{v1} and t_{v2} are the firing times of v_1 and v_2 ; $u(t)$ is an unit step function and $\varepsilon(\cdot)$ is defined in (3.3).

Fig. 3.10. Clustering results using the network with lateral connections. Crosses denote the RBF neurons. Color circles denote the cluster indicator selected from each group of synchronized RBF neurons. (a) Result without noise; (b) Result with noise.

After this operation, those RBF neurons whose lateral connections are strong enough, i.e. $L_{12} = L_{21} \geq \theta$ (θ is the threshold in (3.4)), will synchronously be triggered by the augmented potentials when a data point of the corresponding cluster is retrieved. This phenomenon will not happen to neurons coding for different clusters due to the rareness of data points in between. Thus correct clustering can be implemented by selecting one neuron as a cluster indicator from each group of synchronized RBF neurons. Fig. 3.10(a) shows the clustering

results using the network with lateral connections. In the case of noisy data, where the signal to noise ratio is 8.33, this scheme can still give 100% correct clustering rate as shown in Fig. 3.10(b).

Fig. 3.11. Pulsed neural network for function approximation.

3.7 Supervised Learning for Nonlinear Function Approximation

In previous sections, we have investigated the pulsed RBF neural network for unsupervised clustering. In this section, we will extend this neural network for supervised learning schemes. As a typical application of supervised methods, nonlinear function approximation will be discussed to evaluate its performance.

3.7.1 Network model

The proposed pulse-based RBF neural network in Fig. 3.1 can be extended by adding a layer Y as shown in Fig. 3.11. Without lose of generality, we explain using only one neuron in

layer Y. Here, we consider two types of definitions of the layer Y. The first is based on winner-takes-all operation. The other is based on optimal interpolation. For both methods, the RBF neurons need to be determined by algorithm 3.1. A bias neuron v_0 with a constant time of pulse is added only for optimal interpolation method to improve its performance. Here we use an augmented input vector instead of only the original input vector as follows,

$$\mathbf{Z}_k = \begin{bmatrix} \mathbf{X}_k \\ y_k \end{bmatrix}, \quad (3.23)$$

where $\mathbf{X}_k$ is the input vector of the k^{th} training data; y_k is the desired output of this training data. The use of this augmented vector can generate a distribution of RBF neurons on the basis of a complete space by taking y into account. Once this phase of algorithm is completed, only the components corresponding to the original function variables are extract from each resulting vector to construct the respective RBF neuron in Fig. 3.11 so that the inputs of this extended network is still equivalent to the function variables. In the following subsections, winner-takes-all and optimal interpolation schemes are discussed for different type of nonlinear function approximations.

3.7.1.1 Winner-takes-all

This is a direct extension of the RBF network. Assume an input vector is fed to the network in Fig. 3.11. The first-fired RBF neuron v_w will inhibit all other RBF neurons v_j , $j \neq w$. This means that the output of y_1 is determined by the winner v_w only. This is known as winner-takes-all method. To simplify the process, we design the post-synaptic terminal from

layer J to layer Y so that the winner activates its related terminal at a constant time, say T_y .

Then the output pulse from neuron y_1 can be expressed as

$$y(t) = \delta(t - T_y - c_{wy}), \quad (3.24)$$

where c_{wy} denotes the delay from the post-synaptic terminal to the output neuron y_1 . $\delta(\cdot)$ is the unit impulse function. Here, the firing time of the output pulse is

$$t_y = T_y + c_{wy}. \quad (3.25)$$

The analog output can be decoded as follows,

$$c_{out} y = T_{out} - t_y, \quad (3.26)$$

where c_{out} and T_{out} are constant.

This network can be applied directly to those enumerative functions (e.g. XOR) by selecting its RBF neurons as the enumerative input vectors. For continuous functions, it is a piecewise approximation whose resolution depends on the number of the RBF neurons.

3.7.1.2 Optimal interpolation

In this case, the neurons in layer Y are designed as integrate-and-fire models. Assume the RBF neurons v_j , $j=1, \dots, N_j$, have been determined. After the pulses fired by neuron v_j arrive at the respective post-synaptic terminals, the post-synaptic potentials (PSP) are activated. For the neuron in layer Y, the PSP is computed as follows ($j=0$ is for bias),

$$y(t) = \sum_{j=0}^{N_j} w_j \xi(t - t_j) . \quad (3.27)$$

where the weight values w_j are the synaptic efficacies, $\xi(t)$ is the response function of PSP.

Once the integrated potential $y(t)$ reaches the threshold $\theta_y(t)$, the output neuron y_1 will fire

a pulse at $t_y = \arg_t(y(t) = \theta_y(t))$. The firing time t_y can be considered as a function of the

input set X . From section 3.2.1, we know $t_j \in [2T_{in}, 3T_{in}]$. To manage the temporal

computation conveniently, we define the threshold $\theta_y(t)$ as follows so that t_y is confined in

the trigger zone $t_y \in [T, T_y]$, where $T = 2T_{in}$,

$$\theta_y(t) = \begin{cases} \infty & 0 < t < T \\ T_{in} - t & T \leq t \leq T_y. \end{cases} \quad (3.28)$$

Moreover, we assign $T_y = 3T_{in}$.

Let $\xi(t)$ be piecewise linear in the trigger zone of neuron y_1 ,

$$\xi(t) = \begin{cases} 0 & t < 0 \\ t & 0 \leq t. \end{cases} \quad (3.29)$$

We have the following equation at time t_y ,

$$\theta_y = \sum_{j=0}^{N_j} w_j \xi(t_y - t_j) = \sum_{j=0}^{N_j} w_j (t_y - t_j) = t_y \sum_{j=0}^{N_j} w_j - \sum_{j=0}^{N_j} w_j t_j. \quad (3.30)$$

This is equivalent to

$$t_y \sum_{j=0}^{N_j} w_j = \theta_y + \sum_{j=0}^{N_j} w_j t_j. \quad (3.31)$$

Consider the above equation. As the output of an RBF unit, t_j can be written approximately as $t_j = \phi(\|X_k - c_j\|)$, where X_k is an input data point, and c_j is equivalent to the j^{th} center vector of the dataset reflected in the synaptic delay pattern linking to the corresponding RBF neuron. Given an input data set X and the corresponding desired temporally coded output set $\bar{t}_d = F(X)$, the interpolation matrix of the function F can be obtained based on (3.31) and those RBFs' firing times $t_j = \phi(\|X_k - c_j\|)$, $j = 1, \dots, N_j$, which can be computed by the network in Fig. 3.11. Thus the pulsed neural network is equivalent to the RBF neural network of Broomhead and Lowe [69]. The weight values w_j usually can be resolved by using pseudo inverse of the interpolation matrix. Here we employ another option, the well-known gradient descent method, to show the application of temporal representation in the derivation of the learning algorithm so that the overall neural network acts consistently in bio-inspired manner.

3.7.2 Gradient descent learning

The gradient descent learning algorithm has been widely used for real-valued neural network training [40]. For pulsed neural networks, most researchers prefer biologically plausible algorithms such as Hebbian [15, 37]. In this section, we will derive the gradient descent learning algorithm based on the temporal representation of pulsed neural network. These derivations utilize the basic idea of gradient descent learning in [39, 40]. As compared with

those multilayer BP network, our RBF network has only one gradient decent layer. So it could avoid the local minimum as compared with the multilayer BP network. In contrast to the Bohte's work [39], the proposed designs of threshold and PSP activation function simplify the computations and make the model more convenient for hardware implementations. Moreover, the model can circumvent the uncertainty incurred by the nonlinear activation function in Bohte's algorithm as its computation is equivalent to linear interpolation as explained in section 3.7.1.2.

For the k^{th} input-output data-point in a dataset coded in temporal representation, a squared error measure is defined as

$$E_k = \frac{1}{2} (t_d^k - t_y^k)^2. \quad (3.32)$$

where t_d^k is the desired output, and t_y^k is the actual network output firing time.

For simplicity of presentation, the above equation is rewritten as

$$E_k = \frac{1}{2} (t_d - t_y)^2, \quad (3.33)$$

where t_d is the desired output, and t_y is the actual network output firing time. Then the weight values in Fig. 3.11 can be updated using the following gradient descent rule,

$$\Delta W(l) = -\eta \frac{\partial E_k}{\partial W}, \quad (3.34)$$

$$W(l+1) = W(l) + \Delta W(l), \quad (3.35)$$

where $W = [w_1, \dots, w_{N_j}]^T$, and l is the step number, η is a learning factor.

The gradient vector can be derived from (3.33) as

$$\frac{\partial E_k}{\partial W} = (t_y - t_d) \frac{\partial t_y}{\partial W} = (t_y - t_d) \frac{\partial t_y}{\partial y} \frac{\partial y}{\partial W}, \quad (3.36)$$

where y is the integrated potential of the output neuron y_1 .

Fig. 3.12. Relationship between Δt_y and Δy .

To determine $\frac{\partial y}{\partial W}$ and $\frac{\partial t_y}{\partial y}$, we considering (3.27) and (3.29), and obtain

$$y(t) = \sum_{j=0}^{N_j} w_j(t - t_j). \quad (3.37)$$

At time $t = t_y$, performing partial differentiation to (3.37) yields

$$\frac{\partial y}{\partial W} = t_y \underbrace{[1, \dots, 1]^T}_{N_j} - (\bar{t}_j), \quad (3.38)$$

where $\bar{t}_j = [t_1, \dots, t_{N_j}]^T$.

To derive the expression of $\frac{\partial t_y}{\partial y}$ in terms of W , we consider Fig. 3.12, which illustrates the relationship between Δt_y and Δy in a differential interval around t_y . The solid line denotes y as the function of time t linearized in the differential interval. The dashed line denotes the variant y due to the increment Δt_y . Let the gradient of the solid line be λ , i.e.

$$\frac{\partial y}{\partial t} = \lambda, \quad (3.39)$$

then we have the following geometric relationship with taking (3.28) into account,

$$|\Delta y| = (1 + \lambda) |\Delta t_y|. \quad (3.40)$$

It is also observed that $\frac{\Delta t_y}{\Delta y}$ is negative. Thus we can infer that

$$\frac{\partial t_y}{\partial y} = \frac{-1}{1 + \lambda} = \frac{-1}{1 + \frac{\partial y}{\partial t}}. \quad (3.41)$$

Substituting (3.37) into (3.41) yields

$$\frac{\partial t_y}{\partial y} = \frac{-1}{1 + \sum_{j=0}^{N_j} w_j}. \quad (3.42)$$

Substituting (3.38) and (3.42) into (3.36), we can obtain the solution of $\frac{\partial E_k}{\partial W}$.

As the overall error measure of the dataset is

$$E = \sum_{k=1}^M E_k, \quad (3.43)$$

the batch gradient descent rule is

$$\Delta W(l) = -\eta \frac{\partial E}{\partial W} = -\eta \sum_{k=1}^M \frac{\partial E_k}{\partial W} \quad (3.44)$$

and

$$W(l+1) = W(l) + \Delta W(l). \quad (3.45)$$

In some pulse-based neural models that are based on nonlinear activation functions [18, 39], BP algorithm may have surge phenomenal in the training procedure due to its derivative computation. In contrast, our design of linear activation function in (3.29) could avoid the surge of differential PSP due to its constant incremental rate. The complete learning algorithm for the network in Fig. 3.11 includes two phases. In phase one, the neurons (their synaptic delays) in layer J are determined using algorithm 3.1. In phase two, values of weights between layer J and layer Y are determined using supervised training rule given in (3.45). The performance of this scheme for nonlinear function approximation will be evaluated using an example in the following section.

3.7.3 Example of nonlinear function approximation

In this section, we verify the performance of the proposed pulsed neural network and learning algorithms by using an example. To make a fair comparison with counterpart of pulsed neural network, we employ the example function in [60] as follow,

$$f(x) = (2x - 1.6)^3 - 2x + 4. \quad (3.46)$$

For this study, 150 points of input-output data, which are sampled randomly in the interval $x \in [0, 1.5]$, are applied to the network learning. Another set of 150 data points are also obtained in the same way for verification.

The first phase is for determining the RBF neurons. Training data are organized as in (3.23). Nine RBF neurons are generated with $\alpha = 0.2$. Their normalized center vectors in terms of synaptic delays are given in Table 3.2. In the second phase, we approximate the function in both ways of winner-takes-all and optimal interpolation. The final approximated function values are obtained by decoding the firing times of the network output neuron. Fig. 3.13 (a) and (b) illustrate the results of these two cases respectively. For the optimal interpolation, the mean absolute error (MAE) trend during the gradient descent learning is shown in Fig. 3.14. Table 3.3 gives the weight values w_j for the optimal interpolation obtained by gradient descent learning. In [60], they approximate the same function using its own pulse-based neural model. Its scheme is similar to our winner-takes-all method except its pulsed neurons are selected based on constant interval of the variable x . With the same number of pulse based neurons, Table 3.4 lists their mean absolute errors for the verification data set. From the results, we observe that its approximation error is larger than our winner-takes-all method.

Moreover, our optimal interpolation has the best accuracy among all the methods. Table 3.4 lists their mean absolute errors for the verification data set.

Table 3.2. RBF neurons

Normalized X	0.0014	0.0400	0.0933	0.1667	0.3600	0.5533	0.7467	0.9200	0.9867
Normalized Y	0.0049	0.1912	0.3948	0.5898	0.7488	0.6344	0.5516	0.7544	0.9509

Table 3.3. Weight values w_j

w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
0.1519	-2.2879	-0.6612	0.7360	1.8621	1.0374	0.0001	-1.5590	-1.2381	2.4698

Table 3.4. MAE for various function approximation methods

	Literature [60]	Winner-takes-all	Optimal interpolation
MAE	0.1697	0.1403	0.0534

Fig. 3.13. Result of nonlinear function approximation. (a) winner-takes-all scheme. (b) optimal interpolation scheme. Blue dots denote the function value of verification data. Red ‘x’ markers denote the estimated value by the pulsed neural network.

Fig. 3.14. Mean absolute error (MAE) during gradient descent learning.

3.8 Conclusions

In this chapter, we propose and investigate a pulse-based RBF neural model. Different from the conventional RBF networks, the proposed pulse-based RBF network utilizes the synchronism of temporal pulses generated by model neurons to perform RBF computations. Based on a new learning algorithm, it gives comparable performance when benchmark with the conventional counterparts, such as k -means and SOM in clustering applications. Utilizing the rank order of neurons' output pulses, the pulsed neural network is extended to deal with some complex clustering problems such as processing of concave clusters which is difficult if k -means and SOM are used. The temporal pulse based paradigm is also extended to nonlinear function approximation applications [60]. It outperforms the benchmarked pulse-based neural networks reported in recent literature. Some salient points of the proposed model are listed as below:

- Neurobiological phenomenon, i.e. integrate-and-fire, is utilized to perform RBF computation. The computation based on the synchrony of temporal pulses reflects the parallelism characteristics of the bio-inspired design.
- It can deal with concave clusters by employing rank order of neurons' firing times to establish lateral connection among local centers.
- The RBF essence avoids convergency issues of well-known BP algorithm for nonlinear function approximation.
- Various implementation-oriented designs, including winner selection by rank order of pulses and multiplierless 1-norm distance, may yield more compact neural circuits for the full parallelism on programmable digital platform such as FPGA.

In the next chapter, we will investigate the design and implementation of the neural circuits based on these advantages.

Chapter 4

On-Chip Implementations of the Pulsed RBF Neural Circuits and Their Applications

4.1. Introduction

In chapter 3, the pulsed RBF neural networks and the related learning algorithms have been developed. As summarized in section 3.8, these algorithms are designed for real applications in mind. In this chapter, we will investigate their hardware implementations and present their practical applications.

Most existing applications of artificial neural networks are implemented using software simulators. This limits one of the most important advantages of neural networks, namely the massive parallelism. Hardware implementation is the direct way for fully parallel computation, and VLSI designers and neuromorphics researchers have put in a lot of effort [63-66, 70] in this topic. There are two basic options, namely the analogue and digital techniques, for neural-circuits designers. Analogue technique is attractive for reasons of compactness, potential speed and low power consumption. However, its robustness to noise and design flexibility are far from satisfaction. Moreover, the customized IC design is costly and restricts its availability for user-oriented applications. From this point of view,

inexpensive programmable digital technique is more attractive for researchers. Due to the frequent upgrading, the programmable digital technique is hopeful to circumvent its disadvantages on compactness and power consumption. In [70], Marchesi et al proposed a scheme for hardware neural circuits. To improve the compactness, they utilized shift registers to approximate multipliers in a real value neural network. However, their hardware circuits are only available for the forward computation. The network learning is still conducted using offline simulator. Hikawa [66] proposes a frequency-based hardware scheme with on-chip learning. Both his neural model and implementation scheme are based on back-propagation (BP) networks, which are prone to local minimum [67]. In this research, we design hardware circuits for the proposed pulsed neural network using field programmable gate array (FPGA). The merits of the pulsed RBF model listed in section 3.8 are realized in the circuit design. Firstly, the computation based on the synchrony of temporal pulses provides the parallelism of neural circuits. Secondly, operations for rank order of neurons' firing times are suitable for implementation using digital logic circuits. Thirdly, the usage of multipliers that is generally acknowledged as a bottleneck of digital circuits, will be drastically reduced. Fourthly, the RBF essence avoids convergent issues of BP algorithm.

Owing to its fast processing, the design for the hardware implementation of the pulsed RBF network provides a basis for various practical applications. By systematic extension, these pulse-based neural circuits will be validated experimentally for several real-time applications in this chapter. Firstly, the pulsed RBF network is extended to an on-chip nonlinear look-up table for friction estimation of a precision linear stage. Secondly, proposed clustering algorithm is enhanced to implement object-tracking in video recording. Finally, the proposed parallelism-oriented RBF neural network is combined with a parallel GA to perform a supervised learning algorithm for satellite image processing.

Fig. 4.1. Block diagram of the hardware system. The blocks framed with dashed lines are circuits for network learning. Other circuits perform forward computations.

4.2. Implementation of Pulse-Based RBF Model with Unsupervised learning

Fig. 4.1 shows the block diagram of the hardware system that has been developed. Without loss of generality, we show a system with two input channels and two neurons of v_j , $j = 1, 2$, as an example. For the actual system, the number of channels can be extended depending on the capacity of the FPGA. As shown in Fig. 4.1, the system consists of forward network and the network-learning computation.

The circuits for the forward network of the pulse-based RBF model includes synapse units and circuits of neuron v_j . Each synapse unit is designed to accept input pulse and compute

the excited time-dependent PSP defined in (3.3). The circuit of each neuron v_j is designed to implement the integrate-and-fire model as described in section 3.2.

The circuits for network-learning computation, which are framed with dashed lines in Fig. 4.1, are designed to implement the arithmetic and logic operations of the learning algorithm given in section 3.4. This portion includes input registers ax_i , computing units for updating RBF center c_{ij} , circuit for comparison of cluster radius α and the firing time of the winner neuron, unmasking logic circuit and logic circuit for enabling/disabling of synapse update. Amongst these components, register ax_i is used to convert the timing of the input pulse to the digital value based on encoding/decoding rule given in (3.1). Here, it is a decoding operation because we assume the input signals are temporally coded. The resultant ax_i value will be used either as an operand for updating synaptic delay c_{ij} with learning rule (3.14) or as a value for initializing the synaptic delay connected to a newly added neuron v_j as (3.13). It should be noted that the possible maximum number of neurons must be preset with all related circuits solidified initially. In other words, the phrase “add new neuron” in the learning algorithm (step 2 of algorithm 3.1) means only to unmask those existing neural circuits. Thus the unmasking logic circuit is in the diagram. For the developed system, it is designed to mask or unmask the system clock of the circuits. As described in the step 2 of algorithm 3.1, different results of the inequality $\min_{j=1, \dots, N_j} (t_j) > \alpha$ will lead to different operation. If $\min_{j=1, \dots, N_j} (t_j) > \alpha$ is true, a new neuron v_j will be added, and the operation for updating the existing synaptic delays should be disabled. Otherwise, delay values of the synapses connected to the winner neuron will be updated. Therefore, a logic circuit to

enable/disable the updating operation is designed. The detailed circuit designs will be described in the following subsections.

Fig. 4.2. Circuits for synapse unit, integrate-and-fire and register of ax_i .

4.2.1. Synapse unit

A key function of a single synaptic model is the computation of its time-dependent postsynaptic potential (PSP). From the pulsed neural model (3.2)-(3.3) and L_1 -norm based r_0 in (3.9), the PSP excited by a single input pulse can be obtained as

$$PSP_{ij}(t) = \begin{cases} t - (T_{in} + T_{in} - (t_i + c_{ij})) & t_i + c_{ij} \leq T_{in} \\ t - (t_i + c_{ij}) & t_i + c_{ij} > T_{in} \end{cases} \quad (4.1)$$

The synaptic unit in block A of Fig. 4.2 shows the circuit realization that computes the PSP using an up/down counter. The input signals of the synaptic unit are temporally coded pulses. An external cycle control signal is applied to provide a timing reference for the neural dynamics excited by the input pulses. Here, the term cycle refers to the period of activity for a pulse-coded data point to go through the neural network. Related control signals are logicized to implement the function of the synaptic model. Fig. 4.3 shows the logic waveforms. At the beginning of a cycle, the initial synaptic delay c_{ij} is loaded into the counter from the corresponding “delay-value register”. The input temporal pulse will enable the clock signal so that the counter starts to count. The counter’s overflow signal and the cycle control signal are combined to control the up/down operation of the counter. Note that the counter value represents the practical PSP value and can be exported as PSP only if $counter\ value > 0$ and $up/down=up$. In Fig. 4.3, the first cycle demonstrates the case of $t_i + c_{ij} \leq T_{in}$. The second cycle demonstrates the case of $t_i + c_{ij} > T_{in}$. We can observe that the output of the counter is exactly the numerical results given in (4.1).

Fig. 4.3. Signal waveforms of synaptic counter.

4.2.2. Integrate-and-fire circuits

The integrate-and-fire (IAF) model is designed as a digital circuit as shown in block B of Fig. 4.2. The IAF unit receives digital PSP values from related synapses and sums them up. Once the sum exceeds the preset threshold, the circuit will fire an output pulse. In winner-takes-all mode, this output pulse will drive all the postsynaptic terminals into refractory state by resetting all the synaptic counters.

Fig. 4.4. Masking/unmasking circuit and comparison circuit of α and firing time of the winner neuron.

4.2.3. Circuits for network learning

As depicted in the block diagram (Fig. 4.1), the circuits in the framed blocks are designed to implement the learning algorithm developed in section 3.4. The masking/unmasking logic mentioned earlier in this is described in details in framed block of the masking/unmasking circuit in Fig. 4.4. At the beginning of the very first learning cycle, only the clock signal of neuron v_1 , i.e. clk_v_1 , is unmasked. If the firing time of neuron v_1 exceeds α , an unmasking signal will be produced to trigger the first flip-flop in the mask/unmask circuit. The rising edge of 'Q' will signal the delay-value registers (block A of Fig. 4.2) related to neuron v_1 to load the values of ax_i , $i = 1, 2, \dots, N_i$, from register circuits of ax_i values. Block C in Fig. 4.2 shows one of these register circuits. This is actually a countdown timer controlled by the cycle control signal and the input pulse. At the beginning of each learning cycle, the timer is signaled to load the value of T_{in} as its initial value and start counting down. An input pulse, which reaches at t_i , will stop the timer. At this time, the value of the timer is $T_{in} - t_i = ax_i$, which is the decoding form of (3.1). The registered value of ax_i will be used either for initialization of newly added neuron or for adjustment of synaptic delays. It should be noted that the synaptic delay values are random during the very first cycle. Thereafter, the registered values of ax_i become available for network learning.

Next, we describe the circuit that performs the comparison of α and the firing time of the winner neuron as shown in block A of Fig. 4.4. A timer is used to count the value of firing time of the winner of neuron v_j , $j = 1, 2, \dots, N_j$. In each learning cycle, the timer is started from reset state by the rising edge of the cycle control signal, and is stopped by the first arrival pulse fired by the winner neuron. The value stored in the timer, which reflects

$\min_{j=1,\dots,N_j} (t_j)$ mentioned in the learning algorithm, is compared with α . If $\min_{j=1,\dots,N_j} (t_j) > \alpha$, the output of the comparator is used to unmask a neuron. Meanwhile it disables the circuits for the adjustment of synaptic delays c_{ij} . Otherwise, it enables the update computation of corresponding c_{ij} without unmasking any neuron.

The update of synaptic delays is designed as shown in Fig. 4.5. Assume the winner neuron v_j fires at time t_j . The output pulse will require the registers of synaptic delays connected to v_j to export the delay values onto the data bus. Once the data are ready, the corresponding computing units for updating c_{ij} are signaled to read in the data of delay values respectively. These units also read in the value of the corresponding ax_i . Then the new c_{ij} were computed and exported onto the data bus. And the registers of synaptic delays connected to v_j are signaled to read in these updated values of c_{ij} respectively. In each computing unit for updating c_{ij} , a right shift register instead of a multiplier is used to implement the multiplication of learning rate γ_0 in (3.14).

With the cycle control signal, the temporally encoded pulses are input to the circuits cycle by cycle. The learning algorithm proposed in section 3.4 can be implemented with the logical and numerical computation performed by the proposed neural circuits.

Fig. 4.5. Circuit diagram for updating the synaptic delays.

4.3. Application to Friction Estimation Based on Nonlinear Look-up Table

4.3.1. Pulsed RBF neural circuit as a nonlinear look-up table for friction estimation

Precision linear stage is an accurate positioning system that uses direct linear motor drive. It is increasingly required in advanced factory automation for the microelectronics, photonics and biomedical industries. A significant operating problem of such system is the friction between the sliding part (i.e. the forcer) and the stationary magnet way of the linear machine. The degrading effect is more pronounced for relatively short move at a low speed, say less than 0.05 m/s. In [74], the degrading phenomenon due to the friction has been studied, and a dynamic model to estimate the friction for compensation and improvement of system performance has been developed. The parameters of their friction model were estimated based on offline experimental data. Fig. 4.7 shows the friction-speed map obtained using offline experimental data. Consequently, the efficiency of this parametric method is limited as the parameters may be varied over time caused by factors such as wear and tear, temperature etc. In this section, we apply the proposed pulse-based circuits to perform on-line estimation of the friction so that the system can be adaptive to the time-varying factor.

Fig. 4.6. Experimental linear stage system.

Fig. 4.7. Experimental friction-speed curve.

An experimental set-up is shown in Fig. 4.6. According to [74], the friction F_r and the velocity v of the forcer have a nonlinear dynamic relationship. Assume this dynamics can be expressed as the following one-step-ahead estimation function,

$$F_r(t+1) = f(F_r(t), v(t)). \quad (4.2)$$

To approximate this mapping with the pulsed neural circuits, we formulate the training data vector as

$$\mathbf{X}_k = [v(t), F_r(t), F_r(t+1)]^T. \quad (4.3)$$

where the symbol $\mathbf{X}_k$ is used to be consistent with the definition of training data in (3.12).

A three-input neural network is constructed for training based on the circuits described in section 4.2. In the retrieval process, the components of the data vector $[v(t), F_r(t)]^T$ are input to the corresponding input channels of the network to determine the winner neuron.

Consequently, the value stored in the winner's synaptic delay corresponding to the component $F_r(t+1)$, is just the result of the one-step-ahead estimation. This method is of a nonlinear look-up table.

To acquire the on-line training data, the dynamic model of the linear motor is utilized. Fig. 4.8 shows the simplified block diagram of the linear motor system. In the figure, i_q is the armature current, v and M_a are the velocity and total mass of the moving part (i.e. the forcer and the camera load). K_f is the force constant. Detailed description of the system parameters is summarized in Table 4.1. At time t , $F_r(t)$ can be calculated as

$$F_r(t) = K_f i_q(t) - a(t)M_a, \quad (4.4)$$

where $a(t)$ is the measurable acceleration of the forcer. $F_r(t)$ and $F_r(t-1)$ can be calculated on-line to formulate the training data in (4.3) for the pulsed neural network.

Fig. 4.8. Block diagram of the linear motor system.

Table 4.1. Linear motor's specifications

Nominal Specifications	Symbols	Values	Units
Electrical Resistance	R	6.1	Ω (line-to-line)
Back EMF constant	k_e	13.8	$V_{pk}/msec^{-1}$ (line-to-line)
Force Constant	K_f	16.8	N/A_{rms}
Peak Current	I_p	7.1	A_{rms}
Mass (Camera, Platform)	M_a	5.8	Kg
Pole pair	p	1	N.A.
Inductance (Line to Line)	L_{line}	1.3	mH

4.3.2. Experimental results

In the experiment, the cycle time T of the neural computation (see section 4.2.1) is set at 2 msec to be consistent with the sampling period of the servo system of linear motor. This also means the neural circuits can complete 500 epochs of network learning in one second. As the network is adaptive, it is capable of updating the friction estimator in real time through network learning. As presented in the section 4.3.1, measurable variables, including the velocity $v(t)$ and the armature current i_q , are acquired on-line. At each sampling step, the instantaneous friction $F_r(t)$ can be calculated using (4.4), and the training data can be formatted as (4.3). A three-input neural network is constructed for the training data vector $[v(t), F_r(t), F_r(t+1)]^T$. The size of one batch of training data is 100. Fig. 4.9 shows two sets of primitive data of velocity and motor current acquired. Fig. 4.10 shows the instantaneous relationship between the velocity and the friction calculated with (4.4) based on the primitive data. As an example, Fig. 4.11 illustrates the distribution of the RBF neurons generated after training. The mean absolute percentage errors (MAPE) for verification of the estimated

friction are listed in Table 4.2 with respect to various clustering radius α and the number of RBF neurons generated. It is observed that the accuracy of the estimator is improved with decreasing α . This is due to the fact that the local coverage of each RBF neuron becomes finer with smaller clustering radius. However, this is at the expense of more RBF neurons being generated to cover the data space, and more hardware resource for implementation. In the system with 10 RBF neurons, a total of 279K gates is used on the FPGA chip, where registers and counters are designed as 10-bit units. To evaluate the adaptive performance of the proposed method, the parametric model given in [74] is selected as the counterpart for comparison. In case 1, we can observe that the accuracy of the neural circuits is comparable to the parametric model when the number of neurons increased to 10. In case 2, the ambient temperature is increased by over 10 °C. We observe that the MAPE of the parametric model is increased by about 10%. This is because the parametric model was identified offline, and it is unable to adapt to the varying operating condition. In contrast, there is no significant impact to the accuracy of proposed method due to its adaptive learning.

Fig 4.9. Acquired data of velocity and motor current.

Fig 4.10. Instantaneous relationship between the velocity and the calculated friction.

Fig 4.11. Distribution of the RBF neurons after network training based on the training data set. Red circles denote the RBF neurons. Blue cross marks denote the points of training data.

Table 4.2. Results for friction estimation using different methods and parameters

method category	model parameters		Case 1	Case 2	Hardware resource (equivalent gate number)
	clustering radius	number of RBF neurons	MAPE	MAPE	
proposed method	$\alpha = 0.33$	6	17.08%	16.67%	153k
	$\alpha = 0.28$	7	12.39%	12.71%	177k
	$\alpha = 0.25$	8	9.58%	10.10%	213k
	$\alpha = 0.22$	9	9.06%	8.85%	242k
	$\alpha = 0.20$	10	8.54%	8.49%	279k
parametric model in [74]	parameters of LuGre friction model from identification in [75]		8.58%	18.33%	N.A.

4.4. Application to Object-Tracking Based on Clustering

To automatically track multiple moving objects in a scene, conventional vision algorithms have to sequentially scan all pixels in each frame, and frame by frame. This leads to a tremendous computational burden if high resolution camera is used and it becomes a bottleneck for dynamic vision [76, 77]. Various methods have been proposed to cope with this issue. In [78], an optical flow algorithm is used to detect moving objects with a processing cycle time of 66-166 ms. However, some frames may not be processed if a camera with a frame rate of higher than 25 fps is used. This would limit the response of the servo controller that is used to position the vision system. k -means algorithms are popular in clustering-based object tracking schemes [79]. The key weakness is that they cannot deal well with concave data sets on practical scenarios [80]. In [81], the hardware improvement in the sensor layer is attempted by Lichtsteiner and his colleagues to increase the processing speed. They propose a dynamic vision sensor (DVS), which is designed based on addressed

event representation (AER) technique and is different from the conventional frame-based CCD and CMOS sensors. Although its dynamic performance is good, it has weakness in target's detail recognition. In addition, its customized IC design is too specialized for most applications. Its resolution is a bottleneck due to the drastically increasing cost with the increasing compactness. Its latest design is a 128×128 DVS reported in [81], whose resolution is still far lower than the conventional 640×480 video standard.

Fig. 4.12. Block diagram of the object-tracking system.

In this research, we propose a scheme based on the pulsed RBF neural network presented in Chapter 3 is designed for object tracking. Different from the conventional frame-based

methodology, the proposed design utilizes parallel arrays of circuits to extract pixels based on the Otsu's threshold for temporal contrast, which are indicators of moving objects. It can be implemented on a FPGA chip, where the bottleneck of resolution is overcome, and the tracking performance is improved dramatically. Moreover, its integrate-and-fire neural model is implemented in fully parallelism for clustering, where concave data sets can be tackled.

4.4.1. Designs of neural circuits and algorithms

Fig. 4.12 shows the system block diagram under investigation. As shown in the figure, the proposed vision system consists of real-time processing arrays for extracting indicator pixels of moving objects, and parallel neural circuit for real-time clustering, which is a variant of the pulsed RBF neural network presented in Chapter 3. For the indicator extracting module, the current frame of data is stored in buffering memory while the temporal contrast between the current frame and the previously stored frame will be computed on the fly. Thus the significant pixels as indicators of the moving object in the view will be extracted and addressed. The coordinates mapped from the address are then treated as features and transmitted to the dynamic clustering module, which is designed based on a pulsed RBF neural model. Assume frames 1 and 2 are two sequential frames obtained from the frame grabber. A pixel will be extracted as a significant pixel of a moving object if its temporal contrast is greater than an Otsu's threshold θ_g , i.e.

$$|g_n(t_1) - g_n(t_2)| > \theta_g, \quad (4.5)$$

where $g_n(t)$ is the intensity value of the n^{th} pixel at time t . To identify and locate multiple moving objects in the view, a pulse based RBF neural network is used to cluster the extracted significant pixels whose cluster center is dynamic. Let the significant pixels be

$\mathbf{X}_k = (x^k, y^k)^T$ where $k = 1, \dots, M$ is an index number, M is the size of the dataset, x^k and y^k are coordinates of pixel $\mathbf{X}_k$. Moreover, we denote the potential centers of clusters as vectors $\mathbf{c}_j = (c_{xj}, c_{yj})^T$, $j = 1, 2, \dots, N_j$, N_j is the total number of clusters. A pulse based RBF neural network can be constellated to perform the clustering computation as shown in Fig. 4.12. The inputs of the network are pixels' x-y coordinates. The components of cluster center vectors are stored in the pre-synaptic weights between layers I and J. The starting time when an individual postsynaptic potential is excited depends on the difference between the input and the corresponding pre-synaptic weight values. For each output neuron in layer J, its integrated dynamic potential is obtained by integrating all postsynaptic potentials connected to it.

For simplicity of presentation, we denote the coordinate feature of a significant pixel $\mathbf{X}_k = (x^k, y^k)^T$ as $x_1^k = x^k$, $x_2^k = y^k$. Then $\mathbf{X}_k$ is denoted as $\mathbf{X}_k = (x_1^k, x_2^k)^T$. As a variant of IAF model defined in (3.2) and (3.3), the postsynaptic potential (PSP) activated between neurons u_i and v_j is defined as a function of time t as follows,

$$PSP_{ij}(t) = t - (c_{ij} - x_i)^2, \quad i = 1, 2, \quad j = 1, \dots, N_j. \quad (4.6)$$

The integrated potential of neuron v_j is computed as

$$S_j(t) = \sum_{i=1}^2 (t - (c_{ij} - x_i)^2). \quad (4.7)$$

Assume neuron v_j emits an output pulse at time t_j when the integrated potential $S_j(t)$ reaches a predefined threshold θ , then t_j can be derived by substituting θ into (4.7) as

$$t_j = \frac{\theta}{2} + \frac{1}{2} \sum_{i=1}^2 (c_{ij} - x_i)^2. \quad (4.8)$$

In (4.8), t_j reflects a RBF of the Euclidean distance between x_i and c_{ij} . Thus t_j can be utilized for a competitive learning algorithm for clustering. The output neuron v_j is of the integrate-and-fire (IAF) type [20]. When its integrated potential $S_j(t)$ reaches the threshold θ , an output pulse emits. In the competitive algorithm, the neuron v_j with the earliest output pulse will win the competition.

To implement the clustering algorithm in a FPGA, the computation is simplified. Let $f(c_{ij} - x_i) = (c_{ij} - x_i)^2$ and perform the first order Taylor expansion. Neglecting the high order terms and (4.6) can be simplified as

$$PSP_{ij}(t) = t - 2\rho |c_{ij} - x_i|. \quad (4.9)$$

where ρ is a constant. Thus t_j in (4.8) can be written as $t_j = f(\|\mathbf{c}_j - \mathbf{X}_k\|^1)$. That means the firing time t_j reflects the RBF of L_1 -norm distance between center $\mathbf{c}_j$ and pixel $\mathbf{X}_k$ as discussed in (3.8).

The corresponding RBF based clustering algorithm can now be formulated as follows:

Algorithm 4.1

Step 1. Set $k=1$. The initial number of neurons in layer J is set as $N_j = 1$. Seed a new cluster center using the first significant pixel determined by (4.5), and initialize its synaptic delays as

$$\mathbf{c}_1 = \mathbf{X}_k, \text{ i.e. } c_{11} = x_1^k, c_{21} = x_2^k. \quad (4.10)$$

Step 2. Set $k=k+1$. Determine the firing time $t_j^{\min} = \min_{j=1, \dots, N_j} (t_j)$ of the first-fired output neuron excited by integrated PSP in (4.9). If $t_j^{\min} \leq \alpha$, the input data point is within the cluster. Count the data point to the cluster and adjust the center as follows.

$$c_{1j} = c_{1j} + \Delta c_{1j}, c_{2j} = c_{2j} + \Delta c_{2j}, j = 1, \dots, N_j, \quad (4.11)$$

where $\Delta c_{1j} = \gamma(x_1^k - c_{1j})$, $\Delta c_{2j} = \gamma(x_2^k - c_{2j})$, γ is a learning factor. If $t_j^{\min} > \alpha$, the data point is not in any of the existing clusters. Let $N_j = N_j + 1$ and augment the layer J. Initialize the new cluster center as $\mathbf{c}_{N_j} = \mathbf{X}_k$.

Step 3. If the number of points counted to a cluster is greater than a predefined value Γ , it is labeled as a potential target.

Step 4. Let M be the total number of data points. If $k \leq M$, go to step 2. If $k > M$, check the learning termination criteria. If they are satisfied, the learning is ended. Otherwise, set $k=0$ and go to step 2.

This competitive learning algorithm is effective only for convex clusters. To deal with concave clusters, the cluster centers stored in the pre-synaptic weights are considered as local centers instead of final cluster centers. Based on the assumption that components belong to the same cluster have certain density of data points among each other, lateral connections between neurons in layer J are established and strengthened once data points between them appear. For pulse based learning, the timing of first- and second-fired output neuron will determine the establishment of their lateral connection. For instance, if the neurons v_1 and v_2

have established the lateral connection, their mutual lateral strengths L_{12} and L_{21} will be strengthened during learning by adding an increment ΔL . In the retrieval phase, their individual dynamic potentials can be augmented from (4.7) and (4.9) as

$$S_1(t) = \sum_{i=1}^2 (t - 2\rho|c_{i1} - x_i|) + L_{21}u(t - t_{v_2}), \quad (4.12)$$

$$S_2(t) = \sum_{i=1}^2 (t - 2\rho|c_{i2} - x_i|) + L_{12}u(t - t_{v_1}), \quad (4.13)$$

where t_{v_1} and t_{v_2} are firing times of v_1 and v_2 ; $u(t)$ is a unit step function. In this application, we set $\Delta L = \theta/8$, where θ is the threshold for pulse emitting described in (4.8). That means data points exists in between the neurons v_1 and v_2 will strengthen their lateral connection by augmenting the integrated potential for their synchronization. Hence, those RBF neurons whose lateral connections are strong enough, say $L_{12} = L_{21} \geq \theta$, will synchronously be triggered by the augmented potentials when a data point of the corresponding cluster is retrieved. Thus concave-oriented clustering can be implemented by selecting one neuron as a cluster indicator from each group of synchronized RBF neurons.

4.4.2. Experimental results

The proposed parallel neural circuits have been implemented using a Spartan-3A FPGA in MicroBlaze mode at 250MHz. In the experimental FPGA system as shown in Fig. 4.13, the circuits for extracting the temporal feature and neural clustering are design in the block of ‘neural circuits’. External SDRAM is utilized as frame buffer. Control software and system parameters are stored in the on-chip memory. The video frame is 640x480 pixel array with 30 fps frame rate. In the design, 89% of the FPGA resources, i.e. 3.03 millions of logic gates

have been used. The parameters for the design described in subsection 4.4.1 are: $\alpha = 0.2$, $\gamma = 1/64$, $\Gamma = 8$. The multi-layer Otsu's threshold $\theta_g = 0.06$.

Fig. 4.13. Experimental FPGA system. OPB - On-chip peripheral bus; PLB - processor local bus.

Three sets of video clips covering different cases are used in the following discussions. In case 1, video clips of moving table tennis balls are analyzed. These balls can be considered as rigid objects, whose contours are of convex clusters with similar radius. In case 2, flying birds are considered as concave clusters for analysis. In case 3, vehicles with various sizes are tested. The images in the upper row of Fig. 4.14 visualize the extracted pixels as the indicators of moving objects. The corresponding lower pictures show that the moving objects are tracked and highlighted in white frames in real time. The proposed scheme is compared with the conventional optical flow algorithm that runs on a Pentium 4 computer at 3GHz. Moreover, k -means based clustering algorithm [82] is also considered as another

benchmarking counterpart. The results are given in Fig. 4.15 and Table 4.3 where the average lags and errors are summarized from tests of 50 video clips of moving table tennis balls, 80 clips of flying birds in the sky, and 80 clips of moving vehicles on the highways. The length of each clip is about 10 seconds. The results show that the tracking lag increases with larger number of objects. But the proposed and k -means methods are far faster than the optical flow approach. It should be noted that the clock rate of the FPGA is only 1/12 of the Pentium 4. It is also observed that there is little difference between the average lags of k -means and the proposed scheme. However, the k -means has a higher average error rate as shown in Table 4.3 due to the fact that it is not competent for concave clusters [80] that may occur in the video clips. For the optical flow approach, its average error rate is better than the k -means approach and is comparable to the proposed scheme due to its sophisticated algorithm.

Fig. 4.14. Results of identifying and locating different objects using clustering algorithm.

Fig. 4.15. Relationship between number of objects and average lags using different methods.

‘*’ for optical flow; ‘∇’ for *k*-means; ‘o’ for the proposed method.

Table 4.3. Error rates of object tracking

method name	case name	1 object	2 objects	3 and more objects
Optical flow	Case 1	0	0	8.0%
	Case 2	0	4.0%	17.5%
	Case 3	0	2.5%	11.3%
	average	0	2.3%	12.9%
<i>k</i> -means	Case 1	0	0	14.0%
	Case 2	3.8%	11.3%	32.5%
	Case 3	0	8.8%	27.5%
	average	1.4%	7.6%	26.2%
Proposed method	Case 1	0	0	6.0%
	Case 2	0	2.0%	13.8%
	Case 3	0	3.8%	12.5%
	average	0	1.9%	11.4%

4.5. Application to Analysis of Satellite Image by Combining with GA

Multi-spectral remote sensing is a powerful technique for monitoring natural resources and the environment. Based on the analyses of satellite image acquired by multi-spectral remote sensors, various applications can be expected, include geological mapping, pollution and water resource monitoring, vegetation and crop forecasting etc. In this work, we will develop a scheme based on supervised learning to classify the terrain characteristics on a multi-spectral satellite image. In previous sections, the research focuses on the self-organizing based unsupervised clustering algorithm. To extend the parallelism-oriented design to supervised learning method, an enhanced GA (Genetic Algorithm) is investigated to cooperate with the pulse-based neural model. It is well-known that GA is of bio-inspired evolutionary computing. In addition to its merit of global optimization, GA has inherent nature of parallelism. The evolution of chromosomes, including reproduction, crossover and mutation, are suitable for bit-wise digital circuits and applicable to simultaneous operations in large scale platform such as FPGA. There is an increase in research in reconfigurable hardware GA [71-73]. In this research, a parallel GA to seamlessly cooperate with the pulsed neural network for the supervised training scheme is developed. Due to the characteristics of GA and pulsed RBF model, their parallel computational circuit design can be remained at a compact level and realized on a FPGA chip for real time applications. Here, we use GA-PNN as the acronym of this scheme.

4.5.1 Enhanced GA for training of pulsed neural net

To illustrate the scheme, we re-draw the pulsed neural network in Fig. 3.1 as shown in Fig.

4.16. Consider that the input neurons receive an input data set X as

$$\mathbf{X} = \{\mathbf{X}_k : \mathbf{X}_k = [x_1^k, x_2^k, \dots, x_{N_i}^k]^T, \mathbf{X}_k \in [0, 1]^{N_i}, k = 1, 2, \dots, M\}, \quad (4.14)$$

where N_i is the dimension of the data and M is the size of the data set. Moreover, it is assumed that $\mathbf{X}$ is to be classified into N_j classes. The output of the network subjected to the input data $\mathbf{X}_k$ is formulated as

$$\mathbf{Y}_k = [y_1, \dots, y_j, \dots, y_{N_j}]^T. \quad (4.15)$$

Here, y_j is labeled as 1 if v_j is the winner neuron and all the other y_j are labeled as 0. This will be operated by the ‘winner selection’ block in Fig. 4.16. Similarly, the targeting output corresponding to $\mathbf{X}_k$ is defined as

$$\mathbf{Y}_k^r = [y_1^r, \dots, y_j^r, \dots, y_{N_j}^r]^T. \quad (4.16)$$

Fig. 4.16. Model of the pulsed neural network.

Assuming the targeting output of each training date-point is given, a GA algorithm can be applied to search the optimal delay patterns for the proposed pulsed neural network based on supervised training concept.

Fig. 4.17. Architecture of the GA-PNN.

4.5.1.1 Representations

A genome (chromosome) is a set of genes. A gene is a set of possible alleles, and each of which codes for some characteristic. In this study, the chromosome is formulated by

concatenating the integer codes of a set of synaptic delays (see inset A of Fig. 4.17), which are connected to an IAF pulsed neuron. As the training algorithm is to be implemented on FPGA in this study, a pool of chromosomes (population) are constructed and subjected to GA in parallel. Furthermore, the pulsed neuron is constructed corresponding to each of the chromosomes to compute the fitness value. This chromosome population can be expressed as

$$\mathbf{P} = \{P_m : P_m = [c_{1j}, c_{2j}, \dots, c_{N_j}]^T, \quad m = 1, 2, \dots, M_p\}, \quad (4.17)$$

where M_p is the size of the population. Consequently, the number of IAF neurons for training is also M_p .

Fig. 4.17 illustrates the network training. Each training data point is formulated as an input-output pair $(\mathbf{X}_k; \mathbf{Y}_k^r)$, where $\mathbf{X}_k$ is the input vector defined in (4.14), and $\mathbf{Y}_k^r$ is the targeting output vector, and k is the index of the training data.

During the training, a common layer of input neurons labeled as $u_1, \dots, u_i, \dots, u_{N_i}$ receives input data in each cycle. This layer of neurons encodes each component of the input vector into a temporal pulse whose firing time is given in (3.1). Thus the feature of the input vector is reflected in multi-channel temporal pulses. These pulses are then parallelly fed to all pulsed neurons constructed earlier. In Fig. 4.17, the bold lines represent these parallel circuits.

4.5.1.2 Fitness function evaluation

For a training data pair $(\mathbf{X}_k; \mathbf{Y}_k^r)$, there will be a neuron that fires an output pulse earlier than the rest of the $M_p - 1$ neurons. The fitness of the chromosome corresponding to this neuron will be increased while other chromosomes' fitnesses remain unchanged. As shown in the inset B of Fig. 4.17, each chromosome is assigned N_j fitness values $f_m^1, \dots, f_m^j, \dots, f_m^{N_j}$

corresponding to N_j classes, where N_j is the number of pattern classes as described earlier in section 4.5.1. According to the definition of Y_k^r in (4.16), only the j^{th} component is 1 if the training data belongs to class j , while others are 0s. That means only the j^{th} AND gate is enabled while others are disabled. Therefore, if the training data belongs to class j , the fitness of the winning chromosome f_m^j is counted (increased by one) while others remain unchanged. After all the training data have been computed in one generation, the maximum value among $f_m^1, \dots, f_m^j, \dots, f_m^{N_j}$ of the chromosome m is chosen as its resultant fitness value, i.e. $fitness(P_m) = \max_j(f_m^j)$. Selection and mating will be conducted based on these resultant fitnesses after the forward computations for all training data points have been finished in one generation. In this study, the fitness function is formulated as

$$J = \sum_{m=1}^{M_p} fitness(P_m). \quad (4.18)$$

4.5.1.3 Selection and mating

In this phase, the probable candidates are chosen for crossover to form the next generation of offspring. The roulette wheel selection scheme for choosing the parents for mating has been used. The selection probability λ_m for each chromosome is given by

$$\lambda_m = \frac{fitness(P_m)}{\sum_{m=1}^{M_p} fitness(P_m)}. \quad (4.19)$$

where the numerator $fitness(P_m)$ is the fitness value of the chromosome P_m determined in section 4.5.1.2 and the denominator signifies the total fitness of the population. The number of copies that a particular chromosome is used in the selection pool is determined as

$$\gamma_m = \frac{M_p \cdot \text{fitness}(P_m)}{\sum_{m=1}^{M_p} \text{fitness}(P_m)}. \quad (4.20)$$

The resultant $\sum \gamma_m$ could be unequal to the population size M_p . In this case, copies of chromosomes will be increased or deleted with uniform probability so that $\sum \gamma_m$ fits to the population size M_p . Thus, a new population of M_p chromosomes is derived and the system proceeds to the crossover stage.

As all the parents are available after the selection phase, crossover operation can be performed in parallel on all the parents. This will result in all M_p offspring being produced instead of 2 offspring at a time. The crossover probability is taken as 40%. After crossover, some of the chromosomes are randomly mutated by inverting some of their bit values. The mutation probability is kept very low at 4%. Chromosomes involved in crossover and mutation will be selected randomly based on the random indexes generated by random number generators.

4.5.1.4 Finalization of GA

After the above operations of one GA generation, the new population is fed as new concatenated delays for the input neurons, and the GA is repeated. To finalize the GA and conclude the training procedure, two conditions must be considered. The first is the number of evolutionary generations, which is usually a heuristic parameter in GA applications. The second is classifying capability of the obtained class centers, which can be related to the difference between $\sum_{m=1}^{M_p} \text{fitness}(P_m)$ for one GA generation and the size of training dataset M .

To analyze the second condition, we give the following remark.

Remark 4.1: In one generation (epoch) of the proposed GA-PNN algorithm,

$$(\sum_{j=1}^{N_j} f_m^j - \max_j(f_m^j)) \rightarrow 0 \text{ for all } m, \text{ if and only if } (M - \sum_{m=1}^{M_p} \text{fitness}(P_m)) \rightarrow 0.$$

Remark 4.1 describe the fact that there should be as little as possible overlap among the

coverage of different class centers, in the case of $0 \leq (M - \sum_{m=1}^{M_p} \text{fitness}(P_m)) < \delta$ (i.e.

$(\sum_{j=1}^{N_j} f_m^j - \max_j(f_m^j)) \rightarrow 0$) where the positive value δ is as small as possible. Then the

classifying capability of the class centers would be considered acceptable.

To define the condition of the finalization of the GA, the number of appearance of

$0 \leq (M - \sum_{m=1}^{M_p} \text{fitness}(P_m)) < \delta$ are recorded during the training procedure. Based on the above

discussions, the conditions under which the GA can be finalized are listed as below:

a. The number of GA generations is not less than the predefined (heuristic) value N_{GA} ;

b. The times the appearance of $0 \leq (M - \sum_{m=1}^{M_p} \text{fitness}(P_m)) < \delta$ is not less than a positive value

N_δ , where δ and N_δ are predefined heuristic values.

If all the three conditions are fulfilled, the proposed GA-PNN training is finalized. The final

selection of the chromosomes is output as the resultant class centers can be conducted as

follows:

$$m_j = \arg(\max_m(f_m^j)), \quad j = 1, 2, \dots, N_j, \quad (4.21)$$

where $m_1, m_2, \dots, m_{N_j}$ are the indexes of those selected chromosomes representing different class centers.

4.5.2 Hardware platform

Both the GA and the pulse neural network have been implemented using the Virtex II XC2V1000fg256 FPGA housed in a RC200 developer board. The FPGA has a maximum capacity of 1,000,000 logic gates including 40x32 configurable logic blocks (CLBs, each made of 4 slices) and 40 select-RAM blocks. The RC200 board is also fitted with two 2-MB ZBT RAM banks.

In the implementation, we have used Handel C [83], which is based on ANSI-standard C, as coding language. It is a programming language designed for compiling programs into hardware such as FPGAs or ASICs. Handel-C targets hardware directly, and provides several hardware optimizing features. At the basic level, the coding language's constructs can be exploited by parallel execution of all simple statements. Another merit is the simultaneous calls to functions. Using arrays of functions, large blocks of functionality can be generated that execute in parallel. As illustrated in Fig. 4.17, large number of parallel components and blocks, such as N_i input neurons, $N_i \times M_p$ synapses, M_p IAF neurons, M_p chromosomes, $N_j \times M_p$ fitness blocks, and parallel GA operations (including reproduction, crossover and mutation etc), are designed in the system owing to the inherent nature of parallelism of GA and pulsed neural network.

A random number generator (RNG) based on the multiple bit lagged Fibonacci generator [84]

is utilized to generate random numbers. The RNG is running in the background of the forward neural computations so that the necessary random numbers are ready for the selection, crossover and mutation operations.

4.5.3 Experimental results for satellite image processing

4.5.3.1 Data and parameters

In this section, a Landsat scene of Chernobyl area [85] is analyzed. Fig. 4.18(a) shows the Chernobyl scene. This scene has a size of 512 pixels by 512 scans. Each pixel is represented by a feature vector of five values, which are the 8-bit quantified values of selected spectral bands listed in Table 4.4. According to the format of the multi-spectral data, the structure of the GA-PNN can be constructed. The number of input neurons is selected as $N_i = 5$ catering to the dimension of the feature vector. Each synaptic delay is quantified as an 8-bit integer. They are stored in the off-chip memory of the RC200 system. Genetic chromosomes are formulated by concatenating the 8-bit codes of a set of synaptic delays according to (4.17). The GA population size $M_p = 2^7 = 128$. This selection will be explained in the ensuing subsection. The crossover rate is 50/128; mutation rate is 5/128. The parameters for GA finalization discussed in section 4.5.1.4 are: $N_{GA} = 50$; $N_\delta = 20$; $\delta = 1$. For the training, 256 pixels are manually selected and labeled for each training dataset, i.e. $M = 2^8$. Based on the ground truth, these labeled training data represent three different terrain characteristics – water, building and vegetation ($N_j = 3$). According to (4.14), each input training data point is represented as a feature vector $\mathbf{X}_k = [\text{band1}, \text{band2}, \text{band3}, \text{band4}, \text{band5}]^T$. The corresponding targeting output formatted in (4.16) is labeled as $\mathbf{Y}_k^r = [1, 0, 0]^T$ for water terrain, $\mathbf{Y}_k^r = [0, 1, 0]^T$ for building terrain and $\mathbf{Y}_k^r = [0, 0, 1]^T$ for vegetation terrain. Fig.

4.18(b) shows the areas for labeling training data. In this study, 50 datasets are selected, where the size of each set is 256. When the trained pulsed neural network is used for terrain classification of the whole scene, a measure of class radius is set. If the firing time of the winning neuron excited by a pixel is within the radius limit, say α , this pixel is classified to the class represented by the winning neuron. Otherwise, the pixel is labeled as an uncertain pixel. The effect of α to the results of terrain classification will be discussed in the following subsection.

Fig. 4.18. Learning-based classification. (a) Chernobyl scene. (b) ground truth for labeled area. (c) classified output for water. (d) classified output for man-made material. (e) classified output for vegetation. ■ – water; ■ – vegetation; ■ – buildings.

Table 4.4. Spectral bands

Band	Spectral Range in μm	Color
1	0.52 — 0.60	Green
2	0.63 — 0.656	Red
3	0.76 — 0.90	Near Infrared
4	1.55 — 1.75	SWIR
5	2.08 — 2.23	SWIR

4.5.3.2 Performance Evaluation of the GA-PNN

As the ground truth of the Chernobyl area is known, the accuracy of the GA-PNN scheme can be evaluated. Fig. 4.19 illustrates the effect of the class radius α on the classification accuracy with respect to the ground truth. In the experiment, 25 pairs of training/test datasets are used. We can observe that the average accuracy falls from the optimal range when α is selected beyond 0.09 to 0.13. In the following experiments, α is set as 0.098.

Experiments have also been to evaluate the effect of the size of chromosome population, M_p .

In Fig. 4.20, we can observe that the required logic gates for implementation are increased drastically when M_p exceeds 128, while the accuracy is saturated around about 94.5%. Thus we set $M_p = 128$ for the experiments.

Based on the selected parameters for the GA-PNN, outputs of the classified terrain areas are visualized in Fig. 4.18 (c), (d) and (e). To further evaluate the classifying performance of the

proposed GA-PNN, a BP (i.e. back propagation) neural network is used as a counterpart for comparison, which is constituted with 5 input neurons, 13 hidden neurons and 3 output neurons as described in [85]. Fig. 4.21 compares the accuracy trends with various epoch numbers of both methods. For both methods, the accuracy increases with the increasing epoch number. The value of fitness function reaches 255 when the epoch number is 46. The accuracy of GA-PNN is saturated at 94.5% with the epoch number greater than 50. In contrast, BP net needs more epoch number to reach its optimal accuracy. Fig. 4.22 visualizes the classified output based on BP net and GA-PNN for this multi-spectral image. The results show that the performance of the proposed method is comparable to the BP neural network.

Fig. 4.19. Effect of α to the classification accuracy.

Fig. 4.20. Effect of M_p to the classification accuracy.

Fig. 4.21. Accuracy of GA-PNN and BP net with various epoch numbers.

Fig. 4.22. Comparison of classification results. (a) Classified output with a BP neural network. (b) Classified output with the proposed neural network. ■ – water; ■ – vegetation; ■ – buildings; White – uncertain.

For the experimental hardware platform, the clock speed of the FPGA board is set at 50 MHz. At this clock speed, the system takes approximately 4 msec to complete one GA generation with 256 training data-points. Overall, the system takes about 200 msec to complete the training of 50 generations. In contrast, the BP neural network is implemented by software simulator because of its complex computations on multiplication and nonlinear activation functions. The BP procedure takes about 43 seconds for 100-epoch training on a PC with Pentium 4 whose frequency is 3GHz.

To investigate the compactness of the computational circuits, the FPGA resource utilization and distribution are summarized in Table 4.5. From the table, it is observed that the one million-gate FPGA can be used to implement the GA-PNN for the terrain classification comfortably.

Table 4.5. Hardware resource utilization

resource name	resource usage	percentage of resource usage
Slices	2401	47%
LUTs	2843	27%
Flip-flops	1376	10%
Block memory	18	45%
Total equivalent gate count	374,557	37%

4.6. Conclusions

The proposed pulsed RBF neural network has been implemented for both the forward computation and learning algorithm with a FPGA board. In contrast to other conventional and pulse-based neural networks, the proposed pulse neural network works as a RBF network by utilizing the synchronism of input pulses. The proposed system requires no multiplication in both its forward computation and learning algorithm. These characteristics improve the compactness of the hardware circuits. As an application demonstration, it has been extended to a nonlinear look-up table and applied to estimate the friction occurs in a precision linear stage. The processing speed is competent for real-time adaption to the time-varying processing. Another application of the proposed neural circuits is investigated in the real-time moving object tracking system. In contrast with conventional frame-based methodology, the proposed design extracts pixels with significant temporal contrast, which are indicators of moving objects. The capability of the proposed model in dealing with concave data set has

also been validated. The experiments show that the proposed scheme outperforms the k -means and optical flow methods in both single and multiple objects cases.

The proposed approach has also been extended to GA based supervised learning algorithm. Its computational circuits have been realized enough for real-time terrain classification of a multi-spectral satellite image. Experimental results show that its classification performance is comparable to a BP neural network while its learning speed is over 200 times faster.

Chapter 5

Multi-Channel Pulse Coupled Neural Networks

5.1 Introduction

In previous chapters, the pulsed neural networks that have the temporal pulse characteristics have been presented. In this chapter, we will investigate a variant version, the pulse coupled neural network (PCNN), where the network topology is changed from fully connected multilayer net to a 2-D inter-connected single-layer net. The basis for PCNN is the linking field model of Eckhorn and his colleagues [16]. It is inspired and proposed based on experimentally observed feature-dependent activity of cortical neurons in the visual cortices of mammals such as cats and monkeys. In a typical PCNN, neurons are constellated as a 2-D array in a single layer so that pixels of an image can be correspondent to the neurons by one-to-one mapping for neural processing. A pulse coupled neuron in Eckhorn's network, as shown in Fig. 5.1, consists of three functional units: feeding receptive field, linking receptive field and a spike generator. Its key concept was the introduction of a secondary receptive field, the linking field, whose integrated input from intra-layer neighboring neurons modulates the primary feeding receptive field input. This coupling action provides an effective way for studying synchronous pulse dynamics in networks in the form of a 2-D

array, which is essential for image processing [43, 46-48]. Its application includes scene segmentation [31, 44], image smoothing [53], image fusion and decomposition [54, 55], path optimization [56] and image feature extraction etc [57- 59]. The typical PCNN model is confined to the processing of grayscale images. One of the reasons is the essence of its inter-neuron coupling is defined in terms of products of scalars. For pixels featured with color vectors, the relationship between the timing of a pulse and a pixel's feature will no longer be scalar-to-scalar. Another reason is the computational burden of the algorithm. A typical PCNN is a 2-D array of neurons with one-to-one pixel to neuron correspondence. Including its complex dynamics and multiple inter-neuron coupling, the computation will be undoubtedly time-consuming for color images if executed in sequential codes.

Figure 5.1. Model neuron of Eckhorn's PCNN [44].

Recently, some modified PCNN have been reported to be applied to multi-channel image fusion and real-time path planning [47, 54-56]. Nonetheless, the solutions to the aforementioned two issues of the conventional PCNN are still limited. In [54], for instance, a

model named m-PCNN is reported to perform medical image fusion. It augments the single-channel PCNN by extending the number of input channels. This direct extension does not overcome the aforementioned two issues of the conventional PCNN, and its application is limited within image fusion technique, which is radically different from image segmentation technique that is the focus of this study. Recent related literatures also include a self-organizing pulsed network, namely pulsed SOM, reported in [38]. The SOM is an iterative organizing algorithm. In contrast, the self-organizing phase of PCNN can be performed without iterative learning. Moreover, the lateral linking of PCNN is designed for linking of adjacent neurons in 2D space but not for inhibiting actions in the pulsed SOM algorithm.

In this research, we propose an improved PCNN model to perform color image segmentation and circumvent the abovementioned problems of conventional PCNN. Firstly, it is designed as a vector-oriented model to deal with color images. This is achieved by introducing a pulse based radial basis function (RBF) [15, 20] into the model neurons of PCNN to work as a bio-inspired computation unit. It uses the timing of individual pulse produced by the neurons to determine distances between pixels' feature vectors and their rank order respectively, so that the fast links can be established among neurons with respect to the spectral feature vectors and spatial proximity of mapped pixels. The model's pulsed behavior is designed to be compatible with PCNN's characteristics so that the algorithm can efficiently deal with color images whose pixels are presented in vectors instead of scalars. Here the dimension of a pixel's spectral feature vector is not limited at three. Thus processing of color images can be considered as a special case of this multi-channel image processor. In general, the proposed approach can be used to deal with multi-spectral or hyper-spectral images. For brevity, we use the acronym, MPCNN, for the proposed multi-channel pulse coupled neural network. Secondly, the segmentation algorithm is designed in full parallelism by utilizing the 2-D

structure of PCNN. In this system, the neurons' connectivity may transfer in parallel in the 2-D space to establish intra/inter-segment linking. To improve the computational speed, some dynamics of pulsed neurons are simplified to make the neural circuits more compact for implementation on large scale programmable digital integrated circuit such as FPGA.

Fig. 5.2. Multi-channel PCNN with 2D array of neurons.

5.2 Multi-Channel PCNN for Color Image Segmentation

5.2.1 Multi-channel PCNN

Similar to conventional PCNN, the proposed network is a 2D array of neurons as shown in Fig. 5.2. In this system, each neuron maps one pixel of the corresponding image. Segmentation of pixels will be implemented by performing couplings among the corresponding neurons. The bottom box in Fig. 5.2 illustrates the model of a single neuron. To be consistent with the conventional PCNN, the input fields are classified as linking and feeding fields. Here the feeding is defined as the color feature vector of the current pixel. The linking field of each neuron is further divided into two subfields for receiving color features and linking pulses from its 8 neighboring neurons respectively to establish interconnections. One of the key differences between the proposed and conventional PCNN is that the latter uses the pixel intensity to implement coupling of pulses while the proposed PCNN utilizes the feature vectors.

To establish vector-oriented coupling, a pulse-based RBF model is employed as a computing unit. As shown in Fig. 5.3, the pulse-based RBF model will compute and rank the distances between the current pixel's feature vector, $\mathbf{x} = [x_1, \dots, x_i, \dots, x_{N_i}]^T$, and the linking spectral vectors, $\mathbf{c}_j = [c_{1j}, \dots, c_{ij}, \dots, c_{N_i j}]^T$, $j = 1, \dots, N_j$, which come from the neighboring pixels and are stored in the synapses of the RBF model. Here j is equivalent to the index of neighboring pixels. In this research, $N_j = 8$ as we only consider the coupling among the current pixel and its 8 neighborhoods. N_i denotes the dimension of the feature vector. For RGB color images used in this study, we can specify $\mathbf{x} = [R, G, B]^T$. The RBF neuron v_j is of the integrate-and-fire (IAF) type [14, 20]. Its computing is described as follows. The distances between $\mathbf{x}$ and

c_j determines the starting time that an individual postsynaptic potential (PSP) will be excited.

The PSP between neurons u_i and v_j is defined as a function of time t as follows,

$$PSP_{ij}(t) = \begin{cases} t - |c_{ij} - x_i| & \text{if } t \geq |c_{ij} - x_i| \\ 0 & \text{if } t < |c_{ij} - x_i| \end{cases} \quad (5.1)$$

In the time period that $t \geq |c_{ij} - x_i|$ for all i , the integrated potential of neuron v_j is computed as

$$S_j(t) = \sum_{i=1}^{N_i} PSP_{ij}(t) = \sum_{i=1}^{N_i} (t - |c_{ij} - x_i|). \quad (5.2)$$

After all the PSPs have been excited, the firing of output pulse can be expected. Assume the neuron v_j emits an output pulse at time t_j when the integrated PSP $S_j(t)$ reaches a threshold θ as shown in the inset in Fig. 5.3, then t_j can be derived by substituting θ into (5.2) as

$$t_j = \frac{\theta}{N_i} + \frac{1}{N_i} \sum_{i=1}^{N_i} |c_{ij} - x_i|. \quad (5.3)$$

In (5.3), t_j reflects a RBF of the Manhattan distance between $\mathbf{x}$ and $\mathbf{c}_j$, which is defined as follows:

$$\|\mathbf{c}_j - \mathbf{x}\|^1 = \sum_{i=1}^n |c_{ij} - x_i|. \quad (5.4)$$

For the 8 neurons v_j , their temporal output pulses also indicate the rank order of distances between the current pixel and its 8 neighborhoods. This rank order will be utilized in the segmentation algorithm presented later.

Fig. 5.3. Pulse based RBF model.

To perform image segmentation, PCNN neurons are supposed to link to each other according to the similarity of the corresponding mapped pixels. As shown in Fig. 5.2, we simplify the conventional inter-neuron *fast links* [17] as hard-wired connections of pulse linking fields. These *fast links* can be established among neurons according to a criterion related to inter-pixel distance computed in (5.3), which reflects the similarity of their spectral feature. The criterion will be described in section 5.2.2. After all the links are established, the representative neuron of each linked region will be excited sequentially. This will cause all the neurons in the linked region to emit output pulses synchronously. A neuron will be forced into refractory state right after it emits an output pulse. It will be recovered to the normal state after all the neurons of the network get into the refractory state. This mechanism provides the basis for image segmentation by linking homogenous regions. An algorithm will be proposed for this PCNN to perform image segmentation in the ensuing subsection.

5.2.2 Algorithm for color image segmentation

In this study, we consider a pixel interacting with its 8 neighbor pixels. Some researchers [50] have showed that PCNN can be designed as an alternative to seeded region growing (SRG) [86-88] algorithm. In the color space, if the maximum distance to its 8 neighbors is less than a predefined value, the pixel is defined here as a *seeded pixel*. The corresponding neuron is named as *seeded neuron*. Consider the Manhattan distance of feature vector as the distance metric, then pixel s , with $\mathbf{x}^s = [R^s, G^s, B^s]^T$, is a *seeded pixel* if and only if

$$\mu_s \leq \theta_\mu, \quad (5.5)$$

where $\mu_s = \max_{j=1,\dots,8} \{\|\mathbf{c}_j - \mathbf{x}^s\|\}$, and $\|\cdot\|$ is defined in (5.4), and θ_μ is a predefined threshold value.

This definition means a *seeded pixel* has a similar feature to its neighborhoods in some predefined degree. In the following algorithm, *seeded pixels* will link to their neighbor pixels under predefined conditions so that seeded regions can be expanded by capturing pixels with similar features. There are two points to be explained. Firstly, the seed selection and growing are operated in parallel, and the connectivity among neurons with proximity may transfer spatially in the 2-D network. Secondly, this linking phase is a preliminary stage of image segmentation before the region-merging phase. Over segmentation is desirable to avoid missing the important details. Thus a small threshold value θ_μ in (5.5) is desired. According to color quantization theory [89], we require approximately 26 quantization levels for each of R, G, and B domain to distinguish 17000 colors, which is the usual capability of human visual perception. As θ_μ is equivalent to the radius of a quantization interval in RGB space, its normalized value can be set as 1/52.

After the seed selection and growing, region-merging operation will be performed. Different from conventional sequential algorithms [87, 90], this region-merging scheme is designed in parallelism to apt to parallel implementation platform. Assume data of pixel array has been stored into the PCNN neuron array by one-to-one mapping. The proposed segmentation algorithm can be performed as follows.

Firstly, the *seeded neurons* are determined using (5.5) for all PCNN neurons. The maximum value in (5.5) can be quantified by the timing of the latest output pulse of the current RBF as shown in Fig. 5.3. The *fast links* will be established between the *seeded neuron* and its 8 neighborhoods as illustrated in Fig. 5.2. Neurons link with a seeded neuron are labeled as *linked neurons*. These operations are parallel and simultaneous for all PCNN neurons and their inter-links. In this step, groups of neurons mapped from homogenous pixels are linked together.

Secondly, a mean feature vector $\sigma^g = [\sigma_1^g, \dots, \sigma_i^g, \dots, \sigma_n^g]^T$ for each group of linked neurons is computed as follows.

$$\sigma_i^g = \frac{1}{M_g} \sum_{j_g=1}^{M_g} x_{ij_g}, \quad (5.6)$$

where g is the group index, j is the pixel index, and M_g is the total number of pixels in the group. As described in section 5.2.1, x with subscript i denotes the color component of a pixel, i.e. R, G or B. The subscript j_g is the pixel index in group g . For RGB image, we can

specify the mean color as $\sigma^g = [\sigma_R^g, \sigma_G^g, \sigma_B^g]^T$. Replace the RGB vectors of the linked neurons using the mean RGB vector of its group.

Thirdly, the unlinked neurons are considered. For any unlinked neuron u that has *linked neighbor neurons*, a fast link will be established with the linked neighborhood, neuron j_u , whose group's mean feature vector is the nearest to this unlinked neuron, i.e.

$$j_u = \arg(\min_j \|\mathbf{x}^u - \sigma^{g_j}\|), \quad (5.7)$$

where g_j is the index of group which neuron j belong to. $\mathbf{x}^u$ is the color vector of the pixel mapped to neuron u . The resulting neuron in (5.7) can be determined by using the rank order of distances indicated by (5.3). This newly linked neuron is termed as a *linked neuron*. There are several points to be noted. First, only *linked neighbor neurons* are qualified for the selection in (5.7). Second, these operations are parallel and simultaneous for all unlinked neurons. Third, to finalize the linking of an unlinked neuron, its distance to the nearest linked neuron must be less than a threshold, say θ_l . This threshold will be gradually loosen until all isolated neurons are linked.

As described previously, threshold value θ_μ in (5.5) is selected based finest perception of human being. The result at this stage has high degree of over-segmentation, which is usually cannot be qualified as final result. To finalize the segmentation algorithm based on the regions obtained by above linking phase, the adjacent regions need be merged if they fulfill the size and color distance conditions. Here three pairs of thresholds for segment size and color distance are used to perform a three-level merging. The details of the first level are described as follows.

If the size of a region is less than θ_a^z , the region will be merged with one of the adjacent regions whose color distance to the current region is less than θ_a^d . Here the superscript z and d denote the region size and color difference (distance in color space) respectively. The subscript a is a step index. Compute and store the mean color of the new region similar to (5.6). Note that if there are multiple adjacent regions satisfy the distance threshold, only one (randomly) will be merged in this epoch. The merging can be implemented by linking neurons on their adjacent edges. The operations are parallel and simultaneous for all regions. Then repeat the operations until no region satisfies the thresholds. Similar operations will be conducted for the second and third level of region merging. Thresholds θ_b^z and θ_b^d are used for the second level, where $\theta_b^z < \theta_a^z$, and $\theta_b^d > \theta_a^d$. Similarly, thresholds θ_c^z and θ_c^d are used for the third level, where $\theta_c^z < \theta_b^z$, and $\theta_c^d > \theta_b^d$. The summary of the complete algorithm is listed as follows.

Algorithm 5.1

- Step 1.** Initializations. In the 2D space, perform simultaneous seeding, linking and grouping based on (5.5).
- Step 2.** Refresh color $\mathbf{x}$ of neurons of each group using group mean σ^g in (5.6).
- Step 3.** If all neurons are linked, go to Step 4. Otherwise, simultaneously link all the qualified unlinked neurons to the nearest adjacent group. This can be done by linking each qualified unlinked neuron to a grouped neuron selected by (5.7). In addition, distance from the unlinked neuron to the grouped neuron must be less than the threshold θ_l . Loosen the threshold θ_l by setting $\theta_l = \theta_l + \Delta\theta_l$. Go to Step 2.

Step 4. Perform the three-level region merging using three pairs of thresholds for segment sizes and color distances respectively.

Fig. 5.4. Observation before and after region-merging operation. Row 1: original images; row 2: pseudo-color results before region-merging; row 3: pseudo-color results after region-merging.

Fig. 5.5. Impact of parameters for region merging. The results are represented in pseudo-color. Column 1: original image; column 2: results by normal parameters; column 3: results after the value of parameters increased by 50%; column 4: results after the value of parameters increased by 100%.

Fig. 5.4 shows the effect of segmentation before and after region-merging operation using some example images. In these examples, the images ‘Mandrill’ and ‘Polar bears’ are highly textured. From the pseudo color results in row 2, we can observe that their degrees of over-segmentation are higher than the other two examples. The image ‘Peppers’ results in the least over-segmentation due to its content of large and uniform color regions. The image ‘Lena’ has a median result. For the results after region merging in row 3, it is observed that all of the results are improved significantly from the over-segmentation status. It is noted that the results of this segmentation algorithm have dependency on the values of the thresholds for region merging in step 4. Some researchers [87, 90] have discussed this problem, and suggested some empirical guidelines for the selection of these thresholds. Based on the guidelines in [87], we refine the value of these thresholds through heuristic tuning using images of Berkeley Dataset [91, 92], where 200 images are used for training, and 100 images

for retrieval test. These values are listed as follows: $\theta_a^z = 1/20$; $\theta_a^d = 0.04$; $\theta_b^z = 1/50$; $\theta_b^d = 0.07$; $\theta_c^z = 1/150$; $\theta_c^d = 0.13$, where the values of size thresholds are the fraction of total pixel number.

The thresholds of step 4 are consistent with the fact that human perception is more sensitive to the changes in smooth areas, which are usual relatively large. We select a small threshold of color difference for merging large regions. The thresholds (θ_b^z and θ_c^z) of further merging steps depend on the sizes of concerned objects and their parts. To avoid missing important details, we intentionally produce slight over-segmentation by using conservative thresholds.

The thresholds of region size are empirical values suggested in [87, 90]. To observe the impact of color-difference thresholds, some random examples are investigated as shown in Fig. 5.5. We can observe that the impact on the results is not significant after the value of parameters are increased by 50%. When the values of parameters are increased by 100%, obvious impact is observed. In the upper row, the left wing of the plane is merged into the shadow. For the image Lena, some detailed face features are also merged into large regions.

5.3 Evaluation on Segmentation Performance

To compare the segmentation performance of MPCNN with classical and state-of-the-art counterparts, we adopt the quantitative measures defined in [91], i.e. *Precision* (P), *Recall* (R) and *F-measure*. According to [91], *Precision* is the fraction of detections that are true positives rather than false positives, while *Recall* is the fraction of true positives that are detected rather than missed. *F-measure* is a weighted harmonic mean of P and R , which is defined as,

$$F = \frac{PR}{(\alpha R + (1 - \alpha)P)}, \quad (5.8)$$

where α is a weighting factor and is set to 0.5 similar to the value used in [91].

Fig. 5.6. Segmentation results of examples. The results are represented in pseudo-color. Column 1: original image; column 2: JSEG; column 3: method in [90]; column 4: SRG; column 5: the proposed approach.

Table 5.1. Comparison of segmentation results among various methods.

row number	JSEG				method in [90]				SRG				proposed			
	N_g	P	R	F	N_g	P	R	F	N_g	P	R	F	N_g	P	R	F
1	30	0.56	0.83	0.67	30	0.72	0.90	0.80	40	0.51	0.85	0.64	49	0.49	0.88	0.63
2	23	0.49	0.75	0.59	57	0.54	0.91	0.68	27	0.53	0.81	0.64	41	0.53	0.91	0.67
3	4	0.79	0.84	0.81	8	0.78	0.93	0.85	5	0.83	0.95	0.87	5	0.81	0.92	0.86
4	35	0.58	0.75	0.65	70	0.56	0.84	0.67	28	0.56	0.72	0.63	39	0.54	0.88	0.66
5	26	0.45	0.82	0.58	47	0.42	0.92	0.58	25	0.41	0.81	0.54	51	0.40	0.95	0.56
6	17	0.74	0.79	0.76	32	0.69	0.92	0.79	36	0.62	0.83	0.71	42	0.61	0.90	0.72

Fig. 5.6 shows the segmentation results of some examples from Berkeley dataset using JSEG [93], method in [90], conventional SRG and MPCNN respectively. Their corresponding P , R , F -measure and the resultant number of regions N_g are summarized in Table 5.1. We can observe that the performance of method in [90] is the best because of its sophisticated synthetic algorithm. MPCNN is comparable with the well-known JSEG and conventional SRG, which are hybrid of boundary-based and region-based methods. From Table 5.1, MPCNN has higher *Recall* values, which add a positive factor to the corresponding values of F -measure. This is because the MPCNN is designed to produce a slight over-segmentation. This is also directly reflected on the number of regions N_g . It is observed that N_g values of the proposed method are either the highest or second highest among all the methods. On the other hand, over-segmentation results in lower *Precision* values as compared with other counterparts. Another phenomenon is that MPCNN and SRG give lower values of P and F -measure for images in rows 1 and 6 of Fig. 5.6. This is because these two methods are relatively less competitive for highly texture images. This weakness has also been reported in [87]. We can observe that MPCNN produces slight over-segmentation in the results of rows 1 and 6 due to the textured patterns of grassland and rough surface of rocks. The resultant N_g

values also verify this observation. In both high-texture cases (row 1 & 6), MPCNN and SRG result in the highest and second highest N_g respectively.

To investigate the noise-resistance of MPCNN, an example image in Fig. 5.7 is analyzed by adding the salt-and-pepper noise with various noise densities. Fig. 5.8 summarizes the F -measure values of the proposed approach as compared with JSEG and SRG. The results show that the MPCNN's resistibility to salt-and-pepper noise is comparable to SRG and JSEG. The visualized results in Fig. 5.7 illustrate that the boundaries of road are less disturbed to the results of MPCNN in contrast to the other counterparts, while the segmentation of the airplane is almost unchanged for all cases.

These evaluations indicate that the performance of the proposed MPCNN is comparable to popular segmentation algorithms such as JSEG and SRG. Although it is not as competitive as some sophisticated synthetic algorithm, its parallelism based design can enable it to have a higher processing speed. This point will be validated in the following investigations.

Fig. 5.7. Segmentation results in pseudo-color for images polluted by salt-and-pepper noise. Column 1: noisy images; column 2: results of JSEG; column 3: results of SRG; column 4: results of MPCNN. Noise density of row 1 is 0.02; noise density of row 2 is 0.05.

Fig. 5.8. Comparison of F -measure versus different noise density.

5.4. Application to Object Detection

The result of segmentation based on MPCNN can be used for object detection. As a post-processing stage of MPCNN, this scheme for object detection consists of three phases. In phase 1, the resultant segments will be grouped under geometric constraints such as area constraints, elongation constraints and separation constraints, which can be obtained from *a priori* knowledge of the targeting objects and imaging systems. Base on the assumption that the object of interest always has one dominant segment whose area is the largest among all segments belong to this object, an algorithm will be designed to search the object from the groups of segments. In phase 2, a shadow analysis is used to reduce the searching range. In phase 3, principle component analysis (PCA) is employed to finalize the target recognition.

5.4.1. Grouping segments

In this research, the investigation is limited within the scope of color aerial photos. The object of interest, say a military object, can be painted in any colors and patterns. Therefore, there is no spectral feature of the object can be considered, i.e. only geometric features (area,

elongation and separation etc) are available for object detection. These geometric features can be obtained from *a priori* knowledge of the related targets and imaging systems. An assumption of this research is that the object of interest has one dominant segment whose area is the largest among all segments belong to this object in the image. That means if this dominant segment is selected as the root of a group to enroll other nearby smaller segments under predefined constraints of geometric dimensions, the object could be certain combination of the segments, including the root, in the resultant group. Assume the segmentation algorithm partitions the image H into a set of subset (regions) $\Phi = \{H_1, \dots, H_m, \dots, H_M\}$, i.e.

$$\bigcup_{m=1}^M H_m = H \quad \text{with } H_m \cap H_n = \text{null}, \quad m \neq n, \quad (5.9)$$

where $\cup$ denotes union operation and $\cap$ denotes intersection operation. The grouping scheme is designed as follows:

Algorithm 5.2

Step 1. Disable the pixels of overlarge and overlong segments (as compared with predefined values of the object). Set $k=1$.

Step 2. Find the largest-area segment H_{R_k} among $\Phi = \{H_1, \dots, H_m, \dots, H_M\}$, i.e.

$$R_k = \arg \max_m (\psi(H_m)), \quad \text{where } \psi(H_m) \text{ denotes the number of pixels (i.e. segment}$$

size) of H_m . Create a group G_k using H_{R_k} as its root subset, i.e. $G_k = \{H_{R_k}\}$. Set

$$\Phi = \Phi - H_{R_k}.$$

Step 3. In Φ , each segment whose centroid is in the predefined range centered by centroid of the root is enrolled in the group if the union of the segment and the root satisfies all constraints of geometric dimensions.

Step 4. If the sum of the areas of all the segments in Φ is less than the predefined lower limit of the object's area, go to Step 5. Otherwise, set $k=k+1$, and go to Step 2.

Step 5. Eliminate the groups whose sum of the areas of all its segments is less than the predefined lower limit of the object's area. Terminate the algorithm.

The algorithm implies that the segments that constitute the object must have been included in one of the obtained groups, say G_k , which is dominated by its root segment H_{R_k} . In each group, any combination of the segments rooted by the root segment could be a potential object if it satisfies area and elongation constraints. Here we use a case study based on the example image in Fig. 5.9 (a) to illustrate the grouping algorithm. The target template of this case is the fighter, whose geometric features have been given by *a priori* knowledge. As the parameters of the aerial imaging system, including image size, field of view and airborne height, are also available, the geometric features of the template can be mapped to the captured image by image scaling. For this case, the upper limits of the segment size and elongation are selected as 5600 pixels and 130 pixels based on the *a priori* knowledge. Fig. 5.9 (c) – (f) show some combinations of potential objects derived from the segmented regions. We can observe that all these potential objects are combinations of segments smaller than the predefined upper limits of the geometric features. The overlarge segments, for example the building roofs, are excluded by the algorithm. In addition, there is a root segment with dominant size in each combination. All potential groups of segments will be

analyzed by further algorithm for target recognition. It is observed that the targeting fighter (Fig. 5.9(f)) is included in the potential results.

5.4.2. Shadow-based object detection

Even though the resultant segments can be grouped using the bottom-up scheme [94] given in section 5.4.1, the number of potential combinations could be too large for practical applications. To tighten the constraint for potential combinations, main shadow of the object will be considered in light of the fact that many objects in aerial photos appear with shadows.

Assuming the direction of sunlight in the aerial photo is known. For each potential object derived in 5.4.1, its adjacent external segments on the supposed-shadow side are investigated based on the method in [55]. If any significant shadow is found, this potential object will be highlighted. For example, suspected shadows are discovered in Fig. 5.9(c) and (f) and highlighted in white color. That means the combinations in Fig. 5.9(c) and (f), excluding suspected shadow segments, are higher potential objects.

After these operations, the well-known PCA based method [95, 96] is employed to recognize the targeting object with respect to its template. Experimental results will be presented in the next section.

Fig. 5.9. Potential objects grouped from segmentation result in (b). (a) is the original image; (b) is the visualized segmentation result. (c) – (f) are potential objects.

5.5. Experimental Results

As compared with the conventional methods, the salient feature of MPCNN is its ease of implementation in digital circuits. The proposed pulse-based neural circuits have been implemented using a Spartan-3A FPGA (XC3SD3400A -4FGG676C) with 3.4 millions of logic gates running at 250MHz clock rate. In this design, 76% of the FPGA resources, i.e. about 2.58 millions of logic gates have been used for the segmentation phase. Fig. 5.10 shows the block diagram of the overall system. For the grouping algorithm, the shadow analysis and the PCA based recognition algorithm, they are implemented on a host PC connecting the FPGA through Ethernet.

Fig. 5.10. Block diagram of the hardware system.

5.5.1. Evaluation of computational complexity

The proposed MPCNN is a parallel oriented design. This drastically reduces its time complexity. In the segmentation algorithm, seed selection and local linking, region-merging by linking are designed in parallelism. Consequently, these computations can be completed in a constant time. To calculate the sizes and mean color for all the regions, the computation is $O(n)$, where O is Landau notation; n is the total number of pixels in an image. In contrast, the time complexity of a typical conventional SRG is determined as $O(n+n\log(n)+mn)$ [87], where m is the number of resultant segments. For a 24-bit-color image with $n=400\times 400$ pixels and $m=600$ segments, the computational requirement may be over 10G clock cycles after computations such as multiplying and square-root solving with floating point values. For JSEG and the method in [90], their time complexities are even higher than the conventional SRG because they employ the region-merging phase of SRG, which is the dominant time-consuming phase [87].

Fig. 5.11 compares the processing speeds of conventional SRG and MPCNN with respect to the resultant number of segments. The conventional SRG algorithm is coded in Visual C++ and implemented on a Pentium4 computer at 3GHz, 12 times of the clock rate of the proposed FPGA. Considering the factor of exponential coordinate, we can observe that the processing times are increased with the number of segments increasing for both methods. But the increasing rates are different. It is not surprising that the parallel MPCNN circuits have a far swifter processing speed than its counterpart. For example, the average processing time of SRG is 2.37 seconds for the image with a range of 50 to 100 segments. This is drastically increased by nearly 10 times to 23.2 seconds for the case of 351 to 400 segments. This reflects the fact that the number of segments is a significant factor to the time complexity of SRG. In contrast, the processing time of MPCNN is only doubled from 15.6 ms to 31.1 ms in the same case study.

Fig. 5.11. Comparison of processing time between SRG and MPCNN with respect to various numbers of resultant segments. ‘o’ for average time of SRG; ‘*’ for average time of MPCNN. Gray ribbons cover the span of all testing cases.

5.5.2. Performance of object detection

In section 5.2, we have demonstrated the impact of the parameters of region-merging phase. As the performance of object detection has direct dependency on the quality of segmentation phase, these parameters also have impact to the object detection phase. Three different case examples are demonstrated in Fig. 5.12(a), 5.12(c) and 5.12(e). In cases 1 - 3, it is assumed that the targets are fighter, storage tanks and military tanks respectively. Table 5.2 summarizes the impact of the segmentation parameters θ_a^d , θ_b^d and θ_c^d . Consider the case of fighter: When the thresholds of color difference are increased by 50%, the *precision* (P value) is increased from 0.45 to 0.48, while the *recall* (R value) is unchanged. This could reduce the degree of over-segmentation and the combination number of potential objects. In Table 5.2, it is observed that the numbers of resultant regions, i.e. N_g , for all cases are decreased as expected. However, overly large parameters increase the risk of missing important parts of the targeting object. This is observed when those parameters are increased by 100%, where the left wing of the plane is merged into shadow as shown in Fig. 5.5. Meanwhile, the *recall* (R value) is decreased from 0.88, obtained in the normal-parameter case, to 0.79 indicating a lower detect rate of the human-labeled object [91].

As mentioned in section 5.4, the targeting object is one of a number of possible combinations of segments fulfilled the predefined geometric constraints. In some cases, the number of potential combinations could be large and shadow analysis would be useful. In Table 5.2, we can also observe the results obtained before and after introducing the shadow analysis, where N_c^{wo} stands for the number of combinations without shadow analysis, and N_c^w stands for the number of combinations with shadow analysis. From the results, it is observed that the shadow analysis reduces the number of possible combinations significantly for cases 1 and 2.

The number of possible combinations is unchanged after shadow analysis in case 3 as no significant shadow can be detected. The corresponding visualized results are illustrated in Fig. 5.12.

Table 5.2. Summary of impact on segmentation & object detection for various cases in Fig 5.12. ($\tau = a, b, c.$)

case name	Impact of segmentation parameters														
	Normal θ_τ^d					$\Delta\theta_\tau^d = (50\%) \cdot \theta_\tau^d$					$\Delta\theta_\tau^d = (100\%) \cdot \theta_\tau^d$				
	P	R	N_g	N_c^w	N_c^{wo}	P	R	N_g	N_c^w	N_c^{wo}	P	R	N_g	N_c^w	N_c^{wo}
Case 1	0.45	0.88	72	16	109	0.48	0.88	66	16	105	0.40	0.79	57	13	78
Case 2	0.42	0.90	86	32	196	0.43	0.88	81	32	182	0.43	0.85	70	28	150
Case 3	0.40	0.82	95	232	232	0.41	0.80	88	204	204	0.38	0.75	69	151	151

N_c^w -- Number of combinations with shadow analysis;

N_c^{wo} -- Number of combinations without shadow analysis.

In PCA-based recognition phase, 2000 aerial images have been utilized, where 100 images are collected for each of the 20 objects. Amongst the 100 images for each targeting object, 80 images are used as templates to constitute the eigenspace of the specific target, while the other 20 images are used for retrieval test. Fig. 5.13 illustrates the PCA method. The potential object extracted from column 1 of Fig. 5.12 is recognized as the targeting fighter because its minimum distance to the templates in the database satisfies the recognition criteria in the eigenspace [95]. In Fig. 5.13, object 2 and object 3 will result in true negative conclusion because the distances of their features to the templates in the database do not satisfy the criteria. The true positive and true negative rates for various objects are summarized in Fig. 5.14. A phenomenon is that the true negative rates of most cases are less than their corresponding true positive rates. The reason is that the images for the experiments

are top-view aerial and satellite photographs. Consequently, they expose less object features of other view point. The higher detection accuracies for objects with salient top-view features, such as the UAV landing mark and flights, also verify this point. For top-view aerial or satellite images, multi- or hyper- spectral data could be one of the solutions to improve the recognition results. The proposed multi-channel PCNN can be conveniently enhanced to deal with images with more bands of spectral information in the future work.

Fig. 5.12. Results of various cases where suspected shadows are highlighted in white color. (a)&(b) for case 1; (c)&(d) for case 2; (e)&(f) for case 3.

Fig. 5.13. PCA based target recognition.

Fig. 5.14. Detection accuracy for various objects.

5.6. Conclusions

In this chapter, a PCNN with multi-channel linking and feeding fields is proposed for color image segmentation. Different from the conventional PCNN, pulse based RBF units are

introduced into the model neurons of PCNN to determine the fast links among neurons with respect to their spectral feature vectors and spatial proximity. These IAF (integrate-and-fire) RBF units utilize the timing of individual pulse produced by the neurons to give competitive effect to reflect the distance between pixels' feature vector so that seeded pixels are determined and homogenous pixels are linked. By mapping each pixel to the neuron of a 2D MPCNN, the segmentation of the input color image can be performed using a FPGA. The results of segmentations have also been utilized to derive potential objects of interest in the image under basic geometric constraints. The possible number of group combinations is refined using a shadow based analysis. Experimental results show that the performance of the proposed MPCNN is comparable to other popular image segmentation algorithms. However, it improves the speed of segmentation drastically. This provides the practical basis for further intelligent processing such as real-time target recognition.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis investigates the pulsed neural networks, which are named as the third generation of neural network models in the literature [14, 18]. Two new models of pulsed neural networks, namely the pulse-based radial basis function (RBF) network and the multi-channel pulse coupled neural network (MPCNN), have been proposed based on the essence of temporal pulses. Their bio-inspired characteristics are utilized to design neural computations that are different from conventional neural networks. Their properties have been analyzed, and their strengths for practical applications have been evaluated experimentally.

Different from the conventional RBF networks, the proposed pulse-based RBF network utilizes the synchronism of temporal pulses generated by model neurons to perform RBF computations. It is highly advantageous due to its parallelism characteristics. Based on a winner neuron selection algorithm, which is designed by employing the rank order of neurons' output pulses, it gives comparable performance when benchmarked with the conventional clustering methods such as k -means and SOM. When it is extended for nonlinear function approximation applications, it outperforms other pulsed neural model reported in [39, 60]. The rank order based paradigm is also extended to deal with some

complex clustering problems such as processing of meshed clusters which is difficult if k -means and SOM are used. Its capability of noise-resistance is satisfactory. The effects of kernel functions based on L_1 -norm, L_2 -norm and L_∞ -norm distance metrics are analyzed to guide the selection of RBF kernel.

The proposed multi-channel pulse coupled neural network (MPCNN) is a 2-D variant of pulsed neural networks. MPCNN is a multi-spectral image segmentation approach that uses the timing of individual pulse produced by the neurons to determine the distances between pixels' feature vectors and their rank order. Consequently, the fast links can be established among neurons with respect to the spectral feature vectors and spatial proximity of mapped pixels. Furthermore, the image segmentation algorithm is designed by utilizing the 2-D structure of PCNN, where neurons' connectivity may transfer in parallel in the 2-D space to establish intra/inter-segment linking. With this vector-oriented design, MPCNN overcomes the shortcoming of the conventional scalar-based PCNN. It has been applied to the processing of color or multi-spectral images. Analysis shows that its sensitivity to the algorithm parameters is significantly reduced as compared to the conventional seeded region growing algorithm. Using common quantitative measures, namely *precision*, *recall* and *F*-measure, the segmentation performance of the proposed MPCNN is comparable to SRG [87] and JSEG [93] algorithms for segmentation of noisy images while its full parallelism can improve the processing speed by up to hundreds of times. Another salient feature of this pulsed neural network is that it can produce satisfactory results without iterative learning. This effectively avoids the potential issues of convergency and time complexity.

This thesis designs the proposed pulsed neural circuits for forward computation, feedback learning and simultaneous 2-D linking on a FPGA. Utilizing temporal pulse behavior, the neural circuits can be integrated with more compactness for hardware implementation in massive parallelism. More specifically, the firing time and rank order of pulses excited by input signals are utilized to compute the vector distance and select the winner neuron in the competitive learning. ART (adaptive resonance theory) and genetic algorithm have been modified to seamlessly merge into pulsed neural circuits based on their suitability to digital computation. These schemes significantly overcome the bottleneck of computational burden of sequential-code-based methods. The proposed methodology provides a basis for further biologically inspired methods that require real-time computing. We have demonstrated experimentally several real-time applications, including satellite image processing, color image segmentation for object detection, real-time object-tracking in video recording, and friction estimation of a precision linear stage. The analyses on computational complexity and performance evaluation show that the proposed approaches outperform their counterparts such as k -means, BP network, SRG and JSEG.

6.2 Recommendation for Future Research

Based on the work in this thesis, some future research directions are recommended as follows:

- The research on pulsed neural networks is motivated by the phenomena of animals' perception systems. Thus most of the research works focus on artificial perception, such as information processing and pattern analysis. The temporal-pulse-based models are seldom designed as neuro-controllers for closed loop systems. On the other hand, pulsed neural networks are more suitable for hardware implementation in contrast to real-valued neural

models. This real-time neural computation provides a potential strength for nonlinear controller design as a future research direction.

- Although heuristic representations of temporal pulse trains could reflect biological reality, they may not be compatible with the popular mathematical tools for theoretical investigation. This point could be one of the major reasons that limit the pulsed neural net from applications such as control systems, where more theoretical analyses are needed. Hence investigation in preliminary mathematics for this area needs to be conducted. To strengthen the research, the definitions for standard representations and exploration of systematic mathematical tools are recommended in the future work.

- For the potential real-time applications, research works on temporal pattern analysis such as auditory processing and Electroencephalogram (EEG) processing are recommended. The nature of these signals is of time sequence, which is apt to be compatible with the temporal pulse based neural model. Meanwhile, the hardware based pulsed neural circuits are able to fulfill the speed requirement of these real-time applications.

- For future implementation, the COMS based neuromorphic design is a potential scheme. The salient point of this design is that the sensor array and neural circuits can be integrated on the same chip. The computing about the data stream from sensor can be conducted on the fly and the degree of real-time processing will be improved. Its low power consumption will make it suitable for pocket systems. Another potential scheme is based on the modern graphics processing unit (GPU). Its architecture of multiple processor cores can be utilized for parallel neural computing. Some latest commercial products, e.g. NVIDIA Tesla, are

equipped with 240 on-chip processor cores. Despite its drawback of power consumption, GPU could be a powerful platform for neural models to perform parallel soft computing in the future.

Author's Publications

- [1] H. L. Zhuang, K. S. Low and W. Y. Yau, "Multi-channel pulse coupled neural network based color image segmentation for object detection," *IEEE Trans. on Industrial Electronics*, 2011. (in press)
- [2] H. L. Zhuang, K. S. Low and W. Y. Yau, "On-chip pulse based parallel neural circuits for object-tracking system," *Electronics Letters*, vol. 46, no. 9, pp. 614-616, 2010.
- [3] H. L. Zhuang, K. S. Low and W. Y. Yau, "A pulsed neural network with on-chip learning and its practical applications", *IEEE Trans. on Industrial Electronics*, vol. 54, no. 1, pp. 34-42, 2007.
- [4] B. Zhao, X. Y. Zhang, S. S. Chen, K. S. Low and H. L. Zhuang, "A 64×64 CMOS image sensor with on-chip moving object detection and localization," *IEEE Trans. on Circuits & Systems for Video Technology*, 2011. (in press)
- [5] H. L. Zhuang, K. S. Low and W. Y. Yau, "Real Time Runway Detection in Satellite Images Using Multi-Channel PCNN," *IEEE Trans. on Geoscience and Remote Sensing*. (submitted)
- [6] H. L. Zhuang, K. S. Low and W. Y. Yau, "A multiplier-less GA optimized pulsed neural network for satellite image analysis using a FPGA," in *Proc. 3rd IEEE Conference on Industrial Electronics and Applications (ICIEA 2008)*, pp. 302-307, 2008
- [7] K. S. Low, V. Krishnan, H. L. Zhuang and W. Y. Yau, "On-chip Genetic Algorithm Optimized Pulse Based RBF Neural Network for Unsupervised Clustering Problem,"

Advances in Natural computation, Lecture Notes in Computer Science, Springer-Verlag GmbH, vol. 4222/2006, pp. 851-860, 2006.

- [8] H. L. Zhuang, K. S. Low and W. Y. Yau, “A Pulse-based RBF neural model for unsupervised clustering,” in *Proc. 3rd International Conference on Computational Intelligence, Robotics and Autonomous Systems*, Singapore, 13-16 December 2005.

Bibliography

- [1] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, vol. 65, no. 6, pp. 386-408, 1958.
- [2] M. L. Minsky and S. A. Papert, *Perceptrons*, Cambridge, MA: MIT Press, 1969.
- [3] E. D. Adrian, *The Basis of Sensation*, New York: Norton, 1928.
- [4] J. Gautrais and S. Thorpe, "Rate coding versus temporal order coding: A theoretical approach," *Biosystems*, vol. 48, no.1, pp. 57-65, 1998.
- [5] M. Abeles, H. Bergman, E. Margalit and E. Vaadia, "Spatio-temporal firing pattern in the frontal cortex of behaving monkeys," *Journal of Neurophysiology*, vol. 70, no. 4, pp. 1629-1638, 1993.
- [6] D. H. Perkel and T. H. Bullock, "Neural coding," *Neuroscience Research Program Bulletin*, vol. 63, no. 2, pp. 221-348, 1968.
- [7] M. Meister and M. J. Berry II, "The neural code of the retina," *Neuron*, vol. 22, no. 3, pp. 435-450, 1999.
- [8] L. F. Abbott, J. A. Varela, K. Sen and S. B. Nelson, "Synaptic depression and cortical gain control," *Science*, vol. 275, no. 5297, pp. 200-224, 1997.
- [9] A. M. Thomson, "Facilitation, augmentation and potentiation at central synapses," *Trends in Neurosciences*, vol. 23, no. 7, pp. 305-312, 2000.

- [10] R. C. deCharms and M. M. Merzenich, "Primary cortical representation of sounds by the coordination of action-potential timing," *Nature*, vol. 381, pp. 610-613, 1996.
- [11] M. Abeles, *Corticonics: Neural circuits of the cerebral cortex*. Cambridge: Cambridge University, 1991.
- [12] R. Eckhorn, R. Bauer and W. Jordan, "Coherent oscillations: A mechanism of feature linking in visual cortex? Multiple electrode and correlation analysis in the cat," *Biol. Cybern.*, vol.60, no. 1, pp. 121-130, 1988.
- [13] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison Wesley, 1989.
- [14] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659-1671, 1997.
- [15] J. J. Hopfield, "Pattern recognition computation using action potential timing for stimulus representations," *Nature*, vol. 376, no. 7, pp. 33-36, 1995.
- [16] R. Eckhorn, H. J. Reitboeck, M. Amdt and P. W. Dicke, "Feature linking via synchronization among distributed assemblies: Simulation of results from cat cortex," *Neural Computation*, vol. 2, no. 3, pp. 293-307, 1990.
- [17] J. L. Johnson, "Pulse-coupled neural nets: Translation, rotation, scale, distortion, and intensity signal invariance for images," *Applied Optics*, vol. 33, no.26, pp. 6239-6253, 1994.
- [18] W. Maass and C. M. Bishop, *Pulsed Neural Networks*, MIT Press, Cambridge, MA, 1999.
- [19] G. A. Carpenter, S. Grossberg and D. Rosen, "ART2-A: an adaptive resonance algorithm for rapid category learning and recognition," *Neural Networks*, vol. 4, no. 3, pp. 493-504, 1991.

- [20] H.L. Zhuang, K. S. Low and W. Y. Yau, "A pulsed neural network with on-chip learning and its practical applications," *IEEE Trans. on Industrial Electronics*, vol. 54, no.1, pp. 34-42, 2007.
- [21] H. L. Zhuang, K. S. Low and W. Y. Yau, "A Pulse-based RBF Neural Model for Unsupervised Clustering," in *Proc. 3rd International Conference on computational Intelligence, Robotics and Autonomous Systems*, Singapore, 13-16 December, 2005.
- [22] H. L. Zhuang, K. S. Low and W. Y. Yau, "Multi-channel pulse coupled neural network based color image segmentation for object detection," *IEEE Trans. on Industrial Electronics*, 2010. (in press)
- [23] H. L. Zhuang, K. S. Low and W. Y. Yau, "A multiplier-less GA optimized pulsed neural network for satellite image analysis using a FPGA," in *Proc. 2008 3rd IEEE Conference on Industrial Electronics and Applications (ICIEA 2008)*, pp. 302-307, 2008.
- [24] H. L. Zhuang, K. S. Low and W.-Y. Yau, "On-chip pulse based parallel neural circuits for object-tracking system," *Electronics Letters*, vol. 46, no. 9, pp. 614-616, 2010.
- [25] H. L. Zhuang, K. S. Low and W. Y. Yau, "Real Time Runway Detection in Satellite Images Using Multi-Channel PCNN," *IEEE Trans. on Geoscience and Remote Sensing*. (submitted).
- [26] L. Lapique, "Recherches quantitatives sur l' excitation electrique des nerfs traitee comme une polarization," *Journal of Physiology and Pathololgy*, vol. 9, pp. 620-635, 1907.
- [27] H. C. Tuckwell, *Introduction to theoretical neurobiology*, vol. 1 and vol. 2, Cambridge: Cambridge University Press, 1988.

- [28] W. Maass, "Lower bound for the computational power of networks of spiking neurons," *Neural Comput.*, vol. 8, no. 1, pp. 1-40, 1996.
- [29] W. Gerstner, "Time structure of the activity in neural network models," *Physical Review E*, vol. 51, pp. 738-758, 1995.
- [30] W. Maass, "On the relevance of time in neural computation and learning," *Theoretical Computer Science*, vol. 261, pp. 157-178, 2001.
- [31] G. Kuntimad and H. S. Ranganath, "Perfect Image Segmentation Using Pulse Coupled Neural Networks," *IEEE Trans. on Neural Networks*, vol.10, no. 3, pp. 591-598, 1999.
- [32] T. J. Sejnowski, "Time for a new neural code?," *Nature*, vol. 376, no. 7, pp. 21-22, 1995.
- [33] S. Thorpe, A. Delorme and R. Van Rullen, "Spike-based strategies for rapid processing," *Neural Networks*, vol. 14, no. 6, pp. 715-725, 2001.
- [34] S. M. Bohte, H. L. Poutré and J. N. Kok, "Unsupervised Clustering With Spiking Neurons By Sparse Temporal Coding and Multilayer RBF Networks," *IEEE Trans. on Neural Networks*, vol.13, no. 2, pp. 426-435, 2002.
- [35] W Gerstner, R. Kempter, J. L. van Hemmen and H. Wagner, "A neuronal learning rule for sub-millisecond temporal coding," *Nature*, vol. 383, no. 1, pp. 76-78, 1996.
- [36] N. E. Cotter and O. N. Mian, "A pulsed neural network capable of universal approximation," *IEEE Trans. on Neural Networks*, vol. 3, no. 2, pp. 308-314, 1992.
- [37] T. Natschläger and B. Ruf, "Spatial and temporal pattern analysis via spiking neurons," *Network: Comput. Neural Syst.*, vol. 9, pp. 319-332, 1998.
- [38] B. Ruf and M. Schmitt, "Self-Organization of Spiking Neurons Using Action Potential Timing," *IEEE Trans. on Neural Networks*, vol. 9, no. 3, pp. 575-578, 1998.

- [39] S. M. Bohte, J. N. Kok and H. L. Poutré, “Error-backpropagation in temporally encoded networks of spiking neurons,” *Neurocomputing*, vol. 48, pp. 17-37, 2002.
- [40] D. E. Rumelhart, G. E. Hinton and R. J. Williams, “Learning internal representations by error propagation,” in *Parallel distributed processing: exploration in the microstructure of cognition*, vol. 1, D. E. Rumelhart and J. M. McClelland, Eds, Cambridge, MA:MIT Press, pp. 318-362, 1986.
- [41] J. Storck, F. Jäkel and G. Deco, “Temporal clustering with spiking neurons and dynamic synapses: towards technological applications,” *Neural Networks*, vol. 14, pp. 275-285, 2001.
- [42] C. Näger, J. Storck and G. Deco, “Speech recognition with spiking neurons and dynamic synapses: a model motivated by the human auditory pathway,” *Neurocomputing*, vol. 44, pp. 937-942, 2002.
- [43] J. L. Johnson, “PCNN Models and Applications,” *IEEE Trans. on Neural Networks*, vol. 10, no. 3, pp. 480-498, 1999.
- [44] R. Eckhorn, “Neural Mechanism of Scene Segmentation: Recordings from the Visual Cortex Suggest Basic Circuits for Linking Field Models,” *IEEE Trans. on Neural Networks*, vol. 10, no. 3, pp. 464-479, 1999.
- [45] W. Singer, “Coherence as an organizing principle of cortical functions,” *International review of neurobiology*, vol. 37, pp. 153-183, 1994.
- [46] M. Yide, Z. Kun and Q. Chunliang, “Study on self-adaptive pulse coupled neural network and its application in field of image processing,” *Journal of System Simulation*, vol. 20, no. 11, pp. 2897-2900, 2008.
- [47] Z. Wang, Y. Ma, F. Cheng and L. Yang, “Review of pulse-coupled neural networks,” *Image and Vision Computing*, vol. 28, no. 1, pp. 5-13, January 2010.

- [48] X. Gu, "Feature extraction using unit-linking pulse coupled neural network and its applications," *Neural Process Letter*, vol. 27, no. 1, pp. 25-41, 2008.
- [49] Y. Lu, J. Miao, L. Duan, Y. Qiao and R. Jia, "A new approach to image segmentation based on simplified region growing PCNN," *Applied Mathematics and Computation*, vol. 205, no. 2, pp. 807-814, 2008.
- [50] R. D. Stewart, I. Fermin and M. Oppner, "Region Growing with Pulse-coupled Neural Networks: An Alternative to Seeded Region Growing," *IEEE Trans. on Neural Networks*, vol. 13, no. 6, pp. 1557-1562, 2002.
- [51] J. A. Karvonen, "Baltic sea ice SAR segmentation and classification using modified pulse-coupled neural networks," *IEEE Trans. Geoscience and Remote Sensing*, vol. 42, no. 7, pp. 1566-1574, 2004.
- [52] Q. Liu, Y. Ma and Z. Qian, "Automated image segmentation using improved PCNN model based on cross-entropy," *Journal of Image and Graphics*, vol. 10, pp. 579-584, 2005.
- [53] Q. Liu and Y. Ma, "A new algorithm for noise reducing of image based on PCNN time matrix," *Journal of Electronics and Information Technology*, vol. 30, no. 8, pp. 1869-1873, 2008.
- [54] Z. Wang, Y. Ma, "Medical image fusion using m-PCNN," *An International Journal on Multi-Sensor Multi-Source Information Fusion*, vol. 9, no. 2, pp. 176-185, 2008.
- [55] X. D. Gu, D. H. Yu and L. M. Zhang, "Image shadow removal using pulse coupled neural networks," *IEEE Trans. on Neural Networks*, vol. 16, no. 3, pp. 692-698, 2005.
- [56] R. Zhao, Y. Ma and Z. Kun, "Three-state cascading pulse coupled neural network and the application in finding shortest paths," *System Engineering and Electronics*, vol. 30, no. 9, pp. 1785-1789, 2008.

- [57] R. Forgac and I. Mokris, "Feature generation improving by optimized PCNN," in *Proc. 6th International Symposium on Applied Machine Intelligence and Informatics*, pp. 203-207, 2008.
- [58] L. Ji, Z. Yi, L. Shang and X. Pu, "Binary fingerprint image thinning using template-based PCNNs," *IEEE Trans. System Man and Cybernetics – Part B: Cybernetics*, vol. 37, no. 5, pp.1407-1413, 2007.
- [59] A. P. Timoszczuk and E. F. Cabral, "Speaker recognition using pulse coupled neural networks," in *Proc. International Joint Conference on Neural Network*, pp.1965-1969, 2007.
- [60] N. Iannella and A. D. Back, "A spiking neural network architecture for nonlinear function approximation," *Neural Networks*, vol. 14, pp. 933-939, 2001.
- [61] S. Kuroyanagi and A. Iwata, "An competitive learning pulsed neural network for temporal signals," in *Proc. 9th Int. Conf. on Neural Information Processing (ICONIP'02)*, vol.1, pp. 348-352, 2002.
- [62] D. L. Wang and G. J. Brown, "Separation of speech from Interfering sounds based on oscillatory correlation," *IEEE Trans. on Neural Networks*, vol. 10, no. 3, pp.684-697, 1999.
- [63] Y. Ota, "VLSI structure for static image processing with pulse-coupled neural network," in *Proc of the 28th Annual Conference of Industrial Electronics Society*, vol. 4, pp. 3221-3226, 2002.
- [64] C. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley, 1989.
- [65] D. J. Mayes, A. Hamilton, A. F. Murray and H. M. Reekie, "A pulsed VLSI radial basis function chip," in *Proc. IEEE Int. Symposium on Circuits and Systems*, vol. 3, pp. 297-300, Atlanta, 1996.

- [66] H. Hikawa, "Frequency-based multilayer neural network with on-chip learning and enhanced neuron characteristics," *IEEE Trans. on Neural Networks*, vol. 10, no. 3, pp. 545-553, 1999.
- [67] H. Hikawa, "A digital hardware pulse-mode neuron with piecewise linear activation function," *IEEE Trans. on Neural Networks*, vol. 14, no. 5, pp.1028-1037, 2003.
- [68] R. A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, vol. 7, pp. 179-188, 1936.
- [69] D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex System*, vol. 2, pp. 321-355, 1988.
- [70] M. Marchesi, G. Orlandi, F. Piazza and A. Uncini, "Fast neural networks without multipliers," *IEEE Trans. on Neural Networks*, vol. 4, no. 1, pp. 53-62, 1993.
- [71] C. Aporntewan and P. Chongstitvatana, "A hardware implementation of the compact Genetic Algorithm," in *Proc. 2001 IEEE Congress on Evolutionary Computation*, pp. 624-629, Seoul, South Korea, 2001.
- [72] E. M. Hossam, I. A. Kadragei and Y. Y. Hanafi, "Hardware implementation of genetic algorithm on FPGA," in *Proc. 21th National Radio Science Conference (NRSC 2004)*, Egypt, 2004.
- [73] W. Tang and L. Yip, "Hardware implementation of genetic algorithm on FPGA," in *Proc. 47th IEEE Intl. Midwest Symposium on circuits and systems*, pp. 549-552, Hiroshima, Japan, 2004.
- [74] K. S. Low and M. T. Keck, "Advanced precision linear stage for industrial automation applications," *IEEE Trans. on Instrumentation and Measurement*, vol. 52, no. 3, pp. 785-789, 2003.

- [75] M. T. Keck, *High Precision Optimum Control of a linear motor drive using Genetic Algorithm*, MEng thesis, Nanyang Technological University, Singapore, 2000.
- [76] W. J. MacLean, "An Evaluation of the Suitability of FPGAs for Embedded Vision Systems," in *Proc. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005.
- [77] J. Díaz, E. Ros, F. Pelayo, E. M. Ortigosa, and S. Mota, "FPGA-based real-time optical-flow system," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 16, no. 2, pp. 274-279, 2006.
- [78] X. Zhang, "An Object Tracking System Based on Human Neural Pathways of Binocular Motor System," in *Proc. 9th Int. Conf. on Control, Automation, Robotics and Vision*, pp. 285-292, Singapore, 2006.
- [79] P. Li, "A clustering-based color model and integral images for fast object tracking," *Singal Processing: Image Communication*, vol. 21, pp. 676-697, 2006.
- [80] M. Francetič, M Nagode and B Nastav, "Hierarchical Clustering with Concave Data Sets," *Advances in Methodology and Statistics*, vol. 2, no. 2, pp. 173-193, 2005.
- [81] P. Lichtsteiner, C. Posch and T. Delbruck, "A 128x128 120dB 15us Latency Asynchronous Temporal Contrast Vision Sensor," *IEEE Journal of Solid-state Circuits*, vol. 43, no. 2, pp. 566-576, 2008.
- [82] M. Estlick, M. Leeser, J. Theiler and J. J. Szymanski, "Algorithmic transformation in the implementation of K-means clustering on reconfigurable hardware," in *Proc. of ACM/SIGDA Int. Symposium on FPGA*, pp. 103-110, 2001.
- [83] *Handel C Language Reference Manual*, Celoxica Limited, 2005.

- [84] P. Chu and R. E. Jones, "Design Techniques of FPGA Based Random Number Generator," in *Proc. Military and Aerospace Applications of Programming Devices and Techniques Conf.*, Laurel, MD. 1999.
- [85] A. D. Kulkarni, *Computer Vision and Fuzzy-Neural Systems*, Prentice Hall, 2001.
- [86] R. Adams and L. Bischof, "Seeded region growing," *IEEE Trans. on Pattern Anal. Machine Intell.*, vol. 16, no. 6, pp. 634-643, 1994.
- [87] F. Y. Shih and S. Cheng, "Automatic seeded region growing for color image segmentation," *Image Vision Computing*, vol. 23, no. 10, pp. 877-886, 2005.
- [88] A. Tremeau and N. A. Borel, "Region growing and merging algorithm to color segmentation," *Pattern Recognition*, vol. 30, no. 7, pp. 1191-1203, 1997.
- [89] K. Bhoyar and O. Kakde, "Color image segmentation based on JND color histogram," *International Journal of Image Processing*, vol. 3, no. 6, pp. 283-292, 2010.
- [90] T. Celik and T. Tjahjadi, "Unsupervised colour image segmentation using dual-tree complex wavelet transform," *Computer Vision and Image Understanding*, vol. 114, no. 7, pp. 813-826, 2010.
- [91] D. R. Martin, C. C. Fowlkes and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, , vol. 26, no. 5, pp. 530-549, 2004.
- [92] D. Martin, C. Fowlkes, D. Tal and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. IEEE Int. Conf. Comput. Vision*, vol. 2, pp. 416-423, 2001.

- [93] Y. Deng and B. S. Manjunath, "Unsupervised segmentation of color-texture regions in images and video," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 8, pp. 800-810, 2001.
- [94] J. Battle, A. Casals, J. Freixenet and J. Marti, "A review on strategies for recognition natural object in colour images of outdoor scenes," *Image and Vision Computing*, vol. 18, no. 6, pp. 515-530, 2000.
- [95] S. K. Nayar, S. A. Nene and H. Murase, "Real-time 100 object recognition system," in *Proc. IEEE Int. Conf. Robotics and Automation*, Minneapolis, Minnesoda, pp. 2321-2325, 1996.
- [96] M. Jager and F.A. Hamprecht, "Principal component imagery for the quality monitoring of dynamic laser welding processes," *IEEE Trans. on Industrial Electronics*, vol. 56, no. 4, pp. 1307-13, 2009.