New self-organizing algorithms for topological maps

Xu, Pengfei

2006

Xu, P. (2006). New self-organizing algorithms for topological maps. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/20865>

<https://doi.org/10.32657/10356/20865>

New Self-organizing Algorithms for Topological Maps

Xu Pengfei

School of Electrical & Electronic Engineering

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

2006

Statement of Originality

I hearby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

Date

Xu Pengfei

Acknowledgments

I wish to express my deepest appreciation to Associate Professor Chang Chip-Hong. I was fortunate to have him as my supervisor, and benefited enormously from his academic expertise, his inspiring guidance and his constant support. The many discussions with him helped me greatly in shaping the research ideas presented in this thesis.

I am also very grateful to Professor Andrew Paplinski from School of Computer Science and Software Engineering of Monash University, Australia. His pertinent suggestions, critical appraisal and careful revision of my manuscript in preparation for the submission to IEEE Transactions on Systems, Man and Cybernetics, Part B has helped to improve the materials presented in Chapter 4 greatly.

Thanks to the Visual Information Processing research group of Edith Cowan University, Australia, for their kind release of the database of over 6000 images collected and built over year of onerous effort for the face segmentation and tracking research.

I am also grateful to Deng Xiaoyun, for her support, collaboration and discussion during the course of this project.

Acknowledgments

Special thanks go to Dr. Gu Jiangmin, Xu Fei, Shao Yu, Yajuan and other graduate students for sharing the memorable time together.

My gratitude also goes to my family for their understanding, support, and encouragements.

Summary

Self-organizing Maps (SOM) were developed by Kohonen in late 1980's in response to the growing amount of large data sets required to represent a problem for thorough analysis with standard tools and techniques. SOMs are widely used in many applications including machine vision and image analysis, speech analysis and recognition, robotics, telecommunications, signal processing and radar measurement, process control, and so on. Despite their popularity in engineering applications and statistical data analysis, the basic SOM and its variants possess some intrinsic weaknesses that limit their dominance in traditional and contemporary learning tasks such as clustering, classification and density estimation.

The aim of this thesis is to bring together new algorithms and insights to construct a framework of self-organizing topological map methods for dealing with stochastic data in a robust and efficient way. The underlying computation algorithms and structures are demonstrated on, and in part motivated by some image processing tasks which we contemplate that the adaptive process of the learning model to be developed can be beneficial. Towards this end, we have proposed a new competitive learning scheme for SOM to sustain stable performance and fast convergence over a dynamic range of control parameters. The proposed frequency-sensitive self-organizing maps (FS-SOM) harmonically blends the neighborhood adaptation with the neuron dependent frequency sensitive learning model, the global butter-

Summary

fly permutation sequence for input randomization, and the reinitialization of dead neurons to harness effective utilization of neurons. The net effect of overcoming the order dependent learning is a significant improvement in adaptation and a well-ordered codebook. More importantly, the performance and topological order of the SOM are now dramatically robust against variations in essential control parameters.

An on-line self-organizing topological tree (SOTT) is also proposed to speed up the best matching neuron search for both the training and testing phases of the SOM at work. The computational complexity of the proposed SOTT is $O(\log N)$ as opposed to $O(N)$ for the best known SOM. The experiment results demonstrate that the reconstruction performance of SOTT is comparable to the full-search SOM and its computation time is much shorter than the full-search SOM and other vector quantizers. In addition, multi-resolution mapping is spawned as a positive side effect of the tree structure of SOTT.

We also proposed a novel hybrid clustering technique that has demonstrated to be computational efficient and possesses good scalability. To fully exploit the advantages of online learning, and reduced computational complexity of the SOTT architecture, a new linkage metric is introduced for the hybrid clustering algorithm. This linkage metric can be updated online during the partition clustering stage and does not need to wait for the patterns to be labeled.

Two applications have been explored to appraise and fine-tune the proposed learning algorithms. We explore methods to integrate our proposed FS-SOM and SOTT on color halftoning, which is an eminent postprocessing step after color quantization to produce high quality rendering of continuous-tone 24-bit digital color

images. The global butterfly jumping sequence (GBJS) with progressive spatial resolution property is used to provide omnidirectional error diffusion method for color halftoning. Albeit the simplicity of the proposed GBJS-based error-diffusion method, subjective evaluation of the color rendered images shows promising perceptual quality improvement. Two new training and testing schemes have also been proposed for evaluating the classification ability of SOTT in pixel-based skin color detection and they are compared against several well-known classifiers in the same application. The experimentation concludes that the power of self-organizing topological clustering can be exploited to reduce the memory requirements of the effective Bayesian classifier. For best classification performance, it is necessary to further investigate how the training should be fine-tuned so that more trained neurons are assigned to the decision border where the class distribution intersect to minimize misclassification.

In summary, self-organizing algorithms have frequently been asserted to require a large number of training steps resulting in unpredictably slow convergence and often onerous calibration before actual application. This research envisages the genuine operational deficiency and characteristics of self-organizing topological maps to improve the learning mechanisms of self-organizing topological maps and create several new facets to their applications.

Contents

Acknowledgments	i
Summary	iii
List of Figures	xi
List of Tables	xv
List of Abbreviations	xix
Chapter 1: Introduction	1
1.1 Motivation	1
1.2 Research Objectives	6
1.3 Research Originality	7
1.4 Organization of the Thesis	10

Chapter 2: Background and Literature Review	13
2.1 Vector Quantization	14
2.2 Color Quantization and Halftoning	20
2.3 Clustering	27
2.4 Self-organizing Maps	32
2.5 Chapter Summary	40
Chapter 3: Frequency-sensitive Self-organizing Map	42
3.1 Introduction	42
3.2 Global Butterfly Jumping Sequence	46
3.3 Frequency-sensitive Self-organizing Maps (FS-SOM)	53
3.4 Simulation Results and Discussion	62
3.5 Chapter Summary	78
Chapter 4: Self-organizing Topological Tree	80

Contents

4.1	Introduction	80
4.2	Structure of SOTT and Related Terminologies	84
4.3	Training Algorithm of SOTT	87
4.4	Hybrid Clustering Algorithm on SOTT	95
4.5	Simulation Results	98
4.6	Chapter Summary	114
 Chapter 5: Color Halftoning and Skin Color Classification		115
5.1	Color Halftoning	116
5.2	Skin Classification	130
5.3	Chapter Summary	144
 Chapter 6: Conclusions and Future Work		146
6.1	Conclusions	146
6.2	Recommendations for Future Work	150

Contents

Author’s publications	155
Bibliography	157

List of Figures

2.1	Voronoi constellation in 2D space	15
2.2	A taxonomy of clustering approaches [74]	27
2.3	Structure of a two dimensional SOM network	33
2.4	Major steps of SOM training algorithm	35
2.5	Taxonomy of proposed techniques, attributes and applications studied in this thesis	41
3.1	Basic operation of global butterfly jumping sequence	49
3.2	Illustration of global butterfly jumping sequence	50
3.3	Effect of global butterfly jumping sequence	56
3.4	Comparison of different learning rate functions.	59
3.5	Test images: (a) Lena, (b) baboon and (c) pepper	63

List of Figures

3.6	The color distributions of (a) Lena (b) Baboon and (c) Pepper . . .	64
3.7	Resultant 1D topological maps of a 64-neuron FS-SOM	64
3.8	Result images of FS-SOM	66
3.9	Mean change of neuron weights during the training process. . . .	68
3.10	Effects of varying ϵ on the performance of FS-SOM and SOM. . . .	69
3.11	Learning tracks of FS-SOM with and without reinitialization	70
3.12	Palettized image quality under different combinations of β and κ . .	75
3.13	Topology quality metric, J under different combinations of β and κ . .	76
4.1	Structure of SOTT (a) 2D quad tree [91] (b) 1D string	85
4.2	Major steps of the SOTT training algorithm	88
4.3	Algorithm for winning path search	90
4.4	Overview of hybrid clustering algorithm	96
4.5	512×512 gray-level test images.	99
4.6	1D string color map of a 3-layer SOTT.	103

List of Figures

4.7	1D string color map of SOM.	104
4.8	Topology order J of SOM and SOTT	105
4.9	Lena images reconstructed from different layers of SOTT	105
4.10	data sets for clustering	110
4.11	Clustering results of our proposed SOTT-hybrid algorithm	111
4.12	Computation time versus size of the database	112
4.13	Clustering results for different number of initial clusters	113
5.1	Block diagram of error diffusion	119
5.2	Floyd-Steinberg error filter.	120
5.3	Test color image: Pool	124
5.4	Comparison of different color reduction algorithms	125
5.5	Images using proposed method under raster and GBS sequence	127
5.6	Frequency domain distortion for luminance of “Pepper” image	128
5.7	ECU face image database example	138

List of Figures

5.8	ROC curves of skin color classifiers	140
5.9	Codebook obtained after SOTT training	141
5.10	Results of different skin color pixel classifiers	142

List of Tables

3.1 Butterfly permutation order for input of a tile of 8×8 pixels 48

3.2 Correspondence between $s_{2(n-i-1)+1} s_{2(n-i-1)}$ and x_i, y_i 52

3.3 Reconstruction quality of FS-SOM 72

3.4 Reconstruction speed of FS-SOM 74

3.5 Mean and standard deviation of PSNR and J value 77

3.6 Compression ratio of using CALIC 78

4.1 Test parameters of SOM and SOTT 100

4.2 Performance of SOTT with varying codebook size 101

4.3 Performance of SOTT for different images 102

4.4 Performance of SOTT at intermediate layers 106

List of Tables

4.5 Performance of SOTT at different parameter setting 107

5.1 Objective measurements of different algorithms 129

5.2 ECU data set used for training and testing of skin classifiers 137

5.3 Normalized speed and memory usage of different classifiers 143

List of Abbreviations

AHC	Agglomerative Hierarchical Clustering
BT-SOM	Binary Tree-structured Self-Organizing Map
CDR	Correct Detection Rate
CL	Competitive Learning
CMYK	Cyan, Magenta, Yellow and Black
ED	Error Diffusion
EM	Expectation-Maximization
FDR	False Detection Rate
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FSCL	Frequency Sensitive Competitive Learning
FSO	Frequency Sensitive Organization
FS-ED	Floyd-Steinberg Error Diffusion
FS-SOM	Frequency Sensitive Self-Organizing Map
GLA	Generalized Lloyd Algorithm, i.e., LBG
GBJS	Global Butterfly Jumping Sequence
GBJS-ED	Global Butterfly Jumping Sequence based Error Diffusion
LBG	Linde-Buzo-Gray algorithm, i.e., GLA
LMS	Least Mean Squares
LVQ	Learned Vector Quantization
MSE	Mean Square Error

List of Abbreviations

PSNR	Peak Signal-to-Noise Ratio
ROC	Receiver Operating Characteristic
SOM	Self-Organizing Map
SOTT	Self-Organizing Topological Tree
TSVQ	Tree-Structured Vector Quantization
TS-SOM	Tree-Structured Self-Organizing Map
VQ	Vector Quantization
WMSE	Weighted Mean Square Error
WTA	Winner-Take-All
WTM	Winner-Take-Most

Chapter 1

Introduction

1.1 Motivation

Scientists and engineers have long been fascinated by the way human brain reasons and processes information. Since the inception of the first digital computer, perpetual attempts have been made to create computational model that could best mediate specific types of behavior in the human brain in order to automate tasks that human can accomplish spontaneously and with greater success than machine. The examination of human brain [69, 131] shows that it is dominated by the mere 2mm thick yet enormously complex structure of cerebral cortex, where different sensory inputs are mapped onto corresponding areas of it in an orderly manner. Neurobiologists [143] believe that these cortical maps are not completely predetermined genetically, but exactly how they are sketched in during the development of

the nervous system remains unclear. The following hypotheses on the formation of cortical maps have been advanced with partial experimental evidence of their own [63]: 1. The target (postsynaptic) structure possesses addresses (chemical signals) that are actively searched by the ingrowing connection (axon). 2. Starting from zero, an informationless target structure self organizes using learning rules and system interactions. 3. Axons, as they grow, physically maintain neighborhood relationships to arrive at the target structure already topographically arranged. 4. Axons grow out in a topographically arranged time sequence, and connected to a target structure that is generated in a matching temporal fashion. Once a cortical maps have been formed, they remain “plastic” to some degree and are responsive to future changes in the environment or sensors. These hypotheses have inspired us to scrutinize and explore more intriguing learning algorithms and computational structures of a special class of artificial neural computational model called the Self-organizing Map (SOM) under specific learning tasks like image processing.

The derivatives of SOM to be studied in this thesis fall under the unsupervised learning paradigm [88]. The SOM belongs to the class of unsupervised artificial neural network. In its basic form it produces a similarity graph of input data by converting the nonlinear statistical relationship between high-dimensional input representations into simple geometric relationship of their data points, usually a low-dimensional array of neurons. The mapping is able to preserve the topological relations while performing a dimensionality reduction of the input space [88]. The SOM resembles the information-processing infrastructure of cerebral cortex. The neighboring cells in a SOM compete in their activities through mutual lateral interaction, and develop adaptively into specific detectors of different signal patterns. Thus, the SOM has been able to spontaneously create spatially organized “internal

representations” to provide a means of abstraction, clustering and visualization of the original data. It has been demonstrated that the self-organization processes can help to discover semantic relationships in sentences [88]. Due to its simple form, online learning nature, nonlinear dimensionality reduction and topology-preserving mapping property, SOMs have been used with phenomenal success in a diversity of fields including computer vision and image analysis [48, 129], speech analysis and recognition [71, 72, 121], robotics, telecommunications, signal processing and radar measurement, process control, and so on [5, 112, 157]. Over 5300 research papers on SOMs have been published since it was invented in 1981 [86]. The spatial segregation of responses and their organization into topologically related subsets of SOM lend itself nicely into a massively parallel computing structure. With the advent of VLSI and ULSI technology, it has become technically viable to implement a cohort of hundreds and thousands of inter-communicated processing elements (cells) to increase the learning speed of SOM to orders of magnitude greater than their traditional rivals. What is even more intriguing of SOM over other feedforward and feedback networks is its graceful degradation of performances [88]. The entire network continues to function, albeit at degraded output quality, even if half the number of cells are defective. Such a highly distributive and scalable architecture make it also attractive for pixel- and region-based image processing functionals like color quantization, feature detection, pattern recognition, etc. encountered in classical human-machine interface problem.

Although the prowess of the basic SOM to reasoning of difficult stochastic signals has been well demonstrated, full exploitation of the online learning and topology-preserving capability of domain specific applications like computer vision, color quantization and clustering requires new insight into the topological structure and self-organizing mechanism to complement its ability to automate the analysis and

extraction of visual features. For example, if SOM were to be targeted for pattern recognition problems, their classification accuracy can be multiplied if the cells are fine-tuned using supervised learning principles. Such modified SOM learning algorithm is known as Learning Vector Quantization (LVQ) [87, 88]. Although large maps are feasible, practical implementation favors hierarchical systems composed of smaller maps. Examples of modification to the basic SOM in this perspective were found in [91, 90]. Recently, some extensions of SOMs, such as Bayesian self-organizing map (BSOM) [162] and self-organizing mixture network (SOMN) [163], are derived for performing density estimation and reported to outperform the classic expectation-maximization (EM) algorithm [162, 163]. There are, however, some practical issues and application specific requirements pertaining to the performance of self-organizing algorithms in vector quantization and clustering problems that precedential SOM and SOM variants have not overcome them satisfactorily. Here we briefly outline some aspects that we believe worth probing and need further evaluation, although we may not be able to completely solve them in this thesis.

- *Complexity of winner-neuron search:*

The SOM is based on competitive learning. The search for a winner neuron requires a computational complexity of $O(N)$, where N is the number of output neurons. When the size of the output neurons is large, both power dissipation and latency of computation will escalate in any implementation platform of SOM. Existing approaches [91, 90] to speed up winner neuron search tends to devastate the topological feature, but lateral search used to alleviate this problem will increase the computation complexity. In addition, they organize the SOMs layer by layer, making the online learning ability of

SOM difficult.

- *Limitation due to predefined map size:*

In SOM algorithm, the map size is fixed before training. In clustering problems, this makes implicit assumptions on the form of clusters, and these assumptions may not be true, thus the clustering result is sensitive to the number of partitions which has to be predefined. In other words, it is hard to determine the number of clusters properly in applications.

- *Constraint on lateral feedback mechanism:*

Lateral feedback is a mechanism used for modifying the form of excitation applied to the SOM. From communication-theoretic point of view, the neurons develop themselves into specific decoders or detectors of their respective stimuli in the input space. These decoders form a computational network whose outputs collectively can be meaningfully interpreted. Due to the unclarified and intricate lateral excitation, assiduous calibration of essential network parameters is needed to achieve the desirable performance and training time for a given source data. In most SOMs, these neuron decoders/detector are usually organized as a 1D string or 2D grid. However, biological and artificial vision probably uses very complicated hierarchical systems in many stages. How the maps should be connected to encode a particular subset of signals, and whether special nonlinear excitation function is needed are closely related problems and their choice for a particular task is itself a dedicated problem.

- *Favoritism of isotropic clusters and order dependent learning:*

In clustering application, SOM based algorithm also suffers from the problem

of favoring equally-sized compact spheroidal clusters, i.e., isotropic clusters. This is caused by vector quantization (VQ) nature of SOM clustering. Another issue arising from the online application is the sensitivity of SOM performance and decision accuracy to the input order of the source data.

1.2 Research Objectives

This thesis aims at the development of new self-organizing topological algorithms stem from a special class of artificial neural computational model called the Self-organizing Map (SOM). The underlying computation algorithms and structures are demonstrated on, and in part motivated by some image processing tasks which we contemplate that the adaptive process of this learning model will be beneficial. Of the major issues to be addressed and improved upon the existing SOM algorithms and their variants underlined in the previous section, the focus of this thesis has been steering towards achieving the following goals:

- to develop new self-organizing learning method with enhanced robustness against the dynamics of the application environment. The aim is to have an SOM with relatively stable and good performance without resorting to onerous calibration and fine tuning of the training parameters.
- to explore hierarchical self-organizing structure and develop new training algorithm to speed up the training and testing phases of its application. While alleviating the winner neuron search problem, the method to be developed shall not undermine the inherent powerful features of SOM, such as

the online learning and topology-preservation properties.

- to instill new features into the learning algorithm and topological maps, such as the multi-scale and spatial-temporal transformation of input data. The importance of multi-resolution (or multi-scale) property has been recognized in many applications [125], and demonstrated by the startling popularity of wavelets and scale-space techniques. It is envisaged that possession of multi-resolution property facilitates progressive embedded encoding, which to some extent will alleviate the fixed map size problem in clustering.
- to explore the potential of the new self-organizing algorithms and techniques in specific applications. By evaluating the performances of the proposed and existing self-organizing algorithms, and other approaches to vector quantization and clustering related problems, the study will provide an insight into the challenges and an impartial appraisal of the self-organizing algorithms to complex digital image processing tasks [95].

1.3 Research Originality

The major contributions of this thesis can be summarized as follows:

- A novel competitive learning scheme is proposed to stabilize the performance of the well-known Self-organizing Maps (SOM) against variation of network parameters when it is employed for vector quantization. The proposed FS-SOM harmonically blends the neighborhood adaptation of the SOM with the

neuron dependent frequency sensitive learning model, the global butterfly permutation sequence for input randomization, and the reinitialization of dead neurons to harness effective utilization of neurons. The net effect is an improvement in adaptation, a well-ordered codebook and fast convergence. More importantly, such holistic approach tones down the reliance of SOM on the trial-and-error calibration of initial state and network parameters for convergence efficiency and quality reconstruction.

- A new on-line tree-structured self-organizing algorithm is proposed. Owing to the layered structure and search method, we call this resulting self-organizing network self-organizing topological tree (SOTT). The computational complexity of the proposed SOTT is $O(\log N)$ as opposed to $O(N)$ for the basic SOM. The experiment results show that the reconstruction performance of the vector quantizer constructed with SOTT is comparable to that of the full-search SOM and its computation time is much shorter than those of the full-search SOM and other vector quantizers. In addition, multi-resolution mapping is made possible by the layered hierarchy of SOTT. The SOTT uses a novel winning path selection and updating method to maintain the intra-layer and inter-layer similarity relations such that the lattice topology at each layer of the network reflects the closeness of dominant features in the input signal space. Unlike the existing tree-structured variants of SOM, fast searching and testing is made possible without sacrificing online learning or compromising the global topological order.
- To avoid favoring isotropic cluster, we also proposed a novel hybrid clustering technique based on SOTT. It was demonstrated to be computational efficient and possesses good scalability. To fully exploit the advantages of online

learning, and reduced computational complexity of the SOTT architecture, a new linkage metric is introduced for the hybrid clustering algorithm. This linkage metric can be updated online during the partition clustering stage and does not need to wait for the patterns to be labeled. Thus, the powerful discrimination of agglomerative hierarchical clustering (AHC) can be performed rapidly at any preset interval of the SOTT. Furthermore, due to the on-line learning nature, the memory requirement of the proposed algorithm depends on the structure of SOTT rather than the size of the data set. These properties are very appealing for large database clustering

- To prevent order dependent learning, a global butterfly jumping sequence and progressive spatial resolution property is instilled into the FS-SOM and SOTT. This analytical sequence can be applied effectively on any dimension of spatially correlated source data (like features of image) to provide an unbiased input order to maximize efficiency for online learning algorithms. This new butterfly jumping sequence is also instilled into the error diffusion technique for color halftoning. Better perceptual quality of the rendered image under this simple and efficient color halftoning technique than that with the conventional raster sequence reveals the intriguing effect of traversal sequence on error diffusion.
- The SOTT performance in classification problem is also studied in skin color detection application. In order to cope with the complex background, robust skin detection is desirable for face segmentation in human-computer interface. Two supervised learning schemes is proposed to use in conjunction with SOTT for pixel-based skin color classification. The receiver-operating characteristic (ROC) curves of the proposed SOTT-based classifiers are ana-

lyzed against traditional skin color classifiers including parametric and non-parametric methods such as fixed range methods [23, 46], Gaussian models [103], SOM [19] and Bayesian classifier [119]. The result shows that our proposed SOTT can be used as a preprocessing tool to effectively reduce the number of bins of Bayesian classifier for a desired classification performance. It is promising as an alternative solution to skin color detection with less amount of training data and memory requirement.

1.4 Organization of the Thesis

This thesis is organized into six chapters. This opening chapter states the motivation of the research topic and its significance, the objective of the research, the original contributions, and the thesis outline. The remaining of the thesis is organized as follows:

Chapter 2 provides an overview of the applications of interest, namely vector quantization, color reduction problem, and clustering. The literature review focuses on accentuating the challenges of these applications, and the merits and demerits of various classical and advanced techniques to these problems. The basic SOM and its principles of operations are discussed. Variants of SOM to overcome the deficiency of the basic SOM are also surveyed. The background information covered in this chapter paves the way for subsequent chapters where these application problems are evaluated on the proposed self-organizing algorithms.

Chapter 3 introduces the novel global butterfly jumping sequence (GBJS) for

randomization of multi-dimensional input data, and a novel competitive learning scheme with stable performance over key parameter variation. The analytical model of global butterfly jumping sequence for permuting multi-dimensional source data with spatial correlation is presented. The underlying principle and evolution of the proposed Frequency-sensitive Self-organizing Maps (FS-SOM) are elaborated. The simulation results of FS-SOM on color quantization of images and the comparison of performance against other related methods are provided. The effect of various measures incorporated into FS-SOM to desensitize order dependent learning is analyzed and discussed.

Chapter 4 presents a novel on-line learning algorithm on a different self-organizing structure called the Self-organizing Topological Tree (SOTT). The SOTT uses a multi-path search and unique winning path updating method to overcome the bias of the tree-structured search and to preserve the topological relationship during learning. The structure of SOTT and the basic terminologies used are introduced before the proposed learning algorithm of SOTT is described. A hybrid clustering method that unifies partition clustering and agglomerative hierarchical clustering (AHC) algorithms on SOTT platform is presented. A new linkage metric for the proposed hybrid clustering on SOTT is defined. Vector quantization of images is used to test and illustrate the learning ability of the proposed SOTT. The results are compared with the basic SOM algorithm and classic LBG algorithm. Clustering performance of the SOTT-based hybrid clustering method is also evaluated.

In Chapter 5, two applications are explored for the proposed self-organizing algorithms. In the first part, high quality color rendering with limited color palettes such as those for low-cost displays and printers is explored. A new color halftoning technique constructed around the GBSJ presented in Chapter 3 is used as a

post-processing step after the color quantization of the proposed self-organizing algorithms. In the second part, the use of SOTT in pixel-based skin color classification problem is studied. As classification necessitates learnt vector quantization, two training and mapping schemes is devised to operate on SOTT with simple supervised learning principles. The evaluation of their efficiency and effectiveness as pixel based skin color classifier is benchmarked against several different well-known skin color classification methods.

Chapter 6 summarizes the work presented in this thesis and concludes by suggesting how the work presented in this thesis can be extended and the relevant problems worthy of future research.

Chapter 2

Background and Literature Review

This chapter presents several image processing related problems and applications to appreciate the motivation, viability and challenges of the new self-organizing algorithms for topological maps to be presented later in this thesis. To set the scene of this thesis, it is appropriate to survey and scrutinize some of these application areas where self-organizing topological maps, among other approaches, are potentially beneficial. We will review antecedent methods to vector quantization, color quantization and halftoning, and clustering, with emphasis on issues and solutions pertaining to the methodologies based on self-organizing algorithms. Towards the end of this chapter, an overview of classical self-organizing maps (SOM) algorithm will be provided, along with a discussion of its shortcomings and a survey of some variants of SOMs that have already been proposed to circumvent specific deficiencies of the basic SOM.

2.1 Vector Quantization

Vector quantization (VQ) is a classical signal-approximation method. It can be considered as a multi-dimensional generalization of scalar quantization (for a detailed review of this topic, see [58]). It realizes a quantized approximation to the distribution of the input data vectors $\mathbf{x} \in \mathbb{R}^n$ using a finite number of so-called codebook vectors $\mathbf{w}_i \in \mathbb{R}^n$ for $i = 1, 2, \dots, k$.

There are two main steps in a vector quantization task. The first step is the vector quantizer design. This process is often referred to as codebook design procedure or training phase. The second step involves the use of quantizer to perform the quantization of given data. This process is also known as mapping procedure or testing phase. Usually, once the codebook is generated, an input $\mathbf{x}$ is approximated by finding a nearest codebook vector $\mathbf{w}_c$ to $\mathbf{x}$ in the input space. In this search, the Euclidean metric is normally used as a measure of the distance between two vectors.

$$c(x) = \arg \min_i \|\mathbf{x} - \mathbf{w}_i\| \quad (2.1)$$

One way of evaluating how optimally a codebook minimizes the expected value of the square of the quantization errors is called the distortion measure, which is defined as

$$E = \int \|\mathbf{x} - \mathbf{w}_c\|^2 p(\mathbf{x}) d\mathbf{x} \quad (2.2)$$

where the integral is taken over the complete input $\mathbf{x}$ space, $d\mathbf{x}$ is a shorthand notation for the n -dimensional volume differential of the integration space, and $p(\mathbf{x})$ is the probability density function of $\mathbf{x}$.

Voronoi tessellation is a concept usually used to visually illustrate the process of

Figure 2.1: Voronoi constellation in 2D space

vector quantization. Figure 2.1 shows a 2D space where the codebook vectors are shown as points positioned according to their coordinates in the input space. This space is partitioned into a finite number of regions, demarcated by lines (in high-dimensional, hyper-planes) such that each partition contains a codebook vector that is the “nearest neighbor” to any vector falling within the same partition. These lines, or hyperplanes of the neighboring codebook vector, together with the codebook vectors, constitute the Voronoi constellation.

The obvious encoding procedure of a vector quantizer is the brute-force nearest neighbor encoding. It compares the source vector $\mathbf{x}$ to be quantized with all reproduction vectors in the codebook C . Since a fixed rate VQ with dimension k and rate R has 2^{kR} codevectors, the number of computations required to do this grows exponentially with the dimension-rate product kR , and quickly gets out of hand. For example, if $k = 10$ and $R = 2$, there are roughly one million codevectors. Moreover, these codevectors need to be stored, which also consumes resources. It is the high complexity of such brute-force techniques that motivates the reduced complexity techniques to be discussed later in this section.

Many algorithms have been proposed for vector quantizer design, including Generalized Lloyd's Algorithm (GLA) [100], simulated annealing [45], deterministic annealing [124], pairwise nearest neighborhood [38], stochastic relaxation [164], self-organizing maps (SOM) [88] and other artificial neural networks [140]. Among them, GLA, also named after the inventors as Linde-Buzo-Gray (LBG) [100], is the most frequently used algorithm. It is an intuitively simple iterative algorithm. However, it is sensitive to the initial conditions and can easily be trapped in local minima. Moreover, it is a batch processing algorithm, which means that all data need to be resided in the memory in order to carry out the vector quantizer design task. This is in contrast to the online learning algorithm of SOM, which can start learning before all data is presented. Due to the topology preserving property of SOM, the codebook generated will be ordered such that the topological relation of codevectors in the voronoi tessellation is maintained. Thus, SOM has promising potential in some applications.

To evaluate the performance of a given quantizer, one might think of a quantizer as being characterized by a quadruple (R, D, C, M) , i.e., the rate R , the distortion D ,

the computation complexity C and the memory complexity M [32, 58, 56, 62, 128]. Besides the consideration of the quadruple (R, D, A, M) , in some applications it is desirable that the output of the encoder be progressively decodable in the sense that a coarse reproduction can be made from the first few bits that it receives, and improved reproductions are made as more bits are received. Such quantizers are said to be progressive or embedded. Now it is true that a progressive decoder can be designed for any encoder (for example, it can compute the expected value of the source vector given whatever bits it has received so far). However, a “good” progressive code is one for which rate-distortion performances are considerably good over a span of quantizer resolutions despite the intermediate distortion achievable at each rate may not be as good as that produced by a dedicated quantizer designed for only that particular rate. Instead of restarting from scratch each time the decoder receives a new bit (or group of bits), a progressive code uses some simple method to update the current reproduction. This is highly desirable in contemporary multimedia applications that the encoding process is progressive for bandwidth adaptation rate control.

2.1.1 *Tree-Structured Vector Quantization*

As mentioned earlier, a brute-force implementation of vector quantizer can be very computation intensive, especially in the case of large codebook C . Therefore, fast search algorithms were developed for unstructured reproduction codebooks, and even faster searches were proposed for reproduction codebooks which are constrained to have a dedicated structure. For example, the codebook of a lattice vector quantizer is a subset of points of a regular lattice. Additional structure can

be imposed for faster searches with virtually no loss of performance, as in Fisher's pyramid VQ [40]. The pyramid VQ takes advantage of the asymptotic equipartition property to choose a structured support region for the quantizer. Generally, the methods to speedup the search time of VQ can be broadly categorized into two groups:

- one that accelerate the search based on the unstructured codebook [58] (Section V-A).
- the other is based on structured quantizer [58] (Section V-B).

Among the many structured quantizer, Tree Search Vector Quantizer (TSVQ) [20, 21, 57] is one of the best known approaches to speed up vector quantization. In its original and simplest form, a k -dimensional tree-structured vector quantizer (TSVQ) [20, 21, 57] is a fixed-rate quantizer with, say, rate R whose encoding is guided by a balanced (fixed-depth) binary tree of depth kR . There is a codevector associated with each of its 2^{kR} terminal nodes (leaves), and a k -dimensional testvector associated with each of its $2^{kR} - 1$ internal nodes. Quantization of a source vector proceeds in a tree-structured search by finding which of the two nodes stemming from the root node has the closer testvector to $\mathbf{x}$, then finding which of the two nodes stemming from this node has the closer testvector, and so on, until a terminal node and its codevector are found. The binary encoding of this codevector consists of the sequence of kR binary decisions that lead to it. Decoding is done by table lookup as in unstructured VQ. Similar to successive approximation scalar quantization, TSVQ naturally yields a progressive structure of embedded code.

It has been reported that when taking both performance and complexity into account, TSVQ is a very competitive VQ method [58, 57]. Indeed, many of the fast-search approaches use a tree-structured pre-quantization. However, in TSVQ the searching tree and the codebook have to be matched in size and character in a specific way that makes them work well together.

The usual (but not necessarily optimal) greedy method for designing a balanced TSVQ [20, 21] is to first design the test vectors stemming from the root node using the Lloyd algorithm on a training set. Then design the two test vectors stemming from, say, the left child by running the Lloyd algorithm on the training vectors that were mapped to the left child, and so on. In general, fixed rate TSVQ performs worse than the full-search VQ in terms of quantization distortion. Improved performances have been observed in variable rate TSVQ and dynamically expanding TSVQ [58]. Tree-structured self-organizing maps were introduced by Koi and Oja [91] and Koi [90] to boost the search speed by using tree-structure search instead of traditional sequential search. Unlike in TSVQ where the search in lower level is deployed among the children nodes of the “winner” in the upper level, searching in these tree-structured self-organizing maps are performed in all children nodes of the upper level “winner” and its neighbors. This is designed to compensate the false ordering induced in the upper level. Nevertheless, the formation of layers in the tree-structured self-organizing maps [91, 90] is similar to that of TSVQ. That is, they are also organizing from root level to leaf level one by one.

The above discussion fuels the work of Chapter 4, where we propose a new tree-structure SOM, called the Self-organizing Topological Tree (SOTT). The SOTT can be used to design efficient tree-structured vector quantizer with strong topo-

logical ordering on-line. Unlike TSVQ and previous tree-structured self-organizing maps, the layers of SOTT are formed concurrently.

2.2 Color Quantization and Halftoning

True color images typically contain up to 16 million different colors. One of the basic tasks of low-level image processing is to reduce the number of colors with minimal visual distortion, since many image display and printing devices provide only a limited number of colors. The representation problem for colors is aggravated when many images are displayed simultaneously, resulting in a palette size of 256 or even smaller number of colors being assigned to images. From computation efficiency viewpoint, a coarse resolution of colors is important for the efficiency of image processing and analysis in the later stage.

The color reduction process, which aims to minimize an objective distortion measurement (e.g. *means square error*) of the reconstructed image, is called color quantization. Examples are the popular median-cut quantizer [65], octree quantization [52], sequential scalar quantization [11, 12] and a variety of clustering methods like LBG [18], agglomerative clustering [156], and self-organizing networks [34]. Other computational-effective clustering approaches could also be used, see [61, 17, 74] for an overview. Color quantization alone gives good result for color reduction oriented towards computational efficiency, but is somehow inadequate for display and printing applications where the reproduction quality is judged by the human eyes. This is due to the fact that statistical distortion measurements do not account for the unique acuity of human vision system.

Digital halftoning methods [145] provides a means after color quantization to address the above problem by exploiting the property of human visual system. Human eyes perceive high frequency spatial variations in color as a homogeneous color. This can be interpreted as an averaging superposition or low-pass characteristic of the contrast sensitivity of human visual system. Therefore, additional perceived colors can be created by spatial mixing. In a common halftoning technique called error diffusion, the quantization error is spread to neighborhood pixels, i.e., the distortion at neighborhood pixels is biased in opposite directions. Several error diffusion filters have been proposed [41, 144, 93]. In addition, many dithering [111, 139] and model-based halftoning [139, 92, 113] techniques have also been developed.

2.2.1 Color Quantization Techniques

The problem of color quantization is a specific instance of the more general problem of vector quantization (VQ) [51]. As the true color image usually have much more color than the number of entries allowed in color palette. The palette selection and mapping of color to palette entry could be viewed as the process of dividing the colors of an image or a group of images into a finite number of clusters in 3-D color space and selecting a representative color for each cluster. The problem is often formulated in terms of minimization of the MSE between the true-color in the original images and their representatives selected from the color palette. The selection of a globally optimal palette under the MSE criterion is a nondeterministic polynomial-time (NP) complete problem [70, 47]. For cases where the number of palettes and true colors is relatively small, a branch and

bound algorithm can be used to determine the optimal solution [94]. However, for most real-world images, this approach is infeasible. Therefore, normally in real applications, suboptimal but efficient palette selection algorithms are used to achieve acceptable performance in reasonable time. For example, the Linde-Buzo-Gray (LBG) algorithm [100], identical to the k -means clustering algorithm [101] used in pattern recognition and classification problem, can be used. The algorithm iteratively improves an existing palette and assigns image colors to the closest palette entry to reach a local minimum with respect to the MSE criterion.

In the palettization schemes based on MSE, no guarantee to the maximum color error was offered. This problem is addressed in the octree quantization algorithm [52] and the center-cut algorithm [77]. In the center-cut algorithm, the cluster with the largest dimension along a coordinate axis is split along the center at each step. In contrast to the top-down schemes, the octree algorithm is a bottom-up approach to the problem. Conceptually, it subdivides the color space into cubes until each cube contains only one image color and then reduces the resulting octree by an averaging and merging process so that each node represents a palette color. This process is similar to the agglomerative hierarchical clustering in Section 2.3. In practice, the expansion and reduction are performed in a single pass. An alternate bottom-up color quantization scheme, that uses preprocessing techniques like histogramming and prequantization with an agglomerative hierarchical clustering algorithm, was also proposed in [10].

Once the design of a palette is over, the second step in color quantization, i.e., the mapping of image pixels to the palette colors, needs to be performed. The simplest approach to this problem is to map each pixel to its nearest neighbor in the palette. Often, the palette design process can be used to obtain a tree structure

that simplifies this nearest neighbor search or provides a good approximation to it. This is particularly true for the binary splitting algorithms that use the classic k -d tree [42] for nearest-neighbor search. Similar procedures can also be readily used with the octree quantization scheme. Note however, that if the LBG algorithm is used for iterative improvement of the obtained palette, the tree structure of the palette is destroyed. Consequently, the nearest-neighbor search is more involved though fast searching schemes can still be developed [110, 68].

While the nearest-neighbor mapping algorithm is optimal from a minimum-average-error standpoint, it often produces objectionable contours in smooth image regions. A significant reduction in visible contours can be obtained by using the halftoning techniques of dithering or error diffusion, which will be briefly discussed later in this section. The use of these techniques in the pixel mapping step has been discussed by a number of researchers [65, 18, 111, 49, 50]. For more information on color quantization, please refer to an excellent review given by [130].

2.2.2 Color Halftoning Techniques

The human eye is extremely sensitive to color variations, and is capable of distinguishing around 10 million colors under optimal viewing conditions [78]. At the same time, color output devices such as halftone color printers and palette based displays are capable of producing only a limited number of colors at each addressable spatial location. However, these devices normally possess rather high spatial resolution, which is often beyond the resolving capabilities of the human eye at normal viewing distances. In reproducing color images on these devices, it

is therefore desirable to use techniques that trade off spatial resolution in favor of a greater range of perceived colors. The term *halftoning* is used to describe a variety of image processing techniques based on this idea.

The eye perceives only a spatial average of the microvariation in spot-color produced by the device, and is relatively insensitive to high-frequency differences between the original continuous-tone image and the halftone image. Halftoning algorithms therefore attempt to preserve this average in the reproduction while forcing a large fraction of the difference between the halftone and continuous tone images into the perceptually irrational high-frequency regions.

In conventional digital color halftoning (for printers), the image is decomposed into a cyan, magenta, yellow, and black separations, which are halftoned independently. The halftoning for each separation is done by comparing the pixel value with a spatially repeated dither array and turning on pixels for which the image exceeds the value in the corresponding dither matrix [123]. The dither matrix is designed so that the thresholds that are close in values occur near each other in the dither array.

Halftoning schemes that employ dither matrices quantize the image pixels in isolation. Considerable improvements in image quality can be obtained by using adaptive schemes that process each pixel according to the result of processing of other pixels. Error-diffusion (ED) [127, 41] is an adaptive scheme that has been widely used. ED works by “diffusing” the error resulting from the quantization of the current pixel to the neighborhood pixels. At each pixel, the diffused error is added to the image value prior to quantization, and the quantization error is again distributed over adjacent pixels. Since the objective of error diffusion is to

preserve the average value of the image over local regions, a unity-gain lowpass finite impulse response (FIR) filter is used for distributing the error. From a signal processing viewpoint, ED can be viewed as a 2D version of $\Sigma - \Delta$ analog-to-digital (A/D) converters [6, 85, 92, 83].

The ED filter needs to be causal if the image is to be halftoned in a single pass. The causality requirement for normal raster processing implies that the filter is asymmetric. This asymmetry often results in visible low-frequency “wormlike” artifacts in bilevel ED. Several schemes, such as a larger area of support for the FIR filter [75, 132], processing on a serpentine raster [84], and symmetrical error diffusion neural networks [6], have been proposed for gray-scale images to overcome these limitations.

The halftoning methods mentioned thus far are heuristic algorithms that exploit the lowpass nature of the spatial response of human eyes. The problem of halftoning can alternately be formulated as an optimization problem that aims at maximizing the “visual similarity” between the original image and the halftoned image. In order to quantify visual similarity, several schemes utilize analytical models for the spatial response of human eyes. These schemes are therefore referred to as *model-based halftoning methods* [139, 92, 105, 113]. The exact optimization in model-based halftoning schemes is an intractable integer programming problem. For display of gray-scale images on binary output devices, a number of researchers have suggested iterative schemes that provide good solutions with varying computational requirements [4, 114]. For color displays with reasonable sized palettes, the problem remains computationally infeasible at present.

Since the computational requirements of most model-based schemes are rather

restrictive, researchers have turned into investigating hybrid schemes that use the halftoning methods mentioned earlier but with an attempt to minimizing a visual model-based error. For gray-scale images, Sullivan et al. [138] have incorporated a visual model in ED. The use of neural networks to minimize a visual model-based distortion function in symmetric error diffusion has also been reported in [6]. For color applications, much less research has been done. The optimization of ED for color display applications has been reported in [115, 93], where optimal ED filter coefficients were determined through a process of autoregressive (AR) modeling of the spatial response of human eyes.

The method of ED continues to be an active area of research. Recent novel developments in this area include the use of adaptive signal processing techniques and embedded quantization schemes. Image-adaptive ED filters that use the well-known least-mean-square (LMS) algorithm [151] have been employed for halftoning gray-scale [153] and color [1, 2] images (albeit using slightly different error criteria). The methods exploit the local characteristics of the image to obtain improvements over a constant filter. Another modification of color vector ED is proposed in [54, 55]. The resulting embedded multilevel ED algorithm allows coarse quantizations of an image to be embedded in finer quantizations, which can be useful in (among other things) rate controlled progressive transmission of color images.

Figure 2.2: A taxonomy of clustering approaches [74]

2.3 Clustering

Clustering is the unsupervised (without human supervision) classification of patterns (observations, data items, or feature vectors) into groups (clusters). It is an important step and approach of exploratory data analysis and has been studied in many applications by many researchers. However, clustering is a difficult problem, since the different assumptions of different applications have made the transfer of useful generic concepts and methodologies slow to occur.

Different approaches to clustering data can be described with the help of the hierarchy shown in Figure 2.2 (other taxonomic representations of clustering methodology are possible; ours is based on the discussion in [73, 74]). At the top level, there is a division between hierarchical and partitional approaches. The partitional result of hierarchical methods is a nested series of partitions, while partitional methods produce only one level of partition result.

In addition to the taxonomy shown in Figure 2.2, there are some other features to compare different clustering algorithms.

- *Agglomerative vs. divisive*: This feature is from the aspect of algorithmic structure and operation. An agglomerative approach begins with each single pattern as a distinct cluster (singleton), and merges closest clusters together at each step until a stopping criterion is satisfied. A divisive method begins with all patterns in a single cluster and performs splitting until a stopping criterion is met.
- *Hard vs. fuzzy*: A hard clustering algorithm will assign each pattern into a single cluster. A fuzzy clustering method might assign a patterns to different clusters (with degree of membership). A fuzzy clustering contains more information than hard clustering algorithm, and can be converted to a hard clustering by assigning each pattern to the cluster with the largest measure of membership.
- *Incremental vs. non-incremental*: The incremental clustering algorithm could perform clustering based on the previous clustering result. This is useful in applications where the pattern set are large or incrementally built, or the constraints on execution time or memory space affect the architecture of the algorithm.

2.3.1 Hierarchical Clustering Algorithms

The hierarchical clustering algorithms normally follows this framework:

1. Each pattern is considered as a single cluster. The proximity matrix containing the distances between each clusters are calculated.
2. The most similar pair of clusters is found using the proximity matrix. These two clusters are merged into a new larger cluster. If all patterns are in one cluster, the algorithm is terminated. Otherwise, the proximity matrix is updated to reflect this merge operation and then this step is repeated.
3. The output of the algorithm is a nested hierarchy of graphs which can be cut at a desired proximity level. The completely connected components in the corresponding graph forms the result clustering.

Most hierarchical clustering algorithms are variants of the single-link [134], complete-link [81], and minimum-variance [150, 106] algorithms. Different metric are used to measure the similarity between a pair of clusters in these algorithms. Among them, single-link and complete-link method are mostly used in application. In the single-link method, the distance between two clusters is the minimum of the distances between all pairs of patterns with one from each of the two clusters. In the complete-link algorithm, the distance between two clusters is the maximum of all pairwise distances between patterns in the two clusters. In either case, two clusters with the minimal distance are merged to form a new, larger cluster. It is reported in [74] that the complete-link algorithm produces tightly bound or compact clusters [9], while the single-link algorithm, in contrast, suffers from a chaining effect [?] and has a tendency to produce clusters that are straggly or elongated.

Hierarchical algorithms are more versatile than partitional algorithms. For example, the single-link clustering algorithm works well on data sets containing non-isotropic clusters including well-separated, chain-like, and concentric clusters, whereas a typical partitional algorithm such as the k -means algorithm works well only on data sets having isotropic clusters [108]. On the other hand, the time and space complexities [33] of the partitional algorithms are typically lower than those of the hierarchical algorithms. It is possible to develop hybrid algorithms [107] that exploit the good features of both categories. These exposition of agglomerative hierarchical clustering algorithms will help to understand the inspiration of the new SOTT based hybrid clustering method to be proposed in Chapter 4.

2.3.2 *Partitional Algorithms*

Unlike the hierarchical clustering algorithm, a partitional clustering algorithm obtains a single partition of the patterns. Partitional methods has less computational complexity than hierarchical algorithms and thus have advantages in applications involving large data sets.

Square error criterion is normally used in partitional clustering techniques, and a classic example is k -means clustering algorithm. It starts with an initial randomly-assigned partition and keeps reassigning the patterns to the clusters with the lowest square error to the pattern until a convergent criterion (e.g., there is no reassignment of any pattern from one cluster to another, or the squared error ceases to decrease significantly after some number of iterations) is met.

The k -means algorithm is popular because it is easy to implement, and its computational complexity is linear to the number of patterns. A major problem with this algorithm is that it is sensitive to the selection of the initial partition and may converge to a local minimum of the criterion function value if the initial partition is not properly chosen.

k-means Clustering Algorithm

1. k cluster centers are initialized in the data space.
2. Each pattern in the data set is assigned to the closest cluster center, according to the distance metric (normally square error)s.
3. Each cluster center is then updated as the mean of all data patterns which has been assigned to the cluster.
4. The steps from Step 2 are iterated until a convergent criterion is satisfied. Typical convergent criteria are: no (or minimal) reassignment of patterns to new cluster centers, or the reduction in squared error saturates around some infinitesimal value.

Several variants [7] of the k -means algorithm have been reported in the literature. Some of them attempt to select a good initialized cluster centers so that the algorithms are more likely to converge to the global minimum value. Another technique to enhance partitional clustering algorithms is to use splitting and merging of clusters. For example in ISODATA [13], a cluster is splitted when its

variance is above a pre-specified split-threshold, and two clusters are merged when the distance between their centers is below another pre-specified merge-threshold.

2.4 Self-organizing Maps

2.4.1 Basic SOM Algorithm

The Self-organizing Maps (SOM) fall into the class of unsupervised competitive learning neural networks. Its learning algorithm essentially implements a characteristic nonlinear projection from a higher-dimensional space onto a much lower-dimensional network in an orderly fashion. It consists of a finite set of models that approximate the open set of input data, and the models are associated with nodes (neurons) that are arranged as a regular, usually 2D grid. The models are produced by a learning process that automatically orders them on the 2D grid along with their mutual similarity. SOM has been used for a wide variety of applications including computer vision [129], image analysis, speech analysis and recognition [72, 71, 121], telecommunications, process control, robotics, data processing and analysis. A comprehensive review of this topic is treated by Kohonen [88, 89].

A typical Self-organizing Map consists of two layers, an input layer and an output layer. The input layer operates as a buffer for the input vectors $\mathbf{x} = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n$, whereby the inputs are transmitted parallelly to all the neurons in the output layer. In the output layer, which is also called the competitive layer, the neu-

Figure 2.3: Structure of a two dimensional SOM network

rons are interconnected to each other to form a linear, planar or multidimensional structured topology. Every neuron i is associated with a parametric reference vector $\mathbf{w}_i = [w_{i1}, w_{i2}, \dots, w_{in}]^T \in \mathbb{R}^n, i = 0, 1, \dots, N - 1$. Figure 2.3 shows a typical SOM structure in which the input data space $\mathbb{R}^n$ is mapped onto a two-dimensional array of neurons.

As indicated in Figure 2.3, all neurons are presented with the same input vector $\mathbf{x}$ in parallel, and all neurons compute its distances between their weights $\mathbf{w}$ and the input vector $\mathbf{x}$ in parallel. Only the neuron with the closest match between the input vector and its weight produces an active output. Each neuron therefore acts as a separate decoder (or pattern detector, feature detector) for the same input and the interpretation of the input is derived from the presence or absence of an active response at each neuron. The lateral interaction between neurons

introduces a sense of topology, such that neighboring units represent inputs that are close together in the input space. Therefore, SOM is also referred to as Self-organizing Topological Map [88].

The algorithm flow of SOM can be summarized as followed:

1. the initial network setup

At $t = 0$, the weights of the neurons, $\mathbf{w}_i(0) = [w_{i1}, w_{i2}, \dots, w_{in}]^T \in \mathbb{R}^n, i = 0, 1, \dots, N - 1$, are initialized according to a selected initialization criterion. The network topology is defined and the size of the neighborhood, $h(0)$ is initialized. The initial learning rates, $\alpha(0)$ for the winning neuron and $\beta(0)$ for the neighbors of the winning neuron are also defined.

2. After the initialization, the input vectors are fed into every neuron of the network. Then the neuron that has a closest matched weight to the input vector is selected as the winning neuron. The winning neuron is also called the *Best Matching Unit* (BMU). Usually, the winning neuron is selected by

$$c(x) = \arg \min_i \|\mathbf{x} - \mathbf{w}_i\| \quad (2.3)$$

where $\mathbf{w}_i$ is the weight of the neuron i . $c(x)$ is the label of the winning neuron which varies with the input vector.

3. The next step, which is also the core operation of SOM, is to update the weights of the winning neuron and its neighbors as follows:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + h_{c(x),i}(t)[\mathbf{x}(t) - \mathbf{w}_i(t)] \quad (2.4)$$

where t is the index of the iteration step and $\mathbf{x}(t)$ is the input vector which varies with t . The scalar multiplier $h_{c(x),i}$ is called the neighborhood function.

Figure 2.4: Major steps of SOM training algorithm

It can be viewed as a smoothing kernel over the topology of the SOM which is defined by the algorithm. Usually, a 1D string or 2D grid topological structure is employed for the SOM network. A commonly used neighborhood function is a Gaussian kernel:

$$h_{c(x),i}(t) = h_0 \exp(-\|\mathbf{r}_i - \mathbf{r}_c\|^2 / \sigma^2) \quad (2.5)$$

where $\mathbf{r}_i$ and $\mathbf{r}_c$ are the coordinates of the neurons i and c , respectively. $h_0 = h_0(t)$ and $\sigma = \sigma(t)$ are some monotonic decreasing functions of time.

4. The input vectors from the training set will be fed into the network iteratively until the network is converged. The convergent criterion is determined by the changes in the neuron weights being asymptotically small.

In short, the basic SOM algorithm consists of drawing sample vectors from the

input pixels and “teaching” them to the neurons in the output layer. The teaching is done by choosing a best matching unit by means of a similarity measure and by updating the weights of the neurons in the neighborhood of the best matching unit. This process is repeated a number of times until convergence. In this manner, the net tends to approximate the probability density function of the input data. The learning rates, $\alpha(t)$ and $\beta(t)$ are used to control the amount of bias added to the winning neuron and its neighborhood. To emulate a fast learning, slow forgetting training process, the learning rates are set high initially. As more input pixels are presented to the network, both $\alpha(t)$ and $\beta(t)$ are gradually reduced to zero according to a linear or an exponential function. With large initial learning rate, the network is able to adapt faster to the clusters of input data at the beginning of the training stage. The neuron weights are fine tuned at the later stage with decreasing $\alpha(t)$ to guarantee convergence.

2.4.2 Application and Variants of SOMs

One distinctive feature of SOM is its ability to form localized clusters (activity bubbles) around the maximally responsive neuron (winner) through competitive learning algorithm with only a simple layer of feed-forward synaptic connections and a competitive winner-take-all (WTA) neuronal layer. The location of each activity bubble is determined by the characteristics of the input data. A competitive neural network, in general has the lateral interactions in the WTA layer to produce the excitatory and inhibitory actions based on the distance from the most active output neuron. SOM has in addition a time-variable topological neighborhood of active neurons around the activity bubble. The winner and its neighborhood

neurons that fall under the influences of the activity bubble have their synaptic weights adjusted towards the input stimulus. This gives the SOM the topological ordering capability whereby the spatial location of a neuron in the lattice corresponds to a particular feature or a group of features of the source. It is this simple network topology and the unsupervised self-annealing property of SOM that underline its strength and interest over classical statistic approaches in emerging new applications that call for clustering of unlabeled multivariate data [39, 37, 53, 88].

A dichotomy in the applications of SOM is that of quantization [58] in which the aim of data clustering is to obtain an optimal set of codebook vectors that maximizes the rate-distortion performance, versus clustering which aims to segregate a chaotic mixture of patterns for the purpose of knowledge discovery and analysis. Unsupervised learning allows the SOM to group datasets on the basis of their perceived closeness in a multivariate hyperspace. The synaptic weights of the trained network provide an approximation to the Voronoi vectors in a self-organizing manner. As the primary function of SOM is to visualize the topological relationship of input vectors, for pattern classification, the trained network needs to be calibrated by a supervised optimization technique called Learning Vector Quantization (LVQ) [88] to minimize the average probability of misclassification.

However, the classic SOM algorithm suffers from several disadvantages in its application to clustering problems: first, The clustering result is sensitive to the number of partitions which has to be predefined; Second, the clustering result of one partition provides knowledge at only one similarity level; Third, SOM based algorithm also suffers from the problem of favoring “equally-sized compact spheroidal clusters”, i.e., isotropic clusters. This is caused by the mean square error (MSE) function and vector quantization (VQ) nature of SOM clustering. Actually, these

limitations are also common to the classic k -means and many other partitional clustering algorithms discussed earlier in Section 2.3. Many variants of SOM have been proposed to mitigate its weaknesses in clustering ability. Growing SOM [3], Growing Grid [43] and incremental grid growing [16] are a class of algorithms that overcomes the first problem by growing the SOM to a suitable number of partitions through the insertion of new neurons. Tree-structured SOM [91, 90] provides a hierarchical structure to reduce the computation complexity and alleviates the first two problems simultaneously. Recently, a Grow Hierarchical SOM (GHSOM) [122] is also proposed to grow a hierarchical SOM. To solve the third problem, some researchers generalize the model of SOM, e.g., SOMN [163], to a network of more generalized distribution models (e.g., Gaussian, Delta) at the expense of increased computational complexity. However, in real applications, the data distribution may not satisfy the distribution assumed a priori. Although many variants of SOM have been proposed, the three problems of SOM mentioned are far from being well solved, the last two problems, in particular.

In vector quantization, yet another dichotomy is based on whether the search for the best matching codevector is constrained or unconstrained. An unconstrained search algorithm, such as the classic Generalized Lloyd Algorithm (GLA) [100], and the basic SOM, requires a full search over the entire set of codevectors to compute the distortion between an input data point and every codevector to locate a best matching codevector. As the computational complexity of the full search VQ is proportional to the number of codevectors, it becomes computationally impractical as the feature space or the number of clusters grows beyond a certain limit. It has been reported in [91, 90] that the basic N -neuron SOM algorithm using full search algorithm to find the best matching neurons takes about 10^4 to 10^5 iterations with floating point operations to organize a M -dimensional input dataset.

To reduce the encoder complexity, a constrained search procedure structures codevectors in such a way that only a small subset of the entire codebook is searched for the best representative codevector with minimal distortion. Among the many constrained search algorithms [58], Tree Search Vector Quantizer (TSVQ) [21] is one of the best known approaches to speed up vector quantizers. In general, fixed rate TSVQ performs worse than the full-search VQ as mentioned in Section 2.1. It is attributable to the accumulation of the bias introduced by the constrained search from the parent nodes to the child nodes. Improved performances have been observed in variable rate TSVQ [58], but direct application of the tree search algorithm on VQ lacks the important topology preserving property of SOM. The first Tree-Structured SOM (TS-SOM) was introduced by Koikkalainen and Oja in 1990 [91], and was later improved by Koikkalainen [90]. Their proposed TS-SOMs are organized layers by layers. That is, after one layer has been organized, its nodes are frozen and the organization proceeds on the next layer. A limitation of this type of TS-SOM is that it is not readily implementable as an on-line learning scheme. All training data are fed into the system repeatedly at every layer which taxes the system resources heavily for large database. In order to improve the performance of learning, TS-SOM uses a lateral search to lower the inherent bias present in the tree-structured search. However, this lateral search increases the computation load and annihilates the hierarchical and progressive mapping of codevectors along the binary- or quad-tree. Another Binary Tree-structured SOM (BT-SOM) was proposed by T. D. Chiueh et al., for vector quantization [30]. They compare their vector quantization performance against GLA based TSVQ. However, the 1D binary tree model limits its scalability in processing time, and its degradation in performance due to the local decision making procedure against the basic SOM has not been evaluated.

2.5 Chapter Summary

This chapter provides an overview and survey of techniques used in vector quantization, color reduction, and clustering for digital image and data processing. The operational principle of SOM and variants of SOM are then introduced. The phenomenal successes of SOM in mining of unlabeled multivariate data suggest the potential of using self-organizing algorithms to tackle some issues of the above application areas reviewed earlier in this chapter. Although the basic SOM offers competitive performance and elegant features such as online learning and topology preservation, some drawbacks such as the full-search complexity are acknowledged and attempts are made to overcome them.

The scope of this chapter preambles our exploration on self-organizing algorithms and topological maps. Two novel self-organizing algorithms will be presented in the chapters that follow. In the next chapter, we will present a novel unsupervised learning algorithm that incorporates frequency-sensitive learning into the self-organizing algorithm to make quantization performance relatively insensitive to change of major network parameters. In Chapter 4, another online self-organizing topological tree will also be presented to reduce the search complexity of the classic SOM and deliver better clustering performance. Applicability and performance evaluation of these two novel self-organizing algorithms in practical problems of color reduction and skin-color classification will be further explored in Chapter 5.

An overview of the proposed algorithms, their features and the target applications studied in this thesis can be summarized with Figure 2.5. The taxonomy shows visually in two quadrants the relation of the salient features of the proposed tech-

Figure 2.5: Taxonomy of proposed techniques, attributes and applications studied in this thesis

niques for self-organizing algorithms, and the applications on which the proposed techniques have been leveraged to enhance certain aspects of their performances. Intersection blocks colored in dark gray mark the greater emphasis and significance of the results than those in light gray.

Chapter 3

Frequency-sensitive Self-organizing Map

3.1 Introduction

In this chapter, we will describe a novel frequency-sensitive self-organizing topological map for vector quantization. Its figure-of-merits are illustrated through color quantization of digital still images. A large portion of this chapter has been published in the *IEEE Transactions on Neural Networks* [26] and presented at the *2004 IEEE International Symposium on Circuits and Systems* [25].

When VQ is implemented on neural network architecture, it is commonly accomplished with a competitive learning (CL) scheme. The computing mecha-

nism of competitive learning scheme is inherently parallel, which promises significant speedup and real-time processing capability when it is realized in massively parallel hardware architecture. Another advantage of CL is its on-line updating property. Unlike the batch algorithms such as the Linde, Buzo and Gray (LBG) algorithm [15], CL schemes do not have to wait for all the training data to be stored in memory before processing can begin. A variety of competitive learning schemes [31, 44, 66, 96, 102], typified by their unique learning rules, have been developed. The simplest and most widely used CL algorithms are based on the winner-take-all (WTA) [66] (or hard competitive learning) paradigm, where adaptation is restricted to the winner that is the single neuron prototype best matching the input pattern. Restricting the winner to only a single neuron prototype greatly simplifies the hardware implementation and allows the network to degrade gracefully when a significant portion of its processing elements and interconnections are defective. Unfortunately, the simple CL algorithm is susceptible to the under-utilization problem [64], which manifest as severe false contouring artifacts in color quantization application. Significant effort has to be expended to eliminate the existence of dead neurons or lower their occurrences.

By relaxing the WTA criteria, neural-gas network [102] and fuzzy competitive learning [31] treat more than a single neuron as winners and update their prototypes by different extents accordingly. These algorithms can be classified under the winner-take-most (WTM) (or soft competitive learning) paradigm. Though effective in eliminating the under-utilization problem, WTM algorithms are too complex and expensive to be used for some applications, such as Color Quantization.

Under the WTA category, Frequency Sensitive Competitive Learning (FSCL) [96]

desensitizes the order-dependent learning by incorporating a frequency sensitive component into the competitive learning rules. The idea is to enhance the winning capability of the less competitive neurons, i.e., potential dead neurons so that they too, have equal opportunity to adapt. Based on their simulation results, the performance of FSCL is still slightly worse than that of the LBG algorithm. A convergence study of FSCL was later carried out by [44]. Founded on the concept of FSCL, Chen et al. [28] developed a multi-path Frequency Sensitive Organization (n-path FSO) method. To further alleviate the under-utilization problem, two additional operations are employed, a delete operation to get rid of the less competitive neurons and the split operation to expand the frequent winners. The improvement in performance is achieved at the expense of much higher computational complexity.

Kohonen's Self-organizing Maps (SOM) [88] is a promising vector quantization technique which employs WTM strategy at the early stage of training and gradually become a WTA when its neighborhood size shrinks to zero. In addition to achieving good reconstruction quality with low distortion, it also realizes a topology preserving mapping through the tapering neighborhood updating scheme. However, the adaptation rate and reconstruction performance of many SOM and FSCL algorithms are very sensitive to the setting of the initial state and the network parameters. Quality performance with acceptable training time is often accomplished through scrupulous tuning and customized tailoring of a set of network parameters.

In this chapter, a novel Frequency Sensitive Self-organizing Maps (FS-SOM) is proposed. It inherits the neighborhood adaptation property of SOM so that greater efficiency will accrue to the coding and transmission of data from the topology

preserving codebook. In contrast to SOM, the learning rate of FS-SOM is adaptive to the winning frequency, or more precisely, the “update counter” of each neuron. Our proposed FS-SOM can be conceived as a generalization of FSCL with the notion of neighborhood adaptation. One salient feature of our proposed FS-SOM is its robustness against key network parameter variation. The training algorithm has been designed with a holistic consideration of convergent efficiency and performance stability. Its performance of VQ and robustness feature will be evaluated in color quantization application.

Another problem of online learning schemes is order dependent problem: when online learning is required, the input order can largely influence the result of quantization. In this chapter we also proposed a new Global Butterfly Jumping Sequence to map multi-dimensional data into one-dimensional sequence. Global Butterfly Jumping Sequence has proven to be more suitable than traditional input sequences for many online learning problems.

The rest of the chapter is organized as follows. In Section 2, the proposed Global Butterfly Jumping Sequence for randomization of multi-dimensional input is introduced. Section 3 presents the underlying principles and evolution of the proposed FS-SOM algorithm. In Section 4, analysis and discussion on the simulation results of FS-SOM and the comparison against other related methods are provided. The chapter is closed with a summary in Section 5.

3.2 Global Butterfly Jumping Sequence

3.2.1 Image and Sequences

In computing, data are usually stored and processed in 1D sequences, but in real life, data usually appear in multidimensional forms. The most common example is in digital image processing, where two-dimensional arrays of image data are processed in raster-scanned sequence by scanning the image one row at a time [8, 160, 98]. From information theory perspective, an image can be considered as two-dimensional data with spatial dependency. To arrange image data into a one-dimensional temporal sequence, the spatial dependency should be considered.

Different applications require different sequence properties. While a simple fast scan of image pixel will suffice for some applications, certain traversal sequences are preferred for others. By preserving the spatial dependencies between data as much as possible for processing, it allows the data sequence to be better modeled by powerful mathematical tool such as Markov chain. The spatial dependency is also very important for data compression efficiency. Sequence writing methods with this property include space-filling curves such as Hilbert curve [135]. Following the need to preserve data dependency, these scans aim to visit spatially close pixels in close time as much as possible.

Interestingly, in some circumstances, the opposite requirement is desirable. For example, in order-dependent learning, the learning paradigm tends to be misled by data fed preemptive when the earlier batch of data do not represent the distribution of the whole data sequence well. On the other hand, the output can

also be overly adjusted by data fed later if the later arrival data pattern deviates significantly from the distribution of the overall data. To improve the quality of learning, we would like the data fed under the online learning paradigm to remain relatively stable over time. In other words, it is desirable that every data segment of reasonable length in the sequence has similar distribution and is representative of the overall data distribution. In this case, a good scan is actually one that scatters the spatially close image pixels into far apart positions of the data sequence, so that two spatially far apart pixels are visited in close time proximity.

In short, when two-dimensional data with spatial dependency are to be arranged in a temporal sequence for online learning problem, the preferred input ordering problem reduces to writing a sequence of natural numbers on the plane so that points close on the plane are labeled by numbers which are far apart.

3.2.2 Global Butterfly Jumping Sequence

In [116], the image is partitioned into many 8×8 or 16×16 arrays of discrete data, and each tile of data is processed in a fixed order following a butterfly jumping pattern as illustrated in the Table 3.1. This butterfly jumping sequence is used to achieve a pseudo randomization of the input data. No formal mathematical model of the blockwise butterfly jumping sequence and detailed illustration of its influence to learning quality is given in [116].

In this section, we will first provide the geometrical interpretation for a tile of data permuted by butterfly jumping sequence. The mapping will be applied to a

Table 3.1: Butterfly permutation order for input of a tile of 8×8 pixels

	0	1	2	3	4	5	6	7
0	0	32	8	40	2	34	10	42
1	48	16	56	24	50	18	58	26
2	12	44	4	36	14	46	6	38
3	60	28	52	20	62	30	54	22
4	3	35	11	43	1	33	9	41
5	51	19	59	27	49	17	57	25
6	15	47	7	39	13	45	5	37
7	63	31	55	23	61	29	53	21

2D array of $2^n \times 2^n$ discrete data as a whole, without losing the desired attribute for online learning. We will also present an elegant formulation of this bijective mapping and generalize it to higher dimension so that the coordinate of data point in multi-dimensional space required at a specific time can be computed on demand for online learning.

As shown in Figure 3.1, a 4×4 array of pixels is partitioned into four tiles, each is an array of 2×2 pixels. We use the raster oriented coordinate system where $[0, 0]$ corresponds to the top-left corner pixel of the image. Then the first tile is formed by pixels with even column, even row indices, the second tile is similarly mapped from four pixels with odd column, odd row indices. The remaining tiles corresponds to odd column, even row indices and even column, odd row indices.

For an image consists of an array of $2^n \times 2^n$ pixels (otherwise, padding will be

Figure 3.1: Basic operation of global butterfly jumping sequence

Figure 3.2: Illustration of global butterfly jumping sequence

needed), we can repeat this process iteratively until every pixel in the original image is mapped into a specific position in the sequence, as illustrated in Figure 3.2. It is evident that the mapping between the pixel (in 2D array) and its final position (in 1D time tape) is bijective. In view of its geometrical property, we called this spatial to temporal mapping global butterfly jumping sequence (GBJS).

The resultant sequence has an interesting progressive resolution enhancement property. The first four data points in the sequence produces a subsampling image of four pixels from the original image. Along the sequence, the resolution of the image formed increases progressively. At the $1/4$ point of the 1D time tape, the

pixels visited correspond to a 2×2 down-sampling of the original image.

It is important to locate the coordinate of the pixel in the 2D array to produce the GBS at a particular time through some arithmetic and logical operations. For this, we will need to define some notations to establish the mathematical model of the bijection with reference to Figure 3.2. A multispectral digital image is a 2D image $M[x, y]$ which has a vector of discrete values at each spatial point or pixel. Individual pixel of an image can be addressed by its column number, x and row number, y . We use the raster oriented coordinate system where $[0, 0]$ corresponds to the top-left corner pixel of the image. For an image of $2^n \times 2^n$ pixels, the column and row numbers can be expressed as two n -bit binary numbers, $x = (x_{n-1}x_{n-2} \dots x_0)$ and $y = (y_{n-1}y_{n-2} \dots y_0)$, where x_i and y_i represent the i -th bit of x and y , respectively.

In Figure 3.2, each of the four subblocks descended from the original image are labeled with a unique 2-bit binary code “00”, “01”, “10” and “11” in natural number order from left to right. From the geometry of GBS described above, all pixels in the original image with the least significant bits (LSB) of their coordinates, $x_0 = y_0 = 0$ will be mapped into subblock “00”. All pixels with their coordinates fulfilling $x_0 = y_0 = 1$ will be mapped into subblock “01”. The criteria for pixel coordinates that fall into subblock “10” and “11” are respectively, $(x_0 = 1, y_0 = 0)$ and $(x_0 = 0, y_0 = 1)$.

If the subblocks consists of more than one pixels, they will be further partitioned. The descendent subblocks are identified by appending two more bits to the right of their predecessor identifiers. For example, if the identifier of a block is a binary string s , then its four descendent blocks from left to right will be successively

addressed as $s00$, $s01$, $s10$ and $s11$. The correspondence between the linear GBJs coordinate and the 2D spatial coordinate of the pixel is established in Table 3.2.

Table 3.2: Correspondence between $s_{2(n-i-1)+1}s_{2(n-i-1)}$ and x_i, y_i

$s_{2(n-i-1)+1}s_{2(n-i-1)}$	x_i	y_i
00	0	0
01	1	1
10	1	0
11	0	1

If we encode the time order at which a pixel at coordinate $[x, y]$ is visited under GBJs with an integer $s = (s_{2n-1}s_{2n-2} \dots s_0)$, it is noted from Table 3.2 that the tuple $s_{2(n-i-1)+1}s_{2(n-i-1)}$ is determined by x_i and y_i . This provides a way to compute the coordinates of the s -th pixel to be fed by GBJs. Mathematically, GBJs maps an integer sequence number, $s \in [0, 2^{2n} - 1]$ to a 2D spatial coordinate (x_1, x_2) , where $x_1, x_2 \in [0, 2^n - 1]$ are expressed as:

$$x_1 = \sum_{i=0}^{n-1} 2^{n-i-1} (s_{2i} \oplus s_{2i+1}) \quad (3.1)$$

$$x_2 = \sum_{i=0}^{n-1} 2^{n-i-1} s_{2i} \quad (3.2)$$

This GBJs can be generalized to m -dimensional space. The mapping from the integer sequence number, $s \in [0, 2^{mn}]$ to an m -dimensional vector $(x_1, \dots, x_m)$

where $x_1, \dots, x_m \in [0, 2^n - 1]$ can be expressed as:

$$x_j = \sum_{i=0}^{n-1} 2^{n-i-1} (s_{mi} \oplus s_{mi+j}) \quad \forall j \neq m \quad (3.3)$$

$$x_m = \sum_{i=0}^{n-1} 2^{n-i-1} s_{mi} \quad (3.4)$$

In the classical FFT algorithm, a 1D sequence permutation involving the bit reversal of original bit sequence is used to accelerate the calculation. In fact, this bit reversal operation is identical to the 1D to 1D global butterfly jumping sequence mapping. Thus, the proposed GBS can be seen as the generalization of 1D bit-reversal permutation sequence.

This global butterfly jumping sequence is used to feed data into our online learning algorithms for self-organizing topological maps. Its interesting property will also be studied and exploited in color halftoning application to be presented in Chapter 5.

3.3 Frequency-sensitive Self-organizing Maps (FS-SOM)

3.3.1 Introduction

Frequency Sensitive Competitive Learning (FSCL) is claimed to be effective in eliminating the under-utilization problem and can be easily implemented in hardware [96]. However, it lacks the topology preserving property of SOM, which is

very useful in coding and transmission. For example, in [116], the SOM topology is explored to further compress the encoded image to increase its coding efficiency. Despite SOM's advantages, its performance is often dependent on the optimal selection of training parameters, as mentioned in Chapter 2. Sometimes a small change in the setting of the SOM can result in a significant variation on its performance. Once the network is calibrated for a particular source of inputs, it is very difficult for the end user to re-adjust the SOM parameters to achieve the same optimal performance under an alternative source. It is equally important and preferable for the training algorithm to yield stable performance under different settings of network parameters. Besides circumventing the under-utilization problem, the proposed FS-SOM is also designed to sustain a robust performance. Next, the details of each step of the FS-SOM algorithm will be described with a 1D string topology for the color quantization problem.

3.3.2 Initialization of FS-SOM

Because of the topological neighborhood of FS-SOM, initialization of neuron weights is not as crucial as for some other clustering algorithms, like LBG. A better initialization scheme will enable the network to converge faster but it requires a priori knowledge of the input data distribution. If nothing or little is known about the input data at the time of initialization, we initialize the neurons of FS-SOM at equally spaced points along the major diagonal of the input space, i.e,

$$\mathbf{w}_i = [(i - \frac{1}{2}) \times \frac{255}{N}, (i - \frac{1}{2}) \times \frac{255}{N}, (i - \frac{1}{2}) \times \frac{255}{N}]^T \quad \forall i = 1, 2, \dots, N. \quad (3.5)$$

where N is the number of neurons. This initialization method is very simple. Nevertheless, it provides a better initial topological order of FS-SOM to facilitate

faster convergence than the random initialization.

3.3.3 *Input Sequence*

Online learning causes the learning performance to be order dependent, particularly when the training set contains spatially correlated data. Therefore, training the FS-SOM with a raster scanning order of the input pixels has a strong tendency to over-train some groups of neurons. Here, the Global Butterfly Jumping Sequence (GBJS) of the image data is used as input sequence to let the neurons learn the characteristics of the training source as early as possible to prevent the performance from being degraded by the order-dependent learning. The mapping of GBJS has been detailed in Section 3.2. The difference in the permutation effect of the blockwise localized jumping sequence [116] and our proposed GBJS is illustrated in Figure 3.3. To make the distribution of the permuted data segments visually perceivable, the data of both temporal sequences are read back in raster scan order to form a 2D image. Their spatial characteristics are then compared with the original image from which they mapped. A gray scale ramp image and a natural image (Lena) are used for this comparison in Figure 3.3(a). The results show that under the GBJS (see Figure 3.3(c)), the training data are more evenly diffused than the fractal scanning sequence of Figure 3.3(b). Therefore, the proposed GBJS will ensure that maximally distinctive data are fed in succession. Consequently, the quantization results will be improved over those without using the GBJS for online learning.

Figure 3.3: (a) the original image (b) image permuted by the localized butterfly jumping sequence (c) image permuted by the global butterfly jumping sequence

3.3.4 Winning Neuron Search

To allow every neuron to compete fairly and learn adequately irrespective of its winning sequence, a modified distortion metric, $d_i = f(u_i)\|\mathbf{x}(t) - \mathbf{w}_i(t)\|^2$ is defined in FSCL [96] where an update counter, u_i is used by each neuron to keep track of the number of times it wins. The function, $f(u_i)$ is designed to improve the competitiveness of the less frequently updated neurons. As our FS-SOM has incorporated several effective measures like neighborhood adaptation, frequency sensitive update, and dead neuron reinitialization to overcome the under-utilization problem, we can avoid the difficulty of designing an optimal $f(u_i)$ by adopting the standard SOM best marching unit search with $f(u_i) = 1$. In what follows, the techniques used to overcome the under-utilization problem will be discussed.

3.3.5 Updating of Winning Neuron and Its Neighborhood

The weight vectors and the update counters of the winning neuron and its neighborhood are adjusted according to the following equations:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + G(u_c)H(i, c, m)[\mathbf{x}(t) - \mathbf{w}_i(t)] \quad (3.6)$$

$$u_i(t+1) = u_i(t) + H(i, c, m) \quad (3.7)$$

where $G(u_c)$ is the frequency sensitive learning rate function. It is a monotonic decreasing function of u_c , and its value is usually bounded in the range (0, 1). $H(i, c, m)$ is a tapering neighborhood function which is a hallmark of the SOM algorithms. This neighborhood taper alters the update of each neuron, i within the neighborhood of the winning neuron, c by a different amount at discrete time

interval, m depending on the distance between the weight vectors $\mathbf{w}_i$ and $\mathbf{w}_c$. It is this neighborhood tapering that gives SOM its topological ordering property. In our implementation, the following neighborhood taper used in [116] has been selected:

$$H(i, c, m) = \exp\left(-\frac{\|\mathbf{w}_i - \mathbf{w}_c\|^2}{\sigma^2}\right) \quad (3.8)$$

$$\sigma = N\beta k^m \quad (3.9)$$

where m represents the number of sweeps. A sweep is a period during which the updating parameters remain constant. In our simulations, we set the number of input data in one sweep to be $12 \times N$ where N is the number of neurons. Thus on average, each neuron will be updated 12 times as the winning neuron in a sweep. $\sigma(m)$ is the neighborhood width at the sweep number m . It defines the boundary of the lateral interaction between the winning neuron c and other neurons. β defines the extent of the initial neighborhood interaction. It is usually set to a value between 0 and 0.5, where $\beta = 0.5$ implies that the neighborhood interaction exists in the entire network initially, and $\beta = 0$ implies that there is no neighborhood interaction at all. The constant k is used to control the rate at which the boundary of the neighborhood shrinks, and is usually set to a number less than 1.

In Equation (3.6), the use of a frequency sensitive learning rate function implies that the training time for each individual neuron, rather than the training time of the network, is used to modulate the distance metric for updating. As this mode of learning uniquely characterizes our FS-SOM from other SOM algorithms, it is important to establish a “fast adaptation and slow forgetting” learning function to ensure that the weights settle well during training. We experiment with the nonlinear reciprocal function, $G(u_c) = u_c^{-l}$ for different values of $l > 0$. It is

Figure 3.4: Comparison of different learning rate functions.

found that $l = 0.5$ gives excellent results for different test images and it is set as the default value for our simulation. The comparison of the proposed learning rate function and some popular learning rate functions [28, 116] used by other researchers is shown in Figure 3.4. It is noticed that the proposed learning rate function strives to drag the winning neuron towards the input cluster as quickly as possible within its first 10 to 50 updates, thereafter the neuron is allowed to fine tune gradually in its later wins. For other learning rate functions, the neurons are almost frozen after 50 times of winning.

The intrinsic learning mechanism of our proposed FS-SOM algorithm is fundamentally different from the training algorithm proposed by Pei and Lo [116]. The proposed learning rate function $G(u_c)$ is dependent on the update counter value

of the winning neuron. Hence, although the neurons compete globally across the network, training is adapted locally to each neuron. Whereas for the SOM algorithms, such as that of [116], the learning rate function is time dependent. As a consequence of the globally applied monotonic decreasing learning rate function, some of the neurons may be over trained at the beginning of the training process. The net result is that when the training process is completed, the late winners may become underutilized in the pixel mapping process.

By measuring the mean Euclidean distance between the weight vectors of SOM from one time step to another at discrete interval during the training period, it is observed that the choice of $H(i, c, m)$ has a significant influence on the performance stability of SOM. In contrary, supplemented by the frequency sensitive learning, the proposed FS-SOM is less sensitive to poorly calibrated network parameters in view of its stability in vector quantization performance. Even with the simplistic uniform or linear neighborhood function, the frequency sensitive learning scheme will still manage to resolve the under-utilization problem to a large degree. The robustness of vector quantization performance of our FS-SOM over various functions of $H(i, c, m)$ will be demonstrated in the simulation results to be presented later in Section 3.4. However, a suitable neighborhood function can still accelerate the topological ordering of FS-SOM as in SOM. For this reason, a Gaussian neighborhood taper of $H(i, c, m)$ used by [116] has been selected as Equation (3.8).

3.3.6 Convergence Criteria

At the end of each sweep, the weight vectors of the neurons in the current and the previous sweeps are compared. If their difference is less than a predefined small threshold value, then the training is completed. The convergence criterion can be stated as:

$$\frac{\sum_{i=1}^N \|\mathbf{w}_i(m) - \mathbf{w}_i(m-1)\|^2}{N} \leq \epsilon \quad (3.10)$$

where ϵ is an infinitesimally small number. In our simulation, $\epsilon = 0.01$. In other words, if the average square difference of the neuron weights is less than 0.01, the training is terminated. In Section 3.4, the effects of varying ϵ on the simulation time and performance of the proposed FS-SOM and competing algorithms are studied.

To further increase the stability of the algorithm, we will look for dead neurons ($u_i = 0$) at the end of each predefined training interval. If dead neurons exist, they will be reinitialized. Otherwise, the learning process will be continued without further checking for dead neuron. By reinitialization, the weight of the most frequently updated neuron is copied to the first dead neuron. The weight of the second dead neuron will be reassigned the weight of the second most frequently updated neuron, and so on. The update counters of both neurons in each pair of neurons (one dead neuron and one corresponding frequently updated neurons) will be reset to half of that of its corresponding frequently updated neuron in each pair. If the network parameters are well selected, this reinitialization operation will never be needed. However, if one or more of the parameters cause SOM to adapt poorly, the existence of the dead neurons will quickly be detected and corrected by this reinitialization process.

In [28], a similar concept but different approach is adopted. The major difference is that: in [28], after checking and reinitializing the dead and inactive neurons, the update counters of all neurons are refreshed to zero and a new learning cycle is started. In our proposed algorithm, we halve the update counter and continue with the learning instead of resetting them to zero and starting a new learning cycle. In this way the detection and reinitialization is more purposeful, faster and more effective. Furthermore, we only detect and reinitialize the neurons with zero update rather than all neurons with low winning frequency. This is because the frequency sensitive competitive learning rate function has allowed the less competitive neurons to learn adequately. Only the dead neurons, which have no opportunity to update at all, need to be dealt with. Another reason is that, the minority color can be very important in some color quantization applications. For example, in the images of roadmaps or landscapes, it is useful to preserve the minority colors of prominent landmarks with less utilized but sufficiently trained neurons. It is unwise to sacrifice them completely to trade for a slightly lower global distortion.

3.4 Simulation Results and Discussion

For the purpose of comparison, the original 24 bits per pixel true color images of Lena, Baboon and Pepper as shown in Figure 3.5 are used for the simulation. Lena is a natural image with smooth color transitions. Baboon is an image demarcated by rich high frequency texture with a much broader color range. Pepper has a profusion of small color variations and the illumination on the object surfaces has created a smooth dispersion of colors. The color distributions of these images in

Figure 3.5: Test images: (a) Lena, (b) baboon and (c) pepper

the RGB space are shown in Figure 3.6, where R, G and B represent the three color channels of red, green and blue, respectively.

Figure 3.7 shows the 1D topological maps of the final adaptation results obtained from a 64-neuron FS-SOM for the three different color images. Neighboring neurons linked by solid lines represent adjacent entries in the color palette. The absence of long links indicates a good color-ordering palette, which is a good attribute for speeding up the encoding (pixel mapping) process by local search. By superimposing the topological maps of Figure 3.7 on the color distribution of the original images in Figure 3.6, it is observed that more neurons are assigned to the denser color regions and less neurons are scattered wider and evenly over the sparse color regions. This implies that the color palette generated by FS-SOM is able to match the pixel color distribution very well.

To assess the perceptual quality of the color images reconstructed from the color

Figure 3.6: The color distributions of (a) Lena (b) Baboon and (c) Pepper

Figure 3.7: Resultant 1D topological maps of a 64-neuron FS-SOM: (a) Lena, (b) Baboon, (c) Pepper

palettes generated by FS-SOM, the images of Lena, Pepper and Baboon palettized with different codebook sizes are shown in Figure 3.8. No dithering operation [130] has been applied on these images in order to evaluate the real performance of FS-SOM in clustering the data to match each separate color. Even without post-processing by dithering operation, there is hardly any observable false contouring artifacts for all the reconstructed images generated from the trained FS-SOM of 64 neurons and above. Comparing with the original full color images of Figure 3.5, for the 16- and 32- color images, minor contouring and false edge artifacts can be observed on the rim of the hat and arm of Lena, the cheek of Baboon and the surfaces of the large green and red peppers due to the limited number of colors. Interestingly, the perceivable artifacts appear to be less annoying in the high texture Baboon image than the other two smooth images despite it has a denser and wider color distribution (see Figure 3.6). The less noticeable contouring artifacts may be due to the fewer and smaller regions of smooth color transitions of Baboon.

We can observe the adaptation behavior of the weights at various discrete points by measuring the mean Euclidean distance, $\delta(m)$ between the weight vectors of FS-SOM calculated at successive sweeps during the training process, i.e.,

$$\delta(m) = \frac{\sum_{i=1}^N \|\mathbf{w}_i(m) - \mathbf{w}_i(m-1)\|}{N} \quad (3.11)$$

The mean change in Euclidean distance $\delta(m)$ plotted against the sweep number is shown in Figure 3.9 for different number of neurons. The solid lines represent the $\delta(m)$ responses of FS-SOMs with different 1D string configuration network sizes ranging from 16 to 256 neurons, and the result of the 16-neuron 1D string SOM of [116] is plotted as dotted line for comparison. This figure demonstrates

Figure 3.8: Reconstructed images for (a) Lena, (b) Pepper, and (c) Baboon under different codebook sizes: (from top to bottom) 16, 32, 64 and 128 colors.

the difference between the learning philosophies of SOM and FS-SOM. In SOM, the mean change in Euclidean distance $\delta(m)$ reduces with time gradually and steadily under an optimally tuned learning rate function which is time-decreasing and neuron-independent. Consequently, the longer the elapse time, the smaller the aggregate change in neuron weights between successive sweeps. Depending on the descending rate of the learning function, such brute force convergence (learning rate diminishes to zero as time elapses) does not necessarily guarantee that all neurons are adequately trained upon convergence. On the other hand, the learning rate function of our FS-SOM is dependent on the winning neuron's update counter. The learning rate is large when a less frequently updated neuron wins at the time of observation and the learning rate is small when a more competitive neuron is being updated. The surges on the curves correspond to the updates of infrequently win neurons. The $\delta(m)$ responses undergo three different phases. During the initial fast adaptation phase, the competitive neurons corresponding to majority colors or denser clusters are rapidly updated and settled. During the equalization phase, the skimpily updated neurons correspond to minority colors or sparse clusters are given opportunity to order themselves, giving rises to a number of spikes. After which FS-SOM enters into a final settlement phase where the response of $\delta(m)$ dips steeply till convergence. This equalization phase and the occasional drastic fluctuations of $\delta(m)$ response clearly marks the unique neuron-dependent learning behavior of FS-SOM.

The effects of varying ϵ on the performance of FS-SOM and SOM are given in Figure 3.10. The test image is 512×512 color image Lena, and the codebook size is set to 16. The results show that SOM performs better than FS-SOM when the threshold is large, but it takes longer time than FS-SOM to converge when the same large threshold is used. On the other hand, FS-SOM gives higher

Figure 3.9: Mean change of neuron weights during the training process.

Figure 3.10: Effects of varying ϵ on the performance of FS-SOM and SOM.

reconstruction quality when small threshold is used for applications that emphasize on high quality reconstructed images. Tests on other images and codebook sizes also indicate similar trend.

The effect of reinitialization is analyzed in Figure 3.11. Two situations of under-utilization problem are created by training the smooth images of Lena and Pepper with poor learning parameters. The Peak Signal to Noise Ratio (PSNR) of the reconstructed image is used to measure the quality of the reconstruction during the training process. The PSNR is defined as follows:

$$PSNR = 10 \times \log \frac{255^2}{MSE} \quad (3.12)$$

Figure 3.11: Learning tracks of 16-neuron FS-SOM with and without reinitialization: (a) Lena, (b) Pepper

where $MSE = \sum_{i=1}^N (x_i - x'_i)^2 / N$. x_i and x'_i are the pixel values of the original and the reconstructed images, respectively. N is the total number of pixels.

In Figure 3.11, the trends of reconstruction performance during training for FS-SOM with and without reinitialization are marked respectively by the solid and dotted lines. Reinitialization boosts the PSNR performance of FS-SOM significantly above that without the reinitialization. As shown in Figure 3.11(a), although the training of FS-SOM without reinitialization converges earlier, its final PSNR is far worse. Reinitialization assists FS-SOM to adapt the dead neurons proactively during the fast ordering phase and leave the fine adaptation of the neurons to the frequency sensitive learning scheme and neighborhood adaptation, so that when the training finally converges, a good reconstruction quality can be achieved.

The performance of proposed learning method, FS-SOM is evaluated and compared with the SOM algorithm proposed in [116] by examining the quality of the reconstructed images. FSCL algorithm given in [96] and the celebrated batch algorithm of LBG [100] are also implemented and tested. The convergence criteria for different algorithms are kept the same. All the simulations were run on a Personal Computer equipped with a Pentium Pro. 2.1 GHz Processor and 512 MBytes of system memory. The results are shown in Table 3.3. 1D string topology has been used for both SOM and FS-SOM. As noted in [116], the performance differences of calibrated 1D string and 2D mesh structures are trivial. For consistency, all algorithms in Table 3.3 are trained using the global butterfly jumping sequence described in Section 3.2. This will decouple the effect of input sequence from affecting the analysis of the results due to the core features of the methods being compared. To allow more versatile evaluations on different settings and images than those considered in [116], and we have implemented the SOM based on the description of [116]. As the data are generated using our best implementation of [116] and global butterfly jumping sequence instead of taken directly from [116], slight discrepancy on the PSNR values of Table 3.3 and those of [116] obtained from the same images and settings is expected.

From Table 3.3, it can be seen that the basic competitive learning scheme, FSCL gives the worst performance due to its lack of neighborhood updating. LBG delivers much better performance than FSCL. However, being a batch algorithm, LBG needs the entire training set to be presented before performing an update (though some data squashing and reliable sampling techniques can be used to alleviate this problem) and has long computation time. In comparison with the SOM and proposed FS-SOM algorithms, LBG and FSCL is more sensitive to initialization. The remaining algorithms all have the advantages of fast convergence and on-line

Table 3.3: PSNR (in dB) of various algorithms with different codebook sizes

Codebook Size	Image	FSCL	LBG	SOM	FS-SOM
16	Lena	29.40	29.65	29.65	29.67
	Baboon	24.70	24.91	24.85	24.89
	Pepper	26.50	26.74	26.88	26.91
32	Lena	31.88	32.14	32.14	32.16
	Baboon	26.88	27.16	27.09	27.13
	Pepper	28.66	29.24	29.28	29.33
64	Lena	34.09	34.25	34.25	34.25
	Baboon	28.99	29.16	29.09	29.15
	Pepper	31.14	31.26	31.47	31.65
128	Lena	36.00	36.08	36.23	36.29
	Baboon	30.90	31.08	31.04	31.09
	Pepper	32.85	33.49	33.67	33.70
256	Lena	37.77	37.78	37.74	37.97
	Baboon	32.82	33.00	32.85	33.00
	Pepper	35.25	35.40	35.55	35.58

learning capability. Despite the already superb quality of reconstruction offered by the state-of-the-art SOM algorithm in CQ [116] with enhanced global butterfly jumping sequence, our proposed FS-SOM algorithm can still outperform it slightly for all the different color palette sizes and images simulated. This slight improvement can be attributed to a well blend of neighborhood updating and frequency sensitive learning.

Table 3.4 lists the training time of the different algorithms. From Table 3.4, it is observed that the computation time of the proposed algorithms are compatible to those of the competitive learning schemes, including the basic FSCL and SOM. The classical LBG is much more time consuming due to its batch updating nature. Some techniques [80, 142] can be used to speed up the LBG algorithm by intelligently reducing the number of unnecessary distance calculations. However, many of these techniques are equally applicable to the SOM and FS-SOM algorithms, giving similar or better acceleration than when they are applied to LBG. For this reason, the comparison is done between the classical LBG and other algorithms without applying the reduction techniques.

One distinct advantage of our FS-SOM algorithm is its performance robustness. As the performances of SOM and other quantization algorithms approach their optimum, they become highly susceptible to the variation of parameters and input data distribution. Besides the parameters that control the learning rate function which have already been discussed in Section 3.3 before, two other most important parameters common to the FS-SOM and SOM [116] training algorithms are β and κ . These parameters determine the neighborhood interaction $\sigma(m)$ and they need to be empirically tuned to achieve optimal quantizer performance. The performance variations over a reasonable range of these two parameters are studied for

Table 3.4: Training time (in seconds) of various algorithms with different codebook sizes

Codebook Size	Image	FSCL	LBG	SOM	FS-SOM
16	Lena	0.4	96.1	0.4	0.4
	Baboon	0.4	99.0	0.3	0.5
	Pepper	0.4	113.4	0.3	0.3
32	Lena	3.2	232.5	1.2	1.2
	Baboon	3.0	319.1	1.1	1.3
	Pepper	3.4	355.4	1.6	1.2
64	Lena	5.2	563.6	4.0	4.4
	Baboon	5.6	817.4	4.2	4.7
	Pepper	5.8	1484.3	4.3	4.3
128	Lena	22.4	1623.8	18.6	16.5
	Baboon	21.3	2009.4	18.7	16.7
	Pepper	22.6	2212.5	16.6	18.9
256	Lena	73.4	3783.6	61.2	58.8
	Baboon	76.2	4423.9	56.1	63.0
	Pepper	75.1	3806.5	77.0	61.0

Figure 3.12: Palettized image quality under different combinations of β and κ : (a) SOM, (b) FS-SOM

FS-SOM and SOM. The simulation results are compared in Figure 3.12. The test image is Lena, and the size of the network is 16.

Figure 3.12 shows that the performance of SOM fluctuates apparently with the training parameters. In the best case, it can achieve a PSNR value of 29.62 dB. Its worst PSNR can drop dramatically down by more than 3 dB to around 26.30 dB. On the contrary, our proposed FS-SOM algorithm has a very flat PSNR response at around 29.60 dB over all combinations of β and κ . The difference between the peak and valley of Figure 3.12(b) is less than 0.4 dB.

The topology mapping property of FS-SOM is also studied and compared with the state-of-the-art SOM in CQ [116] under different sets of parameters. In the 1D string neighborhood structure, the topological order can also be measured by

Figure 3.13: Topology quality metric, J under different combinations of β and κ : (a) SOM, (b) FS-SOM

an objective function J , which is defined as [88]:

$$J = \sum_{i=2}^N \|\mathbf{w}_i - \mathbf{w}_{i-1}\| - \|\mathbf{w}_N - \mathbf{w}_1\| \quad (3.13)$$

where N is the number of neurons and $\mathbf{w}_i$ is the weight vector of the i -th neuron. The lower the value of J , the better the codebook is ordered. The results are shown in Figure 3.13.

In Figure 3.13, we can see that FS-SOM is also robust in its topological ordering performance under different settings of parameters. The J value of SOM is very unstable, varying from 13 to 1173 in Figure 3.13(a). Unfortunately, the best topological map of SOM gives the worst quality, as can be seen in Figure 3.12(a). On the other hand, the J value of our proposed FS-SOM in Figure 3.13(b) is very stable, ranging from 293 to 383. FS-SOM gives smaller J than SOM in 24 out of the 30 combinations of β and κ , and of the 6 combinations for which SOM has

smaller J values, its reconstruction quality happens to be the worst as can be seen from Figure 3.12(a).

An objective comparison of FS-SOM and SOM in their robustness in topology preservation is shown in Table 3.5. It compares the mean and standard deviation of PSNR and J value produced by SOM and FS-SOM with widespread combinations of parameters for three different test images. 150 parametric combinations of β ranging from 0.05 to 0.5 in step of 0.05 and κ ranging from 0.15 to 0.85 in step of 0.05 are used. The network size is fixed at 16. The results show that the performance of FS-SOM is more robust against parameter variations, particularly for the topology quality. This is congruent with the subjective evaluation of Fig. 3.12 and Fig. 3.13.

Table 3.5: Mean and standard deviation of PSNR and J value for three test images

		Lena		Baboon		Pepper	
		mean	σ	mean	σ	mean	σ
SOM	PSNR	29.39	0.22	24.85	0.01	26.67	0.16
	J	991.94	185.29	1505.95	215.23	1409.22	271.62
FS-SOM	PSNR	29.57	0.15	24.88	0.01	26.76	0.11
	J	386.88	47.85	712.20	109.18	955.58	213.62

Because of the topological ordering property of SOM and FS-SOM, the encoded image after the pixel mapping process will still maintain a strong spatial correlation as in the original image. The entropy of the encoded images generated by the 256-neuron SOM and FS-SOM are tested by subjecting their indices to CALIC [155], a competitive lossless image coding algorithm. The encoded images

that produce the PSNR values in Table 3.3 are used. The compression results are shown in Table 3.6. Higher compression ratios are achieved by FS-SOM for all the encoded images. This shows that FS-SOM is able to reduce the bit-rate of the encoded image without scarifying its reconstruction quality, and these merits are sustainable under a wide spectrum of combinations of β and κ .

Table 3.6: Compression ratio of the indices of the encoded images compressed by CALIC

	Lena	Pepper	Baboon
SOM	1.59:1	1.58:1	1.20:1
FS-SOM	1.67:1	1.74:1	1.27:1

3.5 Chapter Summary

In this chapter, we have instilled a novel competitive learning scheme with stable performance into the well known Self-organizing Maps (SOM) employed for vector quantization.

The proposed FS-SOM harmonically blends the neighborhood adaptation of the SOM with the neuron dependent frequency sensitive learning model, the global butterfly permutation sequence for input randomization, and the reinitialization of dead neurons to harness effective utilization of neurons. The net effect is an improvement in adaptation, a well-ordered color palette and the alleviation of underutilization problem, which is the main cause of visually perceivable artifacts

of CQ.

Extensive simulations have been performed to analyze and compare the learning behavior and performance of FS-SOM against other vector quantization algorithms. The results show that the proposed FS-SOM outperforms classic competitive learning, LBG and SOM algorithms. More importantly, FS-SOM achieves its superiority in reconstruction quality and topological ordering with a much greater robustness against variations in network parameters than the current art SOM algorithm for CQ.

Chapter 4

Self-organizing Topological Tree

4.1 Introduction

The frequency-sensitive self-organizing maps presented in the last chapter improves the SOM performance for vector quantization and achieves good topology ordering with robustness against variation in key network parameters. In this chapter, an online self-organizing topological tree (SOTT) with faster learning will be proposed. A large portion of this chapter has been published in the *Special Issue on Learning in Computer Vision and Pattern Recognition, IEEE Transaction on Systems, Man and Cybernetics, Part B: Cybernetics* [159] and presented at the *2004 IEEE International Symposium on Circuits and System* [158].

In this chapter, we propose a new multi-resolution Self-organizing Topological Tree

(SOTT) to accelerate the search procedure. As in other tree structured search algorithms like BT-SOM [30] and TS-SOM [91, 90], the structure of their trees are very similar. What differentiates their algorithms as well as ours clearly is the way the search and training are performed. In our proposed SOTT algorithm, neurons on different layers are organized concurrently. Two problems have to be overcome before the SOTT can be made amenable to on-line learning. First, the tree-structured search procedure is usually globally suboptimal and fails to find the real best matching neurons during training. In other words, the winner of the network is not necessarily the closest neuron to the input vector. Second, by imposing the structural constraints with a hierarchically connected lattice, two kinds of neighborhood relationship co-exist in the network: the inter-layer parent-children relationship and the intra-layer sibling relationship. As the synaptic weights of the neurons are updated during training, it is a challenging problem to maintain both topological relationships.

A multi-path search as in [27] can be used to overcome the first problem. In most tree-structured algorithms, the winner of one layer is selected from among the children of the winner of its upper layer. This has the drawback of creating coactivation of responses. The multi-path search is used to overcome the bias introduced by a single path tree search algorithm. The performance enhances as the bias reduces effectively with increasing number of paths, at the expense of the increased computation burden. The multi-path search has been proven to be a good method to control the trade-off between the performance and computation complexity for large trees.

The second problem is more intricate. We tackle this problem by introducing a novel concept of the winning path as opposed to the winning neuron in the

classic SOM. The neurons on the winning path are updated in accordance to the learning rates at each layer. The winners and their inter- and intra-layer neighborhood neurons are updated by specifically designed learning rules to keep both the parent-children relationship and sibling relationship intact. There is still a dilemma: the closest neurons to the input vector on each layer usually do not lie on the same path. There is no easy way of deciding an optimal winner path in this situation. What we have proposed is to use a multi-path search for a winner on the last layer, and the path to this leaf is deemed the winner path. In some applications, however, it is also important to consider the intermediate layers. We have introduced a special parameter to modulate the adaptations at different layers. The proposed updating procedure has another advantage of mitigating the bias brought by limited-path search. This is because with an appropriate modulation, the neurons on a winner path are more likely to learn by the same degree from the input data, so the distances between them will be reduced. Consequently, less bias will be introduced in the subsequent updates.

To effectively alleviate the three clustering problems associated with SOM, we suggest the use of the proposed SOTT in hybrid clustering scheme. In hybrid clustering scheme, a low complexity partition clustering algorithm is first applied to reduce the large amount of data before the computational intensive agglomerative hierarchical clustering (AHC) algorithm is used to merge the resultant clusters progressively to obtain the dendrogram [15]. An AHC algorithm will progressively merge sets of data points, and yields a dendrogram, which is a nested series of partitions representing the merging progress of data points [74]. One important factor for hybrid clustering scheme is the linkage metric, which is a proximity measure used to merge subsets rather than individual points in AHC. The linkage metric directly influences the effectiveness and computation complexity of AHC.

However, majority of the existing effective linkage metrics require a time complexity of $O(N^2)$ and can only be calculated after all patterns have been labeled. This makes AHC computational intensive comparing with partitional clustering techniques like k -mean and SOM. Hybrid clustering, nevertheless, inherits the efficiency of the partition clustering and the prowess of discrimination of AHC, and it had been successfully used in several clustering schemes before [15, 99, 148]. The proposed SOTT hybrid clustering is based on the SOTT learning and an AHC stage with a novel linkage metric. The computation of this novel linkage metric blends well with the winning selection phase of the SOTT, thus it can be evaluated on-line and much faster.

The rest of this chapter is organized as follows. The structure of the SOTT and the basic terminologies used are presented in Section 4.2. In Section 4.3, the proposed learning algorithm of the SOTT is described. A unified partition and AHC algorithm based on SOTT is presented and a new linkage metric for the proposed hybrid clustering on SOTT is discussed in Section 4.4. Vector quantization of images is taken as an example to test the learning ability of the proposed SOTT. The results are compared with the classic SOM algorithm in Section 4.5. The clustering performance of the SOTT hybrid clustering is also evaluated. Section 4.6 concludes the chapter.

4.2 Structure of SOTT and Related Terminologies

The simplest SOM network consists of an input layer connected to a 1D linear array of neurons in the WTA layer. SOMs encountered in practice are more commonly structured with neurons typically arranged on a 2D square or hexagonal grid laid out in a lattice. As the number of neurons increases, the growth of the search space calls for a multi-layer SOM to ease the network from the time consuming training and mapping processes. Irrespective of the geometrical forms of the lattice, as the dimensionality of the data distribution increases, direct extension of the basic training algorithm to a stack of SOM layers is inadequate to cluster the input with a fixed rate TSVQ because the final codebook structure will become highly irregular and unpredictable. In this section, we describe the structure of a Self-organizing Topological Tree (SOTT), and define some terminologies specific to the SOTT. As explained earlier, the structure of SOTT is based on a generic tree, similar to the one proposed in [91]. The main difference and feature lies in the learning algorithm.

A static SOTT can be viewed as a multi-layer SOM, with fixed depth and breadth. A SOTT realizes the mapping of an n -dimensional Euclidean space $\mathbb{R}^n$ onto a finite codebook space $\mathcal{W}$:

$$\Lambda : \mathbb{R}^n \rightarrow \mathcal{W} \quad (4.1)$$

where the input vector $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$ and $\mathcal{W} = \{\mathbf{w}_{ij} \in \mathbb{R}^n \mid i = 1, 2, \dots, L \text{ and } j = 1, 2, \dots, N_i\}$ define a finite set of codevectors, $\mathbf{w}_{ij}$ which are points in $\mathbb{R}^n$. The structure of the SOTT network is characterized by L , the

Figure 4.1: Structure of SOTT (a) 2D quad tree [91] (b) 1D string

number of layers and N_i is the number of neurons at the i -th layer for each $i = 1, 2, \dots, L$. A 2D Quad-tree structured SOTT with $L = 3, N_i = 4^{i-1}$ is shown in Figure 4.1(a). Figure 4.1(b) shows a 1D string structured SOTT with the same number of neurons.

The number of layers, L , of an SOTT is called the depth of the tree. For a static SOTT, the number of children per neuron, B , is fixed and is called the branching factor. The first layer with N_0 neurons is called the root. In general, the i -th layer has $B^{i-1} \times N_0$ neurons. The last layer is called the leaf layer and it has $B^{L-1} \times N_0$ neurons. The neurons on the leaf layer are known as the leaf-neurons.

Two kinds of relationships exist in a SOTT: (1) the intra-layer neighborhood relationship, (2) the inter-layer parent-children relationship. Let G_i be a fully connected graph where its vertices and edges are defined respectively by the neurons and their interconnections at the i -th layer. From the formulation of a near-neighbor mesh of a finite lattice, we define the k -neighborhood N_v of the neuron $v \in G_i$ as the spatial support of the position of v in G_i . A neuron, u is said to be

in the k -distance neighborhood of the neuron v if there is a connected path from u to v and their minimal Euclidean distance, $\|u - v\| \leq k$.

Furthermore, for each non-leaf neuron $v \in G_i$, a subtree H_v of neurons rooted at v with a branch factor of B can be found. A neuron u is said to be a child of v iff $v \in G_i$ and $u \in G_{i+1} \cap H_v$. The neurons of $G_{i+1} \cap H_v$ that have the same parent neuron v , are called the siblings.

It should be noted that, the neighborhood, $N_c(t)$ of a winning neuron, v for each layer is monotonically decreasing with the training epoch time, t . Using a fixed neighborhood or starting the training with a small neighborhood results in a map which is not globally ordered and consists of small patches of local ordering between which the ordering direction changes discontinuously. The temporal function of neighborhood improves the spatial resolution of the map without losing the acquired global ordering through the proposed layer updating process. In addition, the winning neuron and its neighbors will not be updated in a uniform manner. The non-uniform function that alters the update of each weight vector within the winning neighborhood by an amount relative to its topographical distance from the winning neuron is called the neighborhood taper. Because of the parent-children relationship, the winning neurons of each adaptive layer in the winning path must inherit the weight vector of their parents. New parameters are introduced in the search for the winning path during training and the neighborhood tapers for updating the winning path are tailored to maintain this close coupling of the parent-children relationship.

4.3 Training Algorithm of SOTT

Training is a process whereby a set of input vectors are presented to the network one at a time to cause the synaptic weight vectors of a group of neurons in the lattice to be tuned. The SOTT as defined earlier is a fully connected, feed-forward network for which the spatial arrangement of the trained neurons conveys significant meaning. Upon training, the parent neuron has similar but coarser resolution than its children and the siblings have similar synaptic weight vectors of finer resolution. The resolutions of the synaptic weights increase as the neurons descend from the root to the leaf layers. Within the same layer, neighboring neurons have similar weight vectors and as the neurons are further apart, their weight vectors become discriminable. Our proposed algorithm adapts the synaptic weight vectors of the SOTT via an on-line incremental learning scheme. The main steps involved in the training algorithm are illustrated in Figure 4.2.

4.3.1 Initialization of SOTT

Because of the topological neighborhood of SOTT, initialization of SOTT is not as crucial as for some other vector quantization algorithms, e.g., GLA. A better initialization scheme will enable the network to converge faster by using a priori knowledge of the input data distribution. For example, in vector quantization (VQ) of images, the neurons of the 1D string SOTT can be initialized with uniformly distributed weight vectors on the major diagonal of the input space:

$$\mathbf{w}_{ij} = \left[\frac{(j - \frac{1}{2}) \times 255}{N_i}, \dots, \frac{(j - \frac{1}{2}) \times 255}{N_i} \right]^T \quad \forall i = 1, 2, \dots, L \text{ and } j = 1, 2, \dots, N_i \quad (4.2)$$

Figure 4.2: Major steps of the SOTT training algorithm

where $\mathbf{w}_{ij}$ represents the weight vector of the j -th neuron on the i -th layer. This initialization method is very simple, but it provides a good neighborhood and parent-children relationship among all neurons on the lattice.

4.3.2 Butterfly Permutation

For online learning, we use the global butterfly jumping sequence (GBJS) proposed in Chapter 3 for input randomization. The GBJS input permutation has been detailed in Section 3.2 and it will not be reiterated here.

4.3.3 Search for the Winning Path

The updating of the winning neuron and its neighborhood at each layer will not be performed until a winning path has been identified for each input. To trace the winning path, we need to search for a single winning leaf.

Our proposed search algorithm uses two key parameters, λ and κ , to bias the competitiveness of some layers and emulate the positive effect of a multi-path search. This is unlike SOM [88], TS-SOM [90, 91] and other TSVQs [21, 30, 58], where the search for best matching units is performed layer-by-layer, in isolation. The proposed algorithm for the search for a best matching leaf neuron and the winning path is shown in the pseudo-code in Figure 4.3. The overall idea of the algorithm in Figure 4.3 is to find the *winning_child* neurons progressively on each layer, until a winning child at the leaf layer (*win_leaf*) is found. Then the path to the *win_leaf* is set as the winning path (*win_path*).

In Figure 4.3, the subroutine *Compute_error*($\mathbf{x}(t), i, v, e_v$) computes the distortion between the input $\mathbf{x}(t)$ and the B children of the parent neuron, v at level i . The weight vector of a child neuron u of v is denoted by $\mathbf{w}_u$ and its error (distortion) is computed by the Euclidean distance $\|\mathbf{x}(t) - \mathbf{w}_u\|$ biased by a fraction of its parent error, e_v . The fractional constant, called the leakage ratio λ_i is introduced so that adjacent layers, instead of only one layer, are considered in finding the best matching neuron at the leaf layer. The value of λ is fixed at each layer but varying across layers to facilitate the control of performance trade-off at different layers. By tuning the leakage ratio λ_i , some intermediate layers can be emphasized according to the specific requirement of the application. This can be useful

```

Find_win_path( $x(t)$ ,  $root$ )
-- $x(t)$ : input at time  $t$ ;  $root$ : root neuron of SOTT;
begin
  initialize  $\kappa$ ; -- initialize path search factor
  for ( $i = 1$  to  $L$ ) initialize  $\lambda_i$ ;
    --initialize the level of emphasis at each layer
   $v = root$ ;  $e_v = \|x(t) - w_v\|$ ;
    -- compute error at root;
   $V_1 = (v, e_v)$ ;
  ( $v, e_v$ ) = Find_win_child( $x(t)$ , 1,  $V_1$ );
    -- find winning children recursively until a
    winning leaf is found
  win_leaf =  $v$ ;
    --assign the win_leaf as the neuron-error pair
     $v$  returned by Find_win_child
  win_path = Trace_parent( $v$ );
    -- trace the winning path from the winning
    leaf to the root
end

Find_win_child( $x(t)$ ,  $i$ ,  $V_i$ )
--  $x(t)$ : input at time  $t$ ;  $i$ : layer number;  $V_i$ : Set of
neuron-error pairs, ( $v, e_v$ ) at  $i$ -th layer;
begin
  if ( $i = L$ ) return first element of  $V_i$ ;
    -- return winning leaf neuron
  Initialize  $V_{i+1} = \phi$ ;
    -- begin with an empty set of winning neurons
  for (each ( $v, e_v$ )  $\in V_i$ )
     $V_{i+1} = V_{i+1} \cup \text{Compute\_error}(x(t), i, v, e_v)$ ;
    -- compute errors for all children of node  $v$ 
  Delete all except  $\kappa$  neuron-error pairs with
  smallest error values in  $V_{i+1}$ .
  return Find_win_child( $x(t)$ ,  $i+1$ ,  $V_{i+1}$ );
end

Compute_error( $x(t)$ ,  $i$ ,  $v, e_v$ )
--  $x(t)$ : input at time  $t$ ;  $i$ : layer number;  $v$ : neuron;
 $e_v$ : error of  $v$ ;
begin
  Compute  $e_{u_j} = \|x(t) - w_{u_j}(t)\| + \lambda_i e_v \quad \forall u_j \in \text{child}(v)$ 
  return  $\bigcup_{j=1}^B (u_j, e_{u_j})$ ;
end

```

Figure 4.3: Algorithm for the best matching leaf neuron and winning path search

in some applications like progressive compression and transmission. It has been demonstrated by simulation in Section 4.5 that a small sacrifice in performance of the leaf layer can enhance the performance of the intermediate layers.

Another key parameter, κ is introduced in the subroutine *Find_win_child* to produce an equivalent effect of a κ -path search algorithm. *Find_win_child* is a recursive routine that calls *Compute_error* to calculate the errors of the children of a set of input neuron-error pairs. The κ best matching neurons are used as input to search for the κ best matching successor neurons at the next layer until the leaf layer is reached. Searching for the best matching leaf neuron from among the children of a single best matching neuron at the upper-layer is suboptimal. This is because the weight vector of the final winning neuron at the leaf layer for a single path search is most probably not the closest to the input vector. The parameter κ provides the control necessary to minimize the bias caused by branching at the higher level of the hierarchy. The value of κ can be tuned to trade computational complexity for overall performance. From simulation results, SOTT performs much better with $\kappa = 2$ than with $\kappa = 1$. The performance of the SOTT with $\kappa = 2$ is almost the same as that of the full search method but its computation time is much shorter and it is only a little longer than that of $\kappa = 1$. If κ_i is made different for some layers, we have a more complicated but interesting dynamic multi-path search.

The recursive procedure *Find_win_child* is called at the main routine *Find_win_path* with the root neuron and its error equal to the Euclidean distance from the input vector. Once the winning leaf neuron is found, the function *Trace_parent* traverses the winning path bottom-up via the predecessor neurons of the winning leaf neuron until the root neuron is reached.

4.3.4 Updating of Winning Path Neurons and Their Neighborhood

Once the winning path, Γ is found, all neurons along the winning path and their neighbors are updated by the following equations:

$$\mathbf{w}_v(t+1) = \mathbf{w}_v(t) + \alpha(m)[\mathbf{x}(t) - \mathbf{w}_v(t)] \quad \forall v \in \Gamma \quad (4.3)$$

$$\mathbf{w}_u(t+1) = \mathbf{w}_u(t) + \alpha(m)h(u, v, m)[\mathbf{x}(t) - \mathbf{w}_u(t)] \quad \forall v \in \Gamma \text{ and } u \in N_v(m) \quad (4.4)$$

where v is the winning neuron, and $N_v(m)$ denotes the neighborhood of v at sweep number m . As in FS-SOM, a sweep is defined as a period during which the updating parameters remain constant. In our simulations for SOTT, the number of input data in one sweep is set to be 6 times the number of neurons on the leaf layer. On average, each neuron on the leaf layer will be updated 6 times as the winning neuron in a sweep. The learning rate of the winning neurons at the m -th sweep, $\alpha(m) = \alpha(0)k_1^m$, is a monotonic decreasing gain function of sweep time. The initial learning rate, $\alpha(0)$ is usually set to a value slightly less than 1. The learning rate for the neighborhood of the winning neuron is scaled by the taper function $h(u, v, m)$ as in FS-SOM. This neighborhood taper is usually defined as a time-decreasing Gaussian function of the spatial distance between the winning neuron and its neighbors, i.e.,

$$h(u, v, m) = e^{-\frac{\|u-v\|^2}{\sigma_i(m)^2}} \quad \forall v \in \Gamma \text{ and } u \in N_v(m) \quad (4.5)$$

where $\sigma_i(m) = \sigma_0 k_2^m$ is the neighborhood width at the i -th layer at sweep number m . It defines the boundary of the lateral interaction between the winning neuron v and other neurons on the i -th layer. The initial neighborhood width of the i -th layer, $\sigma_i(0)$ is usually set to a value between 1 and $\frac{N_i}{2}$ where N_i is the number

of neurons at the i -th layer. Both the learning rate, $\alpha(m)$ and the neighborhood width, $\sigma(m)$ decrease with m . The constant k_1 and k_2 are set to a value typically between 0.8 – 0.99.

The updating function at each layer is in essence, similar to the method proposed by Pei and Lo [116]. However, being a multi-layer system, the neurons are likely to be dragged far away from their parents in the learning process as mentioned by P. Koikkalainen and E. Oja [91]. If there is no provision to preserve the correlation between the parent and children neurons, the tree structure will be destroyed, making the search for best matching neurons in the winning path more difficult and the overall distortion enlarged.

Maintaining both the intra-layer relationship (the neighborhood relationship) and the inter-layer relationship (the parent-children relationship) correct is important for achieving good performance of the proposed SOTT. To keep the neurons of the same family close during the entire training process, the following updating rules are imposed:

1. The initialization of neighborhood widths, $\sigma_i(0)$ at adjacent layers is made to be proportional to the ratio of the numbers of neurons on those two layers:

$$\frac{\sigma_i(0)}{\sigma_{i-1}(0)} = \frac{N_i}{N_{i-1}} \quad (4.6)$$

2. The children neurons will only be updated if their parent neuron is also updated.
3. The neighborhood neurons will only be updated if it is sufficiently close to the winning neuron of their layer. This is to minimize unnecessary com-

putation with little or no effect as the neighborhood learning rate reduces exponentially with increasing $\sigma_i(m)$. A neighborhood neuron, u is considered to be close enough to the winning neuron, v_i at the i -th layer if

$$\frac{\|u - v_i\|}{\sigma_i(m)} \leq \delta \quad (4.7)$$

The value of δ is usually set to be between 1 and 10. In our simulation, we use $\delta = 5$.

The neighborhood neurons satisfying both Rules (ii) and (iii) will be updated with Equation (4.4).

4.3.5 Convergence Criterion

At the end of each sweep, the weight vectors of the leaf neurons in the current and the previous sweeps are compared. If their difference is less than a small threshold value, then the training is completed. The convergence criterion can be stated as:

$$\frac{\sum_{j=1}^{B^{L-1}} \|\mathbf{w}_{Lj}(m) - \mathbf{w}_{Lj}(m-1)\|}{B^{L-1}} \leq \epsilon \quad (4.8)$$

In our simulation, if the average square difference of the neuron weights, $\mathbf{w}_{Lj}$ at the leaf layer is less than 0.01, the training is terminated.

4.4 Hybrid Clustering Algorithm on SOTT

As mentioned in the introduction, the ability of SOM in clustering suffers from three major disadvantages, which are fixed number of neurons, single-layer structure, and favoritism to isotropic clusters. To unbridle the limitations, hybrid clustering scheme [15, 99, 148] based on SOTT and a novel linkage metric is applied for enhanced clustering performance. The main idea behind the hybrid clustering is to combine the efficiency of the partition clustering and the prowess of discrimination of agglomerative hierarchical clustering (AHC). The general concept of hybrid clustering is illustrated in Figure 4.4. To differentiate the clusters formed in different stages, the clusters formed by the partition clustering are called the atomic clusters since they are indiscernible in the later AHC stage, whereas the clusters merged during the AHC stage are called the interim clusters. At the outset of the AHC, the interim clusters C_i is the same as the atomic clusters, A_i . An iterative process is carried out to optimally merge the interim clusters greedily. At each step, two interim clusters C_i and C_j with the maximum linkage metric are sought for merging into a larger cluster. The process of maximum link search repeats until a predefined criterion is met.

As mentioned earlier, the linkage metric is the key factor for AHC stage. To merge clusters rather than individual points, the distance between individual points has to be generalized to the distance between clusters (sets of points). Such derived proximity measure is called a linkage metric. The type of linkage metric used influences the AHC algorithms and their clustering results and efficiency significantly, since it reflects the closeness and connectivity between clusters, and usually the computation of linkage metric is the major bottleneck of AHC. Traditionally, the

Figure 4.4: Overview of hybrid clustering algorithm

dissimilarity measure (usually, the Euclidean distance) is calculated for every pair of points with one point from each cluster. Then, a specific operation such as minimum (single link), average (average link), or maximum (complete link) [73, 74] is applied to the pair-wise dissimilarity measures. A majority of the existing effective linkage metrics suffers from the $O(N^2)$ time complexity [15, 74]. Besides, they must be calculated only after all patterns are labeled. In view of this, we introduce a new linkage metric for our proposed hybrid clustering algorithm on SOTT.

Before introducing the novel linkage metric, a metric $Bond(A_i, A_j)$ to assess the connectivity of two atomic clusters, A_i and A_j is defined as follows:

$$Bond(A_i, A_j) = \sum_{\forall p | \{w_1(p), w_2(p)\} = \{i, j\}} e^{1 - \frac{d_2(p)}{d_1(p)}} \quad (4.9)$$

where p represents a data point in the data set, $w_i(p)$ represents the neuron that has the i -th minimum distance to p , and $d_i(p)$ represents the distance between $w_i(p)$ and p . The data points that fulfill the criterion for the summation, i.e., $\forall p | \{w_1(p), w_2(p)\} = \{i, j\}$ are those that cause neurons i and j to become the winner and runner-up neurons, in permutable order.

The proposed linkage metric, $Link(C_i, C_j)$ is defined as follows:

$$Link(C_i, C_j) = \max_{S_k \subseteq C_i, S_l \subseteq C_j} \frac{\sum_{A_m \in S_l} \sum_{A_n \in S_k} Bond(A_m, A_n)}{|S_k \cup S_l|} \quad (4.10)$$

$$|S_i| = \sum_{A_j \in S_i} |A_j| \quad (4.11)$$

where C_i represents the i -th interim cluster, and $S_k \subseteq C_i$ is a sub-cluster of C_i .

In traditional effective linkage metric, such as single-link, complete-link, and average-link, the computation complexity for $Link(C_i, C_j)$ is $O(N_i \times N_j)$, where N_i and N_j represents the number of data points in the interim clusters C_i and C_j , respectively. This is because the computation involves all the distances between two data points in the interim clusters C_i and C_j . On the other hand, the proposed linkage metric only involves the $Bond(A_i, A_j)$ of the atomic clusters instead of all data points in the pair of interim clusters (C_i, C_j) . However, the calculation for $Link(C_i, C_j)$ for Equation (4.10) is still quite involved since the bonds of all possible combinations of the sub-clusters, S of each interim cluster must be calculated to seek for the maximum link. In our simulation, we use a greedy growing of the sub-clusters to find the local optimal S_m and S_n for calculating the value of $Link(C_i, C_j)$. So the computational complexity is $O((k_i + k_j)k_i k_j)$, while k_i and k_j represent the number of atomic clusters in the clusters C_i and C_j , respectively.

The most noticeable feature of SOTT hybrid clustering, which differentiates it from other hybrid clustering schemes and makes it attractive in a practical viewpoint, is the on-line clustering feature. With the on-line SOTT of the first stage, one data point is processed and discarded at a time. Only one data pass is needed to obtain the result, and the training process is resumable. In the second AHC stage, the linkage metric only need $|A_j|$ and $Bond(A_i, A_j)$ for each atomic cluster rather than all the data points in the atomic clusters, and the $|A_j|$ and $Bond(A_i, A_j)$ can

be updated on-line during SOTT learning. Thus, the AHC is very fast and can be carried at any preset interval of the SOTT. Furthermore, due to the on-line learning nature, the memory requirement of the proposed algorithm depends on the structure of SOTT rather than the size of the data set. These properties are very appealing for large database clustering [15].

4.5 Simulation Results

In this section, we measure the performance of the proposed SOTT in vector quantization of gray-level images and compare it to the performance of classic SOM, GLA [100] and GLA based balanced TSVQ [21, 27] algorithms. The ability of SOTT quantizer to preserve the topological property of the image is also studied in the 3D color maps with the corresponding color images. Additionally, its performance in multi-resolution mapping is investigated. The two SOTT specific parameters, namely the leakage ratio, λ and the path search factor, κ are tuned to evaluate their effect on the network's learning ability and its computational efficiency.

In our simulations, the training data is fed from the test images to the SOM and SOTT networks using the butterfly permutation order until the networks converge. The test images of size 512×512 are shown in Figure 4.5.

To accurately reflect how well the neural networks have learnt from the training data, the testing and training data are obtained from the same source, and the search method is kept the same in both the training and testing phases. For fair

Figure 4.5: 512×512 gray-level test images.

comparison, the convergence criteria of GLA and GLA-based balanced TSVQ are set to be the same as in the SOM and SOTT algorithms. The SOM used is essentially a one-layer SOTT with the same codebook size (i.e., the number of neurons is the same as the number of leaf neurons of SOTT), and the common parameters of the two networks under test are set to the same values. The parameters for these two networks are listed in Table 4.1. Among these parameters, κ , λ and δ are used to tune the SOTT and they are not used on the original SOM.

Table 4.1: Test parameters of SOM and SOTT

Parameters	κ	λ	δ	$\alpha(0)$	K_1	K_2
Value	2	0.2	5	0.99	0.95	0.85

The quality of the reconstructed image is measured using the peak signal-to-noise ratio (PSNR) in dB. The PSNR is defined as follows:

$$PSNR = 10 \times \log \frac{255^2}{E[\|\hat{\mathbf{x}} - \mathbf{x}\|^2]} \quad (4.12)$$

where $\mathbf{x}$ and $\hat{\mathbf{x}}$ represent the original input vector and the quantized vector, respectively. $E[x]$ denotes the expected value of x .

The performance results of the two networks in terms of the PSNR of the reconstructed image of Lena, and the training time for different codebook sizes are tabulated in Table 4.2. The training time for both networks is measured on a Personal Computer equipped with a Pentium Pro. 2.1 GHz Processor and 512 MBytes of system memory. 1D string neural topology with branching factor, $B = 4$, is used for the SOTT. The first layer of SOTT is either 1 or 2 depending on the

codebook size.

Table 4.2: PSNR (dB) and training time (s) with different codebook sizes for Lena image

Codebook Size	SOM		SOTT		GLA		TSVQ	
	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
4	24.95	0.08	24.95	0.08	25.00	2.5	25.00	2.5
8	28.48	0.13	28.49	0.18	28.69	10.5	28.16	4.6
16	30.60	0.27	30.25	0.31	30.58	44.3	30.48	5.3
32	32.38	0.86	32.23	0.76	32.40	59.1	32.22	11.8
64	33.91	2.53	33.70	1.62	33.85	124.5	33.58	13.3
128	35.40	9.44	35.22	3.89	35.35	184.9	34.92	12.9
256	36.94	35.80	36.55	9.00	36.80	555.2	36.22	16.3
512	38.39	147.20	37.96	22.01	38.14	1062.6	37.72	19.9
1024	39.95	604.13	39.45	64.73	39.50	1745.5	39.06	19.0

The best reconstruction quality and the fastest training time are highlighted in bold print in Table 4.2. As expected, when the number of neurons is small (< 32), there is a small penalty in the training time of SOTT due to its structural overhead. As the network size increases, the SOTT speeds up dramatically. The acceleration over the SOM is more than 10 times for large codebook of size greater than 512. The quality of reconstructed images, based on the codebooks generated by both networks, are comparable.

To better assess the differences in reconstruction performances, different images

with varying degrees of textures and diverse spatial distribution of gray-levels are tested on both networks with the same codebook size of 256. The resulting PSNR of the reconstructed images and training time are shown in Table 4.3. Out of the

Table 4.3: PSNR (dB) and training time (s) of SOM and SOTT for different images

Image	SOM		SOTT		GLA		TSVQ	
	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
Lena	36.94	35.80	36.55	9.00	36.80	555.2	36.22	16.3
Bird	42.33	39.22	41.60	8.73	40.02	10.5	28.16	4.6
Bridge	31.54	43.95	31.19	9.18	31.33	401.8	30.75	18.5
Goldhill	35.46	42.88	34.87	9.28	35.18	355.8	34.22	12.9
Baboon	28.83	40.83	27.02	10.70	28.33	492.2	26.78	16.1

five test images, the qualities of the reconstructed images of Lena, Bird, Bridge, and Goldhill for the SOTT degrade by about 0.5 dB compared with those of the full-search SOM. The deterioration of PSNR of Baboon for SOTT is the worst (more than 1.5 dB). This is attributable to the high frequency of intensity and texture variations of Baboon.

To better visualize the topological property of SOTT, the color images of Lena is used. The red, green and blue component values of every pixel in a color image (Lena), is fed into the SOTT for training. After the network has converged, the weights of the neurons at each layer are shown in the RGB feature space in Figure 4.6. The topological relationship between neighboring neurons is visualized by connecting lines of different styles to distinguish the different layers. Figure 4.7

Figure 4.6: 1D string color map of a 3-layer SOTT.

shows the visual effect of the topological structure of the SOM.

It can be seen from Figure 4.6 that every layer in the SOTT can be considered as a single SOM, which has its own topology comparable to a single SOM. The tightly coupled parent-children topographical relationship is evident from Figure 4.6.

In the 1D string neighborhood structure, the topology order can also be measured by an objective function J , which is defined in [88] as follows:

$$J = \sum_{i=2}^N \|\mathbf{w}_i - \mathbf{w}_{i-1}\| - \|\mathbf{w}_N - \mathbf{w}_1\| \quad (4.13)$$

where N is the number of neurons and $\mathbf{w}_i$ is the weight vector of the i -th neuron. The lower the value of J , the better the codebook is ordered.

In our simulations, we use this function to compare the topology order of the SOM

Figure 4.7: 1D string color map of SOM.

and the leaf layer of SOTT. The result is plotted in Figure 4.8. It can be concluded that their topology orders are comparable.

The multi-resolution progressive decoding performance of the SOTT is evaluated with a network of branching factor, $B = 4$ and depth, $L = 5$. The leaf layer has 256 neurons. The images reconstructed from different layers of the SOTT are shown in Figure 4.9.

Different leakage ratios, λ are experimented on Lena and the quantization results of different layers of the SOTT are presented in Table 4.4. The multi-resolution performance of TSVQ is also listed for comparison. For ease of reference, PSNR of Lena trained by the SOM for the corresponding codebook sizes from Table 4.1 are reproduced in the last column of Table 4.4.

In Table 4.4, the best PSNR for each codebook size are printed in bold. The

Figure 4.8: Topology order J of SOM and SOTT

Figure 4.9: Lena images reconstructed from different layers of SOTT

Table 4.4: PSNR (dB) of Lena image reconstructed with different layers and leakage ratios

Layer	Size	SOTT						TSVQ	SOM
		$\lambda = 0$	$\lambda = 0.05$	$\lambda = 0.10$	$\lambda = 0.15$	$\lambda = 0.20$	$\lambda = 0.25$		
2	4	23.59	24.46	24.56	24.70	24.75	24.79	25.00	24.95
3	16	30.32	30.26	30.40	30.36	30.49	30.50	30.65	30.60
4	64	33.37	33.49	33.52	33.61	33.66	33.62	33.51	33.91
5	256	36.41	36.46	36.48	36.58	36.55	36.45	36.09	36.94

overall quality of the image reconstructed from the codebooks obtained at different layers of the SOTT for a fixed λ is comparable to that obtained from several independently trained SOMs of equivalent codebook sizes. It should be noted that codebooks of different resolutions are generated simultaneously with SOTT through only one pass of training with non-exhaustive search of the best matching neurons as opposed to training $L - 1$ SOMs independently with the full search algorithm. The leakage ratio improves the performance of lower resolution codebook at the expense of a trivial degradation to the performance of the higher resolution codebook.

Different values of the multi-path search factor, κ of the SOTT are also tested to evaluate how the winning path search factor influences the performance of the SOTT. The resulting PSNR and training time are shown in Table 4.5. $\kappa = 1$ is the same as the single path tree-structured search algorithm. Comparing with $\kappa = 1$, $\kappa = 2$ produces much better performance with a slight increase in training time.

Table 4.5: PSNR (dB) and training time of Lena using SOTT with different values of κ

κ	PSNR (dB)	Time(s)
1	35.11	7.07
2	36.55	8.68
3	36.64	10.40
4	36.49	12.50
5	36.56	13.16
6	36.67	14.31
7	36.45	16.97
8	36.54	17.55
9	36.44	18.15
10	36.63	19.42

$\kappa = 3$ gives the optimal performance. There is little or no advantage in increasing the path search factor beyond $\kappa = 3$.

There are occasions when the performance of the SOTT becomes a little worse as κ increases (while maintaining the condition that the encoding κ is still higher than the training κ). This is not surprising as the performances of SOM and SOTT reach near to convergence, they become highly sensitive to the variation of parameters and input data distribution. Although the intuitive interpretation of the network behavior relative to its parameter variations is clear, there is no rigorous mathematical analysis of the dynamics of the self-organizing algorithms and their proof remains yet to be discovered. Tuning of these parameters is highly empirical and is best optimized with a specific application through a long series of simulations. Hence, once the network has been tuned for a minimal acceptable level of performance (quality) for a domain-specific application, computational complexity, resource requirements, implementation simplicity and other features will be the prevailing factors. In this perspective, SOTT has a clear competitive advantage over the SOM and GLA algorithms owing to its fast learning, fast encoding and multi-resolution mappings of progressively enhanced qualities. When comparing with GLA and TSVQ, SOTT has the advantages of the graceful topological ordering feature and on-line learning capability.

Several datasets are used to simulate the clustering performance and computation efficiency of the proposed SOTT based hybrid clustering algorithm to compare them with those achieved by the antecedent clustering algorithms. Figure 4.10 (a)-(d) shows four different 2D data sets used in the simulation. These data sets are chosen, partly because similar data sets have been used extensively by other researchers [59, 79, 99], and partly because it is easier to evaluate the quality of

the clustering results on 2D data sets. Due to the different geometrical shapes and densities of the clusters, and the scattering noise in these data sets, traditional algorithms like the k -mean and AHC will fail to find the proper clusters as proven in [59, 79]. CURE has been tested to prove that it failed in clustering the data sets, Figure 4.10 (b) and (c) in [79], as seen in Figure 4.10 (e) and (f). Spectral clustering, which has proven to be efficient in some challenging clustering problem [109], unfortunately failed in three of the four test data sets, as seen in Figure 4.10 (g), (h), and (i).

Figure 4.11 shows the clusters found by our proposed SOTT based hybrid clustering algorithm for each of the above data sets. For data sets (a), (c) and (d) of Figure 4.10, our SOTT hybrid clustering algorithm produces the natural clustering results as perceived by the human eyes. For data set (b), only the vertically elongated textual patterns next to the horizontally aligned objects are imperfectly clustered into two different clusters instead of one.

The overall computation time of our proposed SOTT hybrid clustering algorithm can be divided into two parts: one on the learning and partition clustering time of the SOTT and the other is the time it used for AHC. In the first stage of the SOTT partition clustering, it can be shown that for n patterns and k atomic clusters, the computational complexity of SOTT learning stage is mainly dependent on the SOTT structure and learning parameters rather than on the size of the data set, while the computational complexity of mapping all the patterns in the data set to the atomic clusters is $O(n \log k)$ due to the tree-structured search. In the second stage of AHC, the computational complexity of $Link(C_i, C_j)$ for all clusters dictates the computational complexity of the stage and it is $O(k^3)$. Usually n is much larger than k , thus the computational complexity in the second stage is

Figure 4.10: (a)-(d) data sets used in the clustering performance evaluation; (e)-(f), failures of clustering (b)-(c) by CURE; (g)-(i) failures of clustering (b)-(d) by spectral clustering.

Figure 4.11: Clustering results of different data sets given by our proposed algorithm

Figure 4.12: Computation time versus size of the database

negligible comparing with that in the first stage. This is in congruence with the observation of our simulation. Therefore, it can be concluded that the proposed SOTT hybrid clustering algorithm is scalable to the data set size. The scalability of our proposed algorithm is manifested in Figure 4.12. In Figure 4.12, it can be seen that the proposed SOTT hybrid clustering algorithm is computationally much more efficient than the recently proposed CURE algorithm [59], especially for large data sets.

Though the use of hybrid clustering algorithm can unbridle the second and third limitations mentioned in the introduction, as observed from the clustering results of Figure 4.13, the number of atomic clusters will still influence the performance of the hybrid clustering algorithm despite its effect has been reduced. However, the number of atomic clusters and the clustering results of the proposed SOTT hybrid clustering algorithm are not as vulnerable to the setting of the learning

(a) $k = 64$ (b) $k = 16$

Figure 4.13: Clustering results for different number of initial clusters

parameters of the SOTT as to the growing variants of SOM. Usually it is difficult to predict the number of atomic neurons in advance. However, the different resolutions created by the SOTT structure provide a flexibility to alleviate the problem associated with a fixed or an inappropriate choice of the number of atomic clusters. With the ability to maintain a cohesive parent-children and sibling relationships in SOTT during the partition clustering stage, the AHC can be applied on different layers which have different number of neurons (i.e., partitions). A clustering validation [60] process can then be added to decide which level of clustering and merging is preferred for the given data.

4.6 Chapter Summary

A novel on-line Self-organizing Topological Tree (SOTT) is proposed in this chapter in which a multi-path search is used to overcome the bias of the tree-structured search. A parameter named leakage ratio is introduced to control the performance tradeoff between different neuronal layers. During the learning phase, specific updating rule is designed to maintain good inter-layer and intra-layer relationships.

The hierarchical structure of the proposed SOTT is also exploited in a hybrid clustering algorithm to overcome the clustering performance deficiencies of classic k -means and SOM algorithms. To fully exploit the advantages of online learning and reduced computational complexity of the SOTT architecture, a new linkage metric is introduced for the AHC. This linkage metric can be updated online during the partition clustering stage and does not need to wait for the patterns to be labeled. The proposed SOTT hybrid clustering algorithm has demonstrated to be computational efficient and possesses good scalability. Besides, its clustering performance is more robust against learning parameter variation compared with the growing variants of SOM.

The experimental results show that, the performance of the proposed SOTT is comparable to the full-search SOM. At the same time, the computation efficiency of SOTT is much better than that of SOM and other vector quantizers. The progressive encoding property offered by SOTT is attractive for developing new efficient and elegant solutions to data clustering and pattern classification applications.

Chapter 5

Color Halftoning and Skin Color Classification

In the last two chapters, we have presented two new unsupervised self-organizing algorithms founded on different topological structures. Their performances have been evaluated in color quantization, vector quantization and data clustering area. A global butterfly jumping sequence (GBJS) featuring progressive resolution enhancement for online learning has been adopted in both algorithms.

In this chapter, we will further explore their potential in two interesting application areas. In the first part, we will extend the use of the output of self-organizing color quantization algorithm to high quality color rendering with limited color palettes for low-cost displays and printers. We will introduce a new color halftoning technique based on GBJS as a post-processing step after the color quantization

using self-organizing algorithms. In the second part, we will study the use of SOTT to pixel based skin color classification problem. Unlike clustering discussed in the previous chapter, which falls in the unsupervised learning paradigm, classification is a supervised learning problem. For this exploration, we devise two training and mapping schemes on SOTT and evaluate their effectiveness and efficiency as pixel based skin color classifier. The work of this part has been revised from our paper presented and published in the Proceedings of the *2004 IEEE Asia-Pacific Conference on Circuits and Systems* [35].

5.1 Color Halftoning

5.1.1 Introduction

True color images typically contain up to 16 million different colors. One of the basic tasks of low-level image processing consists of reducing the number of colors with minimal visual distortion, since many image display and printing devices provide only a limited number of colors. The representation problem for colors is aggravated when many images are displayed simultaneously, resulting in a palette size of 256 or even scantier number of colors being assigned to each image. From computation efficiency viewpoint, a coarse resolution of colors is important for fast image manipulation and analysis.

Digital image processing methods for coding, transmission and computer vision applications usually model or exploit the spectral and spatial acuity of human

visual system. It is therefore desirable that the color rendering of continuous-tone 24-bit digital color images on devices with limited color palettes generate a high-quality output with minimal perceptual distortion. Several techniques have been proposed for the color reduction problem for image analysis and synthesis, most of which are carried out in two steps: the color palette design and color halftoning [14].

The first step involves the selection of a finite number of colorant vectors $m_i, i = 1, 2, \dots, k$ to approximate the continuous probability density function $p(x)$ of the variable x in the color space. Each pixel in the continuous-tone color image is then mapped to a colorant vector in the optimal color palette that is closest to it. This process of color reduction, which attempts to stochastically minimize the distortion usually in terms of the mean square error (MSE), is widely known as color quantization. Examples are the popular median-cut quantization [65], octree quantization [52] and the application of a variety of clustering methods like LBG [18], agglomerative clustering [156], sequential scalar quantization [11, 12], and self-organizing networks [34]. The design of color palette based on FS-SOM and SOTT have been evaluated in Chapter 3 and 4, respectively. In fact, any suitable clustering approach could be used (see [61, 73, 74] for an overview). It is a characteristic of all clustering methods that spatial and contextual information are neglected. While it may be judicial to base on statistical distortion measure for color reduction oriented towards computation efficiency, the perceptual quality of such color rendering measures are inadequate for display and printing applications as the frequency sensitivity of human vision system and its response to color noise have not been considered. Several types of visual artifacts can appear in the quantized image due to the limited number of colors. The most severe and commonly encountered ones being the false contouring artifacts in regions with

smooth color transitions.

The perceptual quality of the color quantized image can be substantially improved by following it up with a digital halftoning [145] step. Digital halftoning shapes the quantization noise power into the high frequency regions where the human visual system is least sensitive. Human eyes perceive high frequency spatial variations in color as a homogeneous color, which can be understood as an averaging superposition or low-pass characteristic of the contrast sensitivity of the human visual system. Therefore, additional perceived colors can be created by spatial mixing. A commonly used halftoning technique is called error diffusion where the quantization error is spread to neighborhood pixels. In essence, it biases the distortion at neighborhood pixels in opposite directions. Several error diffusion filters have been proposed [41, 144, 93]. In addition, numerous dithering [111, 139] and model-based halftoning [139, 92, 113] techniques have been developed to provide high-quality color rendering. In this chapter, we will focus on the study of error diffusion method that can blend well with color quantization method based on the previously discussed self-organizing topological maps.

In general, color error diffusion can be described by the following procedure:

1. For each pixel of the image, find the closest colorant vector from the given color palette.
2. Calculate the difference between the values of the pixel color and the selected color, which is the quantization error.
3. The error value is divided and distributed over the neighborhood pixels which

Figure 5.1: Block diagram of error diffusion

have not been visited yet.

- 4. When a pixel is visited, the errors diffused from its neighborhood pixels are accumulated and added on to its value, The values will be clamped to a permissable range if necessary.
- 5. Repeat the above process until all pixels have been traversed.

Figure 5.1 shows the system block diagram for color error diffusion halftoning. In Figure 5.1, each signal is vector valued, e.g., a vector of RGB values. The quantization error $e(m)$ is fed back, filtered by the error filter $\tilde{h}(m)$, and added to the input continuous-tone image $x[m]$. The net effect is to diffuse the quantization error frequencies and colors.

Figure 5.2: Floyd-Steinberg error filter.

The error filter $\tilde{h}(m)$ has matrix-valued coefficients. To avoid ambiguity we use the overhead “ $\sim$ ” to distinguish a matrix from a vector. The matrix-valued sequence $\tilde{h}(m)$ is defined on a causal support set S with $(0, 0) \notin S$. If the vector color error-diffusion halftoning system is applied to each color separately, $\tilde{h}(m)$ would be a diagonal matrix. A well-known error diffusion error filter is the Floyd-Steinberg error filter, which is shown in Figure 5.2.

In the general color error diffusion procedure portrayed above, we did not specify the neighborhood size and how error are apportioned among the neighborhood pixels. There are many error diffusion techniques, differ mainly in the values of the coefficients of error filter $\tilde{h}(m)$. The order of traversing the pixels, i.e., the sequencing of $x(m)$, has also a subtle effect on color halftoning but often not evaluated or emphasized. Very often, the analysis and modeling of error diffusion

methods assume the use of raster scan or serpentine scan [85, 82, 147]. In line with the color palettes designed with the FS-SOM and SOTT proposed in Chapters 3 and 4, we introduce the global butterfly jumping sequence (GBJS) into the color halftoning step to extend the use of self-organizing topological maps for low-level image processing with limited colors. In the following sections, we will outline the features of progressive resolution enhanced color halftoning. We will also study the perceptual quality of images reconstructed from GBJS-based error diffusion (GBJS-ED) in conjunction with the output levels produced by the SOM color quantizer. We use the input 24-bit RGB color space for display applications to enable us to concentrate on the exposition of essential color quantization and error-diffusion halftoning without having to focus on the issues of device dependent nonlinear color transformation. The approach and experimentation can be extended to other multi-level images, for example the quadruple of cyan, magenta, yellow, and black (CMYK) for printing applications.

5.1.2 Proposed Error Diffusion Algorithm Based On GBJS

Firstly, we need to define a spatial kernel, w_{ijk} to model the human vision system. Here we use a simple Gaussian kernel of identical standard derivation, σ for all color channels. The weights, w_{ijk} for the neighborhood pixels are defined identically for all k as

$$w_{ij1} = w_{ij2} = w_{ij3} \propto \exp\left(-\frac{D_{ij}}{\sigma^2}\right), \quad \sum_{j \in N_i} w_{ijk} = 1 \quad (5.1)$$

where D_{ij} is the Manhattan distance between pixel i and j on the 2D image grid. N_i is the spatial support for the kernel. In our implementation, we set $j \in N_i$ iff $D_{ij} \leq 3$. A Gaussian kernel is adopted for the purpose of simulation. σ is

the standard deviation of the Gaussian kernel and it is set to be 1. $k = \{1, 2, 3\}$ accounts for the three different color channels. In our simulation, the kernels are set to be identical for different color channels for simplicity.

Once the color palette and the spatial kernel have been chosen, the value of the i -th pixel will be quantized to the codevector, $m_{c(i)}$ and the index to the color palette $c(i)$ is selected based on the distance of the continuous-tone pixel value, x_i to the codevectors.

$$c(i) = \arg \min_k \|y_i - m_k\| \quad (5.2)$$

$$\hat{x}_i = m_{c(i)} \quad (5.3)$$

$$y_i = x_i + \sum_{j \in N_i} e_j \circ w_{ij} \quad (5.4)$$

where x_i is the original continuous-tone pixel value at pixel i , $c(i)$ is the index of quantized image at pixel i , y_i is the value at pixel i which have been compensated by the neighborhood quantization error, m_k is the k -th codevector. $\hat{x}_j$ is the quantized value at pixel j , e_j equals to $x_j - \hat{x}_j$ if pixel j has been quantized, and 0 otherwise. $\circ$ denotes the element-wise multiplication. In this proposed mapping method, every pixel will compensate for the distortion induced by its neighborhood pixels, in order to obtain a better image quality perceptually.

The idea of this halftoning scheme is to refine the perceived image resolution progressively. When the errors of the neighborhood pixels are unknown, early mapped pixels are mapped as in the traditional fashion. Thereafter, the pixels are mapped with the aim to compensate for the errors incurred by the previously mapped neighborhood pixels.

Traditional error filters are usually obtained empirically as opposed to deriving

from human vision system model. Consequently, the quantization error can only be diffused into the neighborhood pixels in some specific directions, as shown in Figure 5.2. This directional compensation is incongruous with the human vision system. Furthermore, it tends to accumulate errors and produces color spike, worms, and other objectionable artifacts. With GBS, the quantization errors are scattered omnidirectionally into the neighborhood pixels. Thus it automatically leverages on the human vision systems models to guide the halftoning process. In the next section, the performances of the proposed error diffusion method with global butterfly jumping sequence will be compared against that obtained with standard raster sequence error diffusion.

5.1.3 Simulation Results and Discussions

In this simulation, we compare the perceptual quality of our proposed color quantization and halftoning algorithm against two other algorithms. One is the traditional SOM by Floyd-Steinberg (FS) [41] error diffusion algorithm. The other is Puzicha's Spatial Color Quantization algorithms [120]. In [120], an optimization problem of codebook design and codevector assignment are solved by a computation-intensive simulated annealing technique. In addition, the palletized images obtained by using FS-SOM or SOTT without halftoning are used as reference for comparison. One of the test images, "Pool", as shown in Figure 5.3 is used to illustrate the perceptual quality of the halftoning techniques being investigated.

From Figure 5.4(a), it can be seen that the traditional color quantization without halftoning fails to produce the smooth surface of the balls and the background.

Figure 5.3: Test color image: Pool

Many false edges are produced due to the reduction of the number of colors. From Figure 5.4(b), false edges are reduced by FS error diffusion. However, there are many color impulses in the images, which are very annoying. From Figure 5.4(c), we can see that spatial color quantization is able to reproduce the smooth surface of the balls. There are no color impulses and false edges in the rendered images. The perceptual quality is much better than those produced by the previous two methods. The visually appealing rendered images of our method are shown in Figure 5.4(d). No visible false edges and color impulses are detected in these palletized images. The background and surface of the balls are even smoother than those observed in Figure 5.4(c).

As mentioned earlier, Puzicha's Spatial Color Quantization is a very computational intensive algorithm. The quality of the palletized image suffers somewhat

Figure 5.4: Comparison of different color reduction algorithms (from left to right: 8, 16, and 64 colors): (a) color quantization with FS-SOM, (b) SOM + FS, (c) spatial color quantization, and (d) proposed method with global butterfly jumping sequence.

if multiscale optimization is used to lower the computation cost. Even with the multiscale optimization, the computational complexity can never be lower than that of the traditional color quantization method followed by error diffusion [120]. Compared with Spatial Color Quantization, the proposed method is very simple and has a much lower computational complexity. The computational complexity of the multiplication operations in Equation (5.4) is about three times higher than that of the FS error diffusion. However, the most time consuming task in pixel mapping is to search for the best matching codevector, which is dictated by Equation (5.2). Therefore, the overall computational complexity based on Equations (5.2) to (5.4) is nearly the same as the traditional color quantization methods followed by error diffusion. In a 64 color-palette mapping, the proposed color rendering method with FS-SOM needs only about 7% more computation time than the traditional method. As mentioned in the previous chapter, if SOTT is used, the search time for the best matching codevector can be reduced significantly.

The rendered images obtained by our proposed error diffusion method with the raster sequence and the global butterfly jumping sequence are also compared in Figure 5.5. As expected, the perceptual quality of Figure 5.5(b) under the global butterfly jumping sequence is much better than that of Figure 5.5(a) under the raster sequence. False edges can still be found in the surfaces of Figure 5.5(a), and the color of the cue is also not as natural and smooth as that appeared in Figure 5.5(b).

It is well known that human perception possesses a low-pass filter response. Human eye is more sensitive to low frequency distortion. For this reason, we analyze the distortion of the palletized image due to color quantization in frequency domain. First, an error image is calculated as the difference between the original

Figure 5.5: 16-color rendered image by the proposed method under (a) raster sequence (b) global butterfly jumping sequence

image and the palletized image. Then, a 2D-FFT is performed to display the error distribution in the frequency domain. Figure 5.6(a) is the error distribution due to SOM color quantization only. Figure 5.6(b) is the error distribution due to SOM color quantization followed by the proposed error diffusion halftoning method. It is evident that the proposed method reduces the low frequency distortion significantly.

Unfortunately, it is hard to find an objective measurement to quantify the perceptual quality of the color rendered image. Table 5.1 lists several objective measurements on the rendered images obtained from different algorithms. Mean Square Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) are poor measurements for color halftoning because it does not account for the frequency acuity of human vision. For example, in Table 5.1, although the MSE of the palletized image from CQ alone is the lowest, it has the worst perceptual quality. Weighted Mean Square Er-

Figure 5.6: Distortion visualization in frequency domain for the luminance of “Pepper” image (a) SOM CQ only (b) proposed halftoning method.

ror (WMSE) is designed to mitigate the inferiority of MSE and this metric exhibits better correlation to the actual perceivable quality of image halftoning [104, 105].

However, Venkata et al. [147] pointed out that WMSE is still not good enough. The error image between the original and distorted images is somehow correlated with the original image [83]. Knox also showed that the sharpness of halftones increases when the correlation between the error image and the input image increases. In [147], it is demonstrated that the error image has two attributes: one that is correlated with the image and one that pertains to image uncorrelated noise. These two attributes have different influences on the perceptual quality of the distorted image, such that no any single objective measurement can segregate them easily. In Table 5.1, C_{RI} is used to quantify the degree to which an error image R is correlated with an original image I . The magnitude of the following

Table 5.1: Objective measurements of different algorithms

Algorithms	C_{RI}	MSE	WMSE
SOM CQ	0.175	6.62	67.16
SOM CQ+FS-ED	0.080	11.59	64.23
SCQ	0.070	13.17	51.01
Proposed Method	0.143	8.44	61.53

correlation between them is used [152]:

$$C_{RI} = \frac{|Cov[R, I]|}{\sigma_R \sigma_I} \tag{5.5}$$

where Cov refers to the covariance, σ_R and σ_I are the standard derivations of R and I , respectively. As MSE and WMSE are not good reflections of the perceptual color distortion, the variation of the correlation metric with different algorithms cannot be inferred from the MSE or WMSE values in Table 5.1.

In [147], through mathematical analysis, a model adaptive to their proposed method is first developed to extract the uncorrelated noise of the distorted image due to error diffusion. The distortion of this uncorrelated noise can then be minimized. Similarly, a complex degradation model is recommended in [146]. They developed two measures of degradation distortion measure (DM) and noise quality measure (NQM) [146], based on prudent modeling of the distortion allude to error diffusion [82, 147]. Their approach is based on the ability to separate the image-correlated and image-uncorrelated distortions so that they can be measured independently. This provident method of measurement show stronger congruence with subjective evaluation results. However, it is difficult to use this objective mea-

sure to benchmark the perceptual quality of all color quantization and halftoning algorithms as it relies heavily on an accurate modeling of the distortion model used and may not be feasible as in our case.

Modeling of perceptual quality of digitally processed color images is an emerging field of research which requires onerous effort in building an adequately large and diverse database for verifying the accuracy of the model. At the juncture, the lack of a suitable model for objective evaluation of the quality of color rendered images produced by our method has been acknowledged as a major setback to convincingly prove the effectiveness of our proposed approach. To better understand and benchmark the performance of the proposed color halftoning method against others, an appropriate mathematical model of the distortion induced by the proposed method has to be established, either through rigorous characterization of the error image in the frequency domain or by complex mathematical analysis. The developed model can also be used to design the optimal error diffusion of coefficients and further improve the algorithm.

5.2 Skin Classification

5.2.1 Introduction to Skin Classification

Skin classification aims to identify human skin regions in digital images and video. It has been proved to be an efficient pre-processing step for many computer vision tasks, including face detection, hand gesture analysis, and content filtering of

objectionable images [46, 67, 154, 165]. Reducing the search space for the object of interest through skin color detection in those applications is motivated by the pre-attentive mechanism commonly exhibited in human vision systems. Detection of skin colors help to eliminate irrelevant background information and enable subsequent processing to focus on the interested skin region. Skin detection can usually be done regardless of the variations in pose and scale.

Among numerous existing methods for skin detection [161, 119], the most notable methods are fixed range decision boundary [23, 136, 141, 149], Gaussian models [103], Bayesian classifier [22, 76, 97, 137], multi-layer perceptron (MLP) [126] and self organizing maps (SOM) [19]. In fixed range decision boundary methods, skin and non-skin colors are separated by explicitly defined rectangular boundaries. Chai and Ngan [23] considered skin color range in YCbCr space as $Cb \in [77, 127]$ and $Cr \in [133, 173]$ while Garcia and Tziritas [46] proposed a decision boundary consisting of eight planes in the YCbCr color space. Other fixed range classifiers were also proposed by Solina et al. [136], Takaya and Choi [141], and Wang and Yuan [149]. These methods are simple and fast, but they lack the robustness to achieve good classification because it is difficult to find a suitable color space and a good decision boundary empirically. Gaussian models are used to approximate the skin color distribution, as a single Gaussian or a mixture of Gaussians [118]. Gaussian models are compact classifiers in which the decision boundary is determined by a small set of parameters (i.e., sample means and covariance matrices) that can be learnt from a limited training set. However, the fitness of Gaussian models for skin detection depends heavily on the color space used. For single Gaussian models, this dependency is more pronounced, and for Gaussian mixture models, training is usually computation-intensive. Bayesian classifier is based on Bayesian decision rule, which is a well-established technique in statistical pattern classification [36].

It has been used by a number of authors for skin detection [24, 22, 76, 97, 137]. They usually require the construction of a skin probability map instead of deriving an explicit model of the skin color as in the parametric methods. The performance of Bayesian classifier depends largely on the choice of the training data set. When the training set is large enough these methods are superior to parametric methods in terms of the accuracy and computational cost [22]. Recently, Brown et al. [19] proposed two SOM-based methods for classifying skin and non-skin colors: one uses only skin samples, and the other uses both skin and non-skin samples. They found that the latter method performs better compared to the former method. Although these SOM-based methods were found to have only moderate performance [119], they shown how SOM can be used as a clustering and classification tool for color image segmentation.

In the following subsections, we will propose two skin classification schemes based on SOTT, and compare them with the above classifiers in the simulations.

5.2.2 Skin Classification Schemes Based On SOTT

(A) Semi-Adaptive Scheme

In this scheme, the training image set is separated from the testing set. We use the first 100 images of the ECU database [119] for SOTT training. A training image is input until the network converges before a new training image is fed. A manually segmented skin image set is also required during training to compute the statistics for the classification process. In other words, during the entire training process,

the number of the skin and non-skin pixels won by each neuron will be recorded in addition to the codebook. After the training phase, SOTT is ready to test any image based on the skin color probability of each neuron. The classification of the skin and non-skin neurons are similar to the classification of the skin and non-skin bins in Bayesian classifier. That is, if the ratio of skin pixels over non-skin pixels exceeds a threshold value, the neuron is classified as a skin neuron and all testing image pixels that are mapped to this neuron will be classified as skin pixels. The difference here is that SOTT does the self-organization (clustering) first before the classification process begins, and the number of training images used is much less than that of the Bayesian method. As can be seen in Figure 5.8, the performance of this scheme is poorer than SOTT-Bayesian Scheme and the Bayesian method, though it is far better than the SOM [19] and the single Gaussian model [103]. Nevertheless, it uses much less training data and memory resource than the Bayesian classifier.

(B) SOTT-Bayesian Scheme

The proposed scheme is named SOTT-Bayesian as it combines the clustering property of SOTT and the probabilistic feature of Bayesian. To see how these two techniques are combined, we will begin with the Bayesian formulation of skin classification problem described in [24].

Let $P(\omega_i | x)$ denotes the probability of the observation x belongs to class i . For the skin color detection application of Section 5.2, $P(\omega_s | x)$ denotes the probability that the observed pixel with color x is a skin pixel, and $P(\omega_{ns} | x)$ is the probability of the observed pixel with color x is not a skin pixel. Thus,

$P(\omega_s | x)$ and $P(\omega_{ns} | x)$ are the respective a posteriori probabilities for skin and non-skin color classes. Let C_{fp} be the cost of false positive, and C_{fn} be the cost of false negative, and the costs of both true positive and true negative, i.e. C_{tp} and C_{tn} , are set to zero.

Let $R_s(x)$ and $R_{ns}(x)$ be the conditional cost of classify a feature vector x as skin and non-skin classes respectively. They can be written as

$$R_s(x) = C_{fp}P(\omega_{ns} | x)$$

and

$$R_{ns}(x) = C_{fn}P(\omega_s | x)$$

Based on above two equations, the optimal classifier, which gives the minimum cost, can be represented by the following rule:

$$\begin{cases} \text{if } \frac{P(\omega_s|x)}{P(\omega_{ns}|x)} > \frac{C_{fp}}{C_{fn}} & \text{then } x \in \omega_s \\ \text{else} & x \in \omega_{ns} \end{cases} \quad (5.6)$$

$P(\omega_s | x)$ and $P(\omega_{ns} | x)$ can be written in Bayesian formulation as:

$$P(\omega_s | x) = \frac{p(x | \omega_s)P(\omega_s)}{p(x)}$$

$$P(\omega_{ns} | x) = \frac{p(x | \omega_{ns})P(\omega_{ns})}{p(x)}$$

then the optimum classifier can be represented by,

$$\begin{cases} \text{if } \frac{p(x|\omega_s)}{p(x|\omega_{ns})} > \frac{C_{fp}}{C_{fn}} \cdot \frac{P(\omega_{ns})}{P(\omega_s)} & \text{then } x \in \omega_s \\ \text{else} & x \in \omega_{ns} \end{cases} \quad (5.7)$$

In our method, $p(x | \omega_i)$ is approximated by $p(c(x) | \omega_i)$, where $c(x)$ denotes the neuron to which x mapped. $p(c | \omega_i)$ is approximated in the second step of our

SOTT-Bayesian method as the ratio of N_i^c (the number of skin/nonskin pixels in neuron c) and N_i (the total number of skin/nonskin pixels). Thus,

$$p(x | \omega_s) = p(c(x) | \omega_s) = \frac{N_s^{c(x)}}{N_s}$$

$$p(x | \omega_{ns}) = p(c(x) | \omega_{ns}) = \frac{N_{ns}^{c(x)}}{N_{ns}}$$

Taking these two equations collectively, the optimal classifier can be rewritten as

$$\begin{cases} \text{if } \frac{N_s^{c(x)}}{N_{ns}^{c(x)}} > \frac{C_{fp}}{C_{fn}} \cdot \frac{P(\omega_{ns})}{P(\omega_s)} \cdot \frac{N_s}{N_{ns}} & \text{then } x \in \omega_s \\ \text{else} & x \in \omega_{ns} \end{cases} \quad (5.8)$$

Because the right hand side of the inequality is constant for x , it can be treated as a threshold τ . τ can be adjusted to control trade CDR for FDR or vice versa.

In our implementation, the training phase of Bayesian-SOTT was divided into two phases. During the first phase SOTT is trained as in the Semi-Adaptive Scheme, but without any skin/non-skin statistics being recorded. In the second phase, a large number of training data is mapped to the neurons and the number of skin and non-skin pixels are accounted. This is done with the help of the manually segmented skin image set of the ECU database [119, 117]. 2,500 of them are used in this phase. It is worth noting that during the second phase, the neurons weights are static and will not be updated any more. While mapping these data the Bayesian method explained above is utilized to determine the skin and non-skin neurons. Lastly, testing is done by mapping the color vectors to the neurons to decide the skin/non-skin class of the pixel by the class of the winning neuron to which it is mapped.

5.2.3 Simulation Results

In this subsection, we will compare the skin detection performance of the following pixel color classifiers.

1. the fixed range classifier with the rectangular decision boundary in the 2D CbCr plane(see Figure 5.9), proposed by Chai and Nagan [23].
2. the Bayesian classifier using histogram-based nonparametric density estimation proposed in [24, 119, 117]. In this simulation, we use the Bayesian classifier in the 3D RGB color space with a histogram size n of 64 bins per channel.
3. the Bayesian classifier in the 2D CbCr color space with a histogram size of 32 bins per channel, a similar Bayesian classifier as above but with a reduced storage, is also tested.
4. the Gaussian models: the 2D Gaussian model in the CbCr plane, proposed by Menser and Wien [103]. We used the results generated by S. L. Phung in [119] for this method.
5. the SOM classifier proposed by Brown et al. [19]. The detailed implementation and results were also excerpted from those conducted by S. L. Phung [119].

The ECU database [119, 117] is used as the training and testing source for the experiments and performance benchmarking. This database consists of 4,000 color

images that have been manually segmented for skin regions, thus enables reliable testing of skin detection algorithms. Samples of the color images and the corresponding skin segmented images in the ECU face and skin detection database are shown in Figure 5.7. The same training set and testing set are used for all the skin classifiers under comparison, and are summarized in Table 5.2. This comparison has been largely facilitated by the sharing of the valuable image dataset, as well as many useful information and results provided by Dr. Chai and Dr. Phung, the authors of [23, 24] and [119, 117].

Table 5.2: ECU data set used for training and testing of skin classifiers

Datasets	No. Images	Skin Pixels	Nonskin pixels
Training Set	2,500	116.6 million	564.1 million
Testing set	500	27.1 million	97.5 million

In skin detection based on color pixel classification, the correct detection rate (CDR) is the percentage of skin pixels that are correctly classified. The false detection rate (FDR) is the percentage of nonskin pixels that are incorrectly classified. A good skin detector must have a high CDR and a low FDR. Performance of skin detectors can be analyzed through the plot of CDR versus FDR at different classifier settings. This plot is also known as the receiver operating characteristic (ROC) curve.

Figure 5.8 shows the ROC curves of the SOTT-based skin detectors and the five other skin detectors, namely Fixed Range classifier, Gaussian classifier, SOM classifier and two Bayesian classifiers. The references and the specifications of these classifiers can also be found in Table 5.3. It can be seen that SOTT-Bayesian

Figure 5.7: Examples in the ECU face detection image database: (a) Original image examples. (b) Manually segmented skin images

scheme performs better than the Semi-Adaptive scheme, and they both performs better than SOM, Gaussian and fixed range classifiers. Meanwhile, as expected, SOTT-Bayesian scheme with 64 leaf-layer neurons does not performs as good as Bayesian RGB $64 \times 64 \times 64$ bins, since the latter has much better resolution histogram, i.e., 64^3 bins. However, the performance of SOTT-Bayesian scheme in CbCr space with 64 neurons and that of Bayesian CbCr 32×32 bins are comparable. This is a very interesting result. It indicates that with similar number of clusters, SOTT is able to reasonably allocate its neurons in the color space and obtain better results than simply dividing the color space evenly.

The codebook obtained in SOTT-Bayesian scheme after the training is shown in Figure 5.9. The skin neurons are represented by red “+” s and non-skin neurons are represented by the black “x” s (the neurons are classified with the threshold set to 0.24). The network neighborhood is linked by the dotted lines. As can be seen, the neurons are more densely distributed at the skin color region than those outside the region. The blue rectangle of skin color region from [23] is marked as a reference. A red contour line on Fig. 5.9 marks a single enclosed region that demarcates skin and non-skin neurons. As expected, the closed continent of skin regions from both methods show comparably compact skin color region in CrCb space. This is a good feature because ideally the neurons should have denser distributions in regions where even a small shift in decision boundary may lead to a large change in classification accuracy. Such distribution can reduce the number of neurons required to achieve good performance as a skin color classifier. This is because more neurons are allocated at the areas of high ambiguity while the clearly distinguishable areas are represented by reasonably small number of neurons.

Figure 5.8: ROC curves of skin color classifiers

Figure 5.9: Codebook obtained after SOTT training

Figure 5.10: Results of skin color pixel classification. (a) Bayesian classifier with the histogram technique (64 bins per channel). (b) Gaussian classifier. (c) SOM classifier. (d) Proposed Semi-Adaptive scheme. (e) Proposed SOTT-Bayesian scheme.

Sample skin detection outputs produced by the two proposed SOTT-based schemes and other classifiers are shown in Figure 5.10. The proposed SOTT-Bayesian classifier and SOTT based semi-adaptive classifier have similar skin color outputs as Bayesian classifier, and produces better classification results than those generated from Gaussian classifier and traditional SOM classifier.

Table 5.3: Normalized speed and memory usage (bytes) of different skin color classifiers

Classifier	Specifications	Time (Normalized)	Memory storage (bytes)
Fixed range [23]	CbCr	0.5	32
2D Gaussian [103]	CbCr	17.5	50
SOM [19]	100 neurons	93.0	3,200
Bayesian [119]	RGB, 64 ³ bins	1	2,097,152
Bayesian [119]	CbCr, 32 ² bins	1	8,192
Semi-Adaptive Scheme	CbCr, 64 leaf neurons	6.6	1,856
SOTT-Bayesian Scheme	CbCr, 64 leaf neurons	6.6	1,856

Table 5.3 shows the comparison of speed and memory storage requirement of the classifiers being investigated. Note that our estimations are based on the direct implementations of the classifiers, without using any heuristic to reduce the classifier memory. To achieve a machine-independent comparison of classifier speed, we normalize all time measures against the running time of the Bayesian classifier. The Bayesian classifier with 64 bins per channel was used to process an image of size 352×288 pixels. In our experiments, the algorithms are coded in C++ and run on a Pentium III 733MHz PC with 256MB RAM. The two proposed schemes exhibit significant speed improvement over the SOM classifier. In terms of memory usage, fixed range method, and Gaussian method use almost negligible memory space. Bayesian classifier with 32^2 histogram bins, SOM classifier, and the two proposed schemes require moderate amounts of memory. On the contrary, Bayesian classifier with 64^3 histogram bins needs 2MB of memory space.

5.3 Chapter Summary

In this chapter, we have explored the performances of self-organizing topological maps in high quality color image rendering with limited color palettes and skin color detection.

The frequency-sensitive self-organizing maps and self-organizing topological tree proposed in Chapters 3 and 4 can be used to perform color quantization of images. However, to improve the color image quality, halftoning techniques are usually used after color quantization. The global butterfly jumping sequence instead of the commonly used raster sequence is embedded into a error diffusion

technique to distribute the distortion errors. With the help of the global butterfly jumping sequence, the performance of our proposed simple spatial compensation algorithm performs better than that of the traditional methods of error diffusion. The perceptual qualities of the rendered images are comparable to those obtained with the latest complex and computationally intensive Spatial Color Quantization method. The preliminary simulation results are promising, but more work is necessary to derive a good yardstick to substantiate the conjecture of superior perceptual quality.

Based on the SOTT's clustering ability, we also proposed two schemes to explore the feasibility of employing SOTT for skin color detection. The first proposed Semi-Adaptive scheme uses a small set of images for training. It outperforms several other well-known methods such as the Gaussian classifier and the SOM, but still has lower classification rates compared to the Bayesian classifier. On the other hand, the second proposed SOTT-Bayesian Scheme has comparable performance to the Bayesian classifier even when its number of neurons is smaller than the number of histogram bins of Bayesian. It needs an extra step for training and testing but otherwise it requires lower memory than the Bayesian method for similar performance. Although our findings show that SOTT might not be competitive to the best classifier, the preliminary assessments have been encouraging. SOTT can play a supplementary role, for example, it can be used to locate the skin neurons more efficiently in order to alleviate the memory burden of Bayesian classifiers. The evaluation have provided us an insight into the strength and limitation of SOTT as a skin color classifier.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The basic principle underlying competitive learning of self-organizing map methods dating back to early studies of cluster analysis in mathematical statistics. The dominance of SOM methods in the field of computer vision and image processing that have a long bearing on vector quantization and clustering is, however, inferior relative to their applications in other disciplines. The impediments arise from some inherent weaknesses of SOM in its basic form. The learning algorithms need to be enhanced in order to overcome its weaknesses and further motivate the choice of SOM for the image processing paradigm. Under this premise, the principal goal of the thesis is to develop new self-organizing algorithms with reduced computational complexity, robust performances, and advanced features over existing

self-organizing algorithms. Several problems and applications have been identified and systematically addressed in the preceding chapters of the thesis. The research brings together new algorithms, methods and insights which are quite generic and may well have broader applications in computer vision and image processing than those investigated. This section summarizes the important findings of our study, and recapitulates the major contributions of the work reported herein.

We preamble by a tutorial on vector quantization, image color reduction, and clustering techniques. We discussed the quantizer design metric in terms of the (R, D, A, M) quadruple and the significance of progressive coding. Existing quantization algorithms and techniques to accelerate the codebook design process have been summarized. Tree structure vector quantizer (TSVQ) as a structure-constrained quantizer with embedded coding feature has been discussed. The review of classic SOM and its variants has also revealed the challenges of their use in image clustering. First, the clustering result is sensitive to the number of partitions which has to be predefined. Second, the clustering result of one partition provides knowledge at only one similarity level and lastly, SOM based algorithms also suffer from the problem of favoring equally-sized compact spheroidal clusters, i.e., isotropic clusters.

The first contribution of our work is the proposal of a new competitive learning method that endows the SOM with greater learning efficiency and robustness in vector quantization performance. The proposed frequency sensitive self-organizing map (FS-SOM) harmonically blends the neighborhood adaptation of the SOM with the neuron dependent frequency sensitive learning model, the global butterfly permutation sequence for input randomization, and the reinitialization of dead neurons to harness effective utilization of neurons. The net effect is an im-

provement in adaptation, a well-ordered codebook and the effective alleviation of underutilization problem, which is the main cause of visually perceivable artifacts of color quantization. Extensive simulations have been performed to analyze and compare the learning behavior and performance of FS-SOM against other vector quantization algorithms. The results show that the proposed FS-SOM outperforms the classic competitive learning, LBG and SOM algorithms. More importantly, FS-SOM achieves its superiority in reconstruction quality and topological ordering with a much greater robustness against variations in network parameters than the current art SOM algorithm for color quantization. System designer can do away with aggressive parameter calibration without fearing of significant performance deviation.

The second major contribution of this thesis is a novel on-line Self-organizing Topological Tree (SOTT) to reduce the bottleneck best matching neuron search complexity from $O(N)$ to $O(\log N)$, where N is the number of neurons. A multi-path search is used to overcome the intrinsic bias of the tree-structured search. A new parameter named leakage ratio is introduced to control the performance tradeoff between different neuronal layers. Specific updating rule is designed to keep both the inter-layer and intra-layer relationships intact during the learning phase.

The hierarchical structure of the proposed SOTT is also exploited in a hybrid clustering algorithm to overcome the clustering performance deficiencies of the classic k-means and SOM algorithms. To fully exploit the advantages of online learning, and reduced computational complexity of the SOTT architecture, a new linkage metric is introduced for the agglomerative hierarchical clustering (AHC). This linkage metric can be updated online during the partition clustering stage and

does not need to wait for the patterns to be labeled. The proposed SOTT hybrid clustering algorithm has demonstrated to be computational efficient and possesses good scalability. Besides, its clustering performance is more robust against learning parameter variation compared with the growing variants of SOM.

The experimental results show that, the performance of the proposed SOTT is comparable to the full-search SOM. At the same time, the computation efficiency of SOTT is much better than that of SOM and other vector quantizers. In addition, the progressive encoding property offered by SOTT will be of value to harnessing new efficient and elegant solutions to data clustering and pattern classification problems.

The remaining contributions pertain to the exploration and performance appraisal of the proposed self-organizing algorithms in two applications. We have proposed a new halftoning technique based on global butterfly jumping sequence (GBJS) error diffusion. Halftoning, like many other image processing tasks, are usually carried out in raster scan order. However, by incorporating the GBJS into the error diffusion method, some intriguing advantages are observed. We compared the proposed GBJS based color halftoning technique with the classic error diffusion technique and a state-of-the-art model based color reduction algorithm. The results are promising and the quality of the color rendered images are dramatic. The key point is that the proposed error diffusion method is very simple and has a much lower computational complexity than the spatial color quantization algorithm with comparable, if not better perceptual output quality. For a complete color reduction application for high quality rendering, if SOTT is used as the color quantizer before halftoning, the search time for the best matching codevector can be reduced significantly.

Finally, based on the SOTT's clustering ability, we proposed two training schemes to explore the feasibility of employing SOTT for skin color detection. The semi-adaptive scheme uses a small set of images for training. It outperforms several other well known methods such as the Gaussian classifier and the SOM classifier, but still has lower classification rates compared to the Bayesian classifier. On the other hand, the SOTT-Bayesian scheme has comparable performance to the Bayesian classifier even when its number of neurons is smaller than the number of histogram bins of Bayesian. It needs an extra step for training and testing but requires lower memory than the Bayesian method for similar performance. The SOTT is constructed as a LVQ with reduced computation time than the SOM while improving the detection accuracy. Overall, SOTT is appraised to be inferior to the best Bayesian classifier in terms of its classification performance. However, it can foreseeably have impact on any effective classifier where homogeneity grouping is of value. In this context, SOTT can supplement Bayesian classifier by eliminating a number of superfluous bins and reallocating the bins to focus subsequent processing on the more dominant ranges.

6.2 Recommendations for Future Work

As usual, there are limitations and variations to the new methods presented. Based on the research presented in this thesis, we suggest a number of possible extensions and potential applications worthy of further investigation.

1. Extension of the balanced self-organizing topological tree learning algorithm to unbalanced tree and define its construction strategy. The SOTT algo-

rithm proposed in Chapter 3 assumes a balanced structure. A balanced or symmetrical structure might be too rigid for some applications as in density estimation and clustering. For example, in the hybrid clustering scheme, we use the merging results of original clusters to achieve better clustering. There is a marked distinction between the function of vector quantization for the compression of signal information, and for segregation of signals into categories. For classification problem, the exact locations of the codevectors within the class is unimportant and most codevectors shall be tuned towards the class borders where the class distributions intersect and cause misclassification errors. In this sense, the variance of samples in a class is a more important factor. With asymmetrical or unbalanced topological tree, two important decisions for classification algorithm, i.e., the selection of the optimal number of codebook vectors for each class and tuning of the neuron weights at each layers for minimum misclassification may be accommodated. Appropriate depth control parameters and different training strategies can be built into the evolvable self-organizing topological tree for better rate-distortion performance or increased classification accuracy depending on the target applications.

2. Raster sequence is usually used as a default sequence for halftoning, but our study shows that the proposed GBS-based error diffusion method exhibits dramatic advantages than the raster, serpentine and space-filling sequence to produce high-quality color rendered image. It appears that defining a universally acceptable objective measure of perceptual quality for color rendered images is far from trivial. More analysis and evaluation will be needed to substantiate the performance of the proposed halftoning method. It would be interesting to establish an analytical model or an empirical model by means

of rigorous experimentation to predict the image dependent and image independent distortion behaviors of our algorithm. The developed mathematical model will enable us to deduce the optimum error diffusion filter coefficients to adapt to the spatial kernel.

3. The application of SOTT to some of the challenging problems, such as image segmentation [29], content based image retrieval(CBIR) [133] and face and eye detection [119] in human-machine interface where the analysis and extraction of multiple visual features in one unified architecture has turned out to be an extremely difficult problem. As solutions to these problems usually entail a heterogeneous mass of techniques, we believe that abstraction of multiple features and clusters may still be amenable to the hierarchical topological tree that subsumes several different maps. The intrinsic potential of SOTT, such as the topology-ordering, multi-scale and tree-structured search can be useful in dealing with different stochastic signals. The effect of using a non-uniform control and time-variant training parameters at different layers of the SOTT is unclear and may be further explored with these applications.
4. Most of the theoretic research, and simulations carried out for the self-organizing network (we are no exception) have been restricted on relatively small networks of few tens to a few hundreds of neurons. The main problem with most realizations seems to be the slow convergence of learning and the off-line learning requirements for some SOMs. One of the motives for exploring SOM and developing better learning mechanisms in this thesis is to be able to effectively leverage on the vast capacities of active circuits that can be manufactured using state-of-the-art semiconductor or optoelec-

Conclusions and Future Work

tronic technologies. There are immense interest and opportunities to VLSI or mixed-signal circuit designers to further the architectural research on the implementation of large parallel FS-SOM or SOTT network with thousands of low or modest complexity computing elements.

Author's Publications

Journal Papers

- [1] C. H. Chang, **P. Xu**, R. Xiao, and T. Srikanthan, "New adaptive color quantization method based on self-organizing maps," *IEEE Trans. Neural Networks*, vol. 16, no. 1, pp. 237–249, Jan. 2005.

- [2] **P. Xu**, C. H. Chang, and A. Paplinski, "Self-organizing topological tree for on-line vector quantization and data clustering," *IEEE Trans. Syst., Man, Cybern.B*, vol. 35, no. 3, pp. 515–526, June 2005.

Conference Papers

- [1] X. Deng, **P. Xu** and C. H. Chang, "Self organizing topological tree for skin color detection," in *Proc. IEEE Asia-Pacific Conference on Circuits and Systems*, Tainan, Taiwan, vol. 2, Dec. 2004, pp. 1097–1100.

- [2] C. H. Chang and **P. Xu**, "Frequency sensitive self-organizing maps and its application in color quantization," in *Proc. IEEE Int. Symp. Circuits and Systems*

(*ISCAS*), Vancouver, Canada, vol. 5, May 2004, pp. 804–807.

[3] **P. Xu** and C. H. Chang, “Self-organizing topological tree,” in *Proc. Int. Symp. Circuits and Systems (ISCAS)*, Vancouver, Canada, vol. 5, May 2004, pp. 732–735.

Bibliography

- [1] L. Akarun, Y. Yardimci, and A. E. Cetin, "Adaptive methods for dithering color images," in *Proc. IEEE Int. Conf. Image Processing*, vol. 3, Washington DC, USA, 1995, pp. 125–128.
- [2] —, "Adaptive methods for dithering color images," *IEEE Trans. Image Processing*, vol. 6, no. 7, pp. 950–955, July 1997.
- [3] D. Alahakoon, S. K. Halgamuge, and B. Srini-vasan, "Dynamic self-organizing maps with controlled growth for knowledge discovery," *IEEE Trans. Neural Networks*, vol. 11, no. 3, pp. 601–614, May 2000.
- [4] J. P. Allebach, T. J. Flohr, and D. P. Hilgenberg, "Model-based halftoning via direct binary search," in *Proc. Int. Conf. Photographic Sci. (ICPS): Physics and Chemistry of Imaging Systems*, 1994, vol. 2, pp. 476–482.
- [5] C. Amerijckx, M. Verleysen, P. Thissen, and J. D. Legat, "Image compression by self-organized Kohonen map," *IEEE Trans. Neural Networks*, vol. 9, no. 3, pp. 503–507, May 1998.
- [6] D. Anastassiou, "Error diffusion coding for A/D conversion," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1175–1186, Sept. 1989.
- [7] M. R. Anderberg, *Cluster Analysis for Applications*. New York: Academic Press, Inc., 1973.
- [8] A. Ansari and A. Fineberg, "Image data ordering and compression using peano scan and lot," *IEEE Trans. Consumer Electron.*, vol. 38, no. 3, pp. 436–445, June 1992.
- [9] R. A. Baeza-Yates, "Introduction to data structures and algorithms related to information retrieval," in *Information Retrieval: Data Structures and Algorithms*, B. Frakes, Ed. Upper Saddle River, NJ: Prentice-Hall, 1992.

Bibliography

-
- [10] R. Balasubramanian and J. P. Allebach, "A new approach to palette selection for color images," *J. Imaging Tech.*, vol. 17, pp. 284–290, Dec. 1991.
 - [11] R. Balasubramanian, C. A. Bouman, and J. P. Allebach, "Sequential scalar quantization of color images," *J. Electron Imag.*, vol. 3, no. 1, pp. 45–49, 1994.
 - [12] —, "Sequential scalar quantization of vectors: an analysis," *IEEE Trans. Image Processing*, vol. 4, no. 9, pp. 1282–1295, Sept. 1995.
 - [13] G. Ball and D. Hall, "Isodata, a novel method of data analysis and classification," Stanford University, Stanford, CA, Tech. Rep., 1965.
 - [14] J. Bares, Ed., *Proc. Int. Soc. Opt. Eng. (SPIE): Color Imaging–Device Independent Color, Color Hard Copy and Graphic Arts*, vol. 2658, 1996.
 - [15] P. Berhkin, "Survey of clustering data mining techniques," Accrue Software, Tech. Rep., 2002.
 - [16] J. Blackmore and R. Miikkulainen, "Incremental grid growing: Encoding high-dimensional structure into a two-dimensional feature map," in *Proc. of the IEEE Int. Conf. on Neural Networks (ICNN)*, vol. 1, San Francisco, USA, 1993, pp. 450–455.
 - [17] J. P. Braquelaire and L. Brun, "Comparison and optimization of methods of color image quantization," *IEEE Trans. Image Processing*, vol. 6, no. 7, pp. 1048–1052, July 1997.
 - [18] G. Braudaway, "A procedure for optimum choice of a small number of colors from a large palette for color imaging," in *Proc. on Electronic Imaging West '87*, Anaheim, CA, Feb. 1987, pp. 71–75.
 - [19] D. Brown, I. Craw, and J. Lewthwaite, "A SOM based approach to skin detection with application in real time systems," in *Proc. of the British Machine Vision Conf.*, Birmingham, UK, 2001, pp. 491–500.
 - [20] P. Burt and E. Adelson, "The Laplacian pyramid as a compact image code," *IEEE Trans. Commun.*, vol. 31, no. 4, pp. 532–540, Apr. 1983.
-

Bibliography

- [21] A. Buzo, A. H. Gray, R. M. Gray, and J. D. Markel, "Speech coding based upon vector quantization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 28, no. 5, pp. 562–574, Oct. 1980.
- [22] D. Chai and A. Bouzerdoun, "A bayesian approach to skin color classification in YCbCr color space," in *Proc. IEEE Region Ten Conference (TENCON)*, vol. 2, 2000, pp. 421–424.
- [23] D. Chai and K. N. Ngan, "Face segmentation using skin color map in video-phone applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 4, pp. 511–564, 1999.
- [24] D. Chai, S. L. Phung, and A. Bouzerdoun, "A bayesian skin/non-skin color classifier using non-parametric density estimation," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, vol. 2, 2003, pp. 464–467.
- [25] C. H. Chang and P. Xu, "Frequency sensitive self-organizing maps and its application in color quantization," in *Proc. Int. Sym. Circuits and Systems*, vol. 5, Vancouver, Canada, May 2004, pp. 804–807.
- [26] C. H. Chang, P. Xu, R. Xiao, and T. Srikanthan, "New adaptive color quantization method based on self-organizing maps," *IEEE Trans. Neural Networks*, vol. 16, no. 1, pp. 237–249, Jan. 2005.
- [27] R. F. Chang, W. Chen, and J. Wang, "Image sequence coding using adaptive TSVQ with multi-path searching," *IEE Proc. I*, vol. 139, no. 1, pp. 9–14, 1992.
- [28] O. T.-C. Chen, B. J. Sheu, and W.-C. Fang, "Image compression using self-organization networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 5, pp. 480–489, Oct. 1994.
- [29] H. D. Cheng, X. H. Jiang, Y. Sun, and J. L. Wang, "Color image segmentation: advances and prospects," *Pattern Recognition*, vol. 34, no. 12, pp. 2259–2281, Dec. 2001.
- [30] T. D. Chiueh, T. T. Tang, and L. G. Chen, "Vector quantization using tree-structured self-organizing feature maps," *IEEE J. Select. Areas Commun.*, vol. 12, no. 9, pp. 1594–1599, Dec. 1994.

Bibliography

-
- [31] F. L. Chung and T. Lee, "Fuzzy competitive learning," *IEEE Trans. Neural Networks*, vol. 7, no. 3, pp. 539–551, 1994.
 - [32] L. D. Davission, "Information rates for data compression," in *Proc. IEEE WESCON*, 1968.
 - [33] W. H. E. Day, "Complexity theory: An introduction for practitioners of classification," in *Clustering and Classification*, P. Arabie, L. J. Hubert, and G. D. Soete, Eds. River Edge, NJ: World Scientific Publishing Co., 1992.
 - [34] A. Dekker, "Kohonen neural network for optimal color quantization," *Network: Computation in Neural Systems*, vol. 5, no. 3, pp. 351–367, Aug. 1994.
 - [35] X. Deng, P. Xu, and C. H. Chang, "Self organizing topological tree for skin color detection," in *Proc. IEEE Asia-Pacific Conference on Circuits and Systems*, vol. 2, Tainan, Taiwan, Dec. 2004, pp. 1097–1100.
 - [36] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: John Wiley & Sons, Inc., 2001.
 - [37] M. Edwards and D. R. Morse, "The potential for computer-aided identification in biodiversity research," *Trends in Ecology and Evolution*, vol. 10, no. 4, pp. 153–158, Apr. 1995.
 - [38] W. Equitz, "A new vector quantization clustering algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 10, pp. 1568–1575, Oct. 1989.
 - [39] E. Erwin, K. Obermayer, and K. Schulten, "Self-organizing maps: ordering, convergence properties and energy functions," *Biological Cybernetics*, vol. 67, no. 1, pp. 47–55, Jan. 1992.
 - [40] T. R. Fischer, "A pyramid vector quantizer," *IEEE Trans. Inform. Theory*, vol. 32, no. 4, pp. 568–583, July 1986.
 - [41] R. W. Floyd and L. Steinberg, "An adaptive algorithm for spatial greyscale," in *Proc. Society for Information Display*, vol. 17, 1976, pp. 75–77.
 - [42] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 209–226, Sept. 1977.
-

Bibliography

-
- [43] B. Fritzke, "Growing grid – a self-organizing network with constant neighborhood range and adaption strength," *Neural Processing Letters*, vol. 2, no. 5, pp. 9–13, 1995.
 - [44] A. S. Galanopoulos, R. L. Moses, and S. C. Ahalt, "Diffusion approximation of frequency sensitive competitive learning," *IEEE Trans. Neural Networks*, vol. 8, no. 5, pp. 1026–1030, Sept. 1997.
 - [45] A. Gamal, L. Hemachandra, I. Shperling, and V. Wei, "Using simulated annealing to design good codes," *IEEE Trans. Inform. Theory*, vol. 33, no. 1, pp. 116–123, Jan. 1987.
 - [46] C. Garcia and G. Tziritas, "Face detection using quantized skin color regions merging and wavelet packet analysis," *IEEE Trans. Multimedia*, vol. 1, no. 3, pp. 264–277, Sept. 1999.
 - [47] M. R. Garey, D. S. Johnson, and H. S. Witsenhausen, "The complexity of the generalized LloydMax problem," *IEEE Trans. Inform. Theory*, vol. 28, no. 2, pp. 255–256, Mar. 1982.
 - [48] Y. Gdalyahu, D. Weinshall, and M. Werman, "Self-organizing in vision: stochastic clustering for image segmentation, perceptual grouping, and image database organization," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 10, pp. 1053–1074, Oct. 2001.
 - [49] R. S. Gentile, J. P. Allebach, and E. Walowit, "Quantization of color images based on uniform color spaces," *J. Imaging Technol.*, vol. 16, no. 1, pp. 11–21, Feb. 1990.
 - [50] R. S. Gentile, E. Walowit, and J. P. Allebach, "Quantization and multi-level halftoning of color images for near-original image quality," *J. Opt. Soc. Amer.-A*, vol. 7, no. 6, pp. 1019–1026, June 1990.
 - [51] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer, 1992.
 - [52] M. Gervautz and W. Purgathofer, "A simple method for color quantization: Octree quantization," in *Graphic Gems*, A. S. Glassner, Ed. New York: Academic Press, 1990.
-

Bibliography

-
- [53] J. Giske, G. Huse, and O. Fiksen, "Modeling spatial dynamics of fish," *Reviews in Fish Biology and Fisheries*, vol. 8, pp. 57–91, 1998.
 - [54] J. R. Goldschneider, E. A. Riskin, and P. W. Wong, "Embedded color error diffusion," in *Proc. IEEE Int. Conf. Image Processing*, vol. 1, Sept. 1996, pp. 565–569.
 - [55] —, "Embedded multilevel error diffusion," *IEEE Trans. Image Processing*, vol. 6, no. 7, pp. 956–964, July 1997.
 - [56] R. M. Gray, *Source Coding Theory*. Norwell, MA: Kluwer, 1990.
 - [57] R. M. Gray and Y. Linde, "Vector quantizers and predictive quantizers for Gauss-Markov sources," *IEEE Trans. Commun.*, vol. 30, no. 2, pp. 381–389, Feb. 1982.
 - [58] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2325–2383, Oct. 1998.
 - [59] S. Guha, R. Rastogi, and K. Shim, "Cure: An efficient clustering algorithm for large databases," in *Proc. ACM Int. Conf. on Management of Data (SIGMOD)*, Seattle, Washington, 1998.
 - [60] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems Archive*, vol. 17, no. 2–3, pp. 107–145, Dec. 2001.
 - [61] J. A. Hartigan, *Clustering Algorithms*. New York: Wiley, 1975.
 - [62] B. Haskell, "The computation and bounding of rate-distortion functions," *IEEE Trans. Inform. Theory*, vol. 15, no. 5, pp. 525–531, Sept. 1969.
 - [63] S. Haykin, *Neural Networks — A Comprehensive Foundation*. New York: Macmillan College Publishing Company, Inc., 1994, ch. 10.
 - [64] R. Hecht-Nielsen, "Counterpropagation networks," *Appl. Opt.*, vol. 26, pp. 4979–4984, 1987.
 - [65] P. Heckbert, "Color image quantization for frame buffer display," *Comput. Graph.*, vol. 16, no. 4, pp. 297–307, July 1982.
-

Bibliography

-
- [66] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Reading, MA: Addison-Wesley, 1991.
 - [67] R. L. Hsu, M. Abdel-Mottaleb, and A. K. Jain, "Face detection in color images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 5, pp. 696–707, May 2002.
 - [68] C. M. Huang, "Fast full search equivalent encoding algorithms for image compression using vector quantization," *IEEE Trans. Image Processing*, vol. 1, no. 3, pp. 413–416, July 1992.
 - [69] D. H. Hubel and T. N. Wiesel, "Functional architecture of macaque visual cortex," *Proceedings of the Royal Society of London*, vol. 198, no. 1130, pp. 1–59, July 1977.
 - [70] L. Hyafil and R. L. Rivest, "Construction of optimal binary decision trees is NP-complete," *Inform. Processing Lett.*, vol. 5, pp. 15–17, May 1976.
 - [71] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 23, no. 1, pp. 67–72, Feb. 1975.
 - [72] F. Itakura and S. Saito, "Analysis synthesis telephony based on the maximum likelihood method," in *Proc. 6th International Congress on Acoustics*, Tokyo, Japan, Aug. 1968, pp. 17–20.
 - [73] A. Jain and R. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
 - [74] A. Jain, N. M. M, and P. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, pp. 264–323, Sept. 1999.
 - [75] J. F. Jarvis, C. N. Judice, , and W. H. Ninke, "A survey of techniques for the display of continuous tone images on bilevel displays," *Comput. Graph. Image Processing*, vol. 5, pp. 13–40, Mar. 1976.
 - [76] M. Jones and J. Rehg, "Statistical color models with application to skin detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Fort Collins, CO, 1999, pp. 274–280.
-

Bibliography

- [77] G. Joy and Z. Xiang, "Center-cut for color-image quantization," *Vis. Comput.*, vol. 10, no. 1, pp. 62–66, Jan. 1993.
- [78] D. B. Judd and G. Wyszecki, *Color in Business, Science and Industry*, 3rd ed. New York: Wiley, 1975.
- [79] G. Karypis, E. H. Han, and V. Kumar, "Chameleon: A hierarchical clustering algorithm using dynamic modeling," *IEEE Computer: Special Issue on Data Analysis and Mining*, vol. 32, no. 8, pp. 68–75, 1999.
- [80] T. Kaukoranta, P. Franti, and O. Nevalainen, "A fast exact GLA based on code vector activity detection," *IEEE Trans. Image Processing*, vol. 9, no. 8, pp. 1337–1342, Aug. 2000.
- [81] B. King, "Step-wise clustering procedures," *J. Am. Stat. Assoc.*, vol. 69, pp. 86–101, 1967.
- [82] T. D. Kite, B. L. Evans, and A. Bovik, "Modeling and quality assessment of halftoning by error diffusion," *IEEE Trans. Image Processing*, vol. 9, no. 5, pp. 909–922, May 2000.
- [83] K. T. Knox, "Error image in error diffusion," *Proc. SPIE*, vol. 1657, pp. 268–279, Feb. 1992.
- [84] —, "Error diffusion: A theoretical view," in *Proc. SPIE: Human Vision, Visual Processing and Digital Display IV*, vol. 1913, 1993, pp. 326–331.
- [85] K. T. Knox and R. Eschbach, "Threshold modulation in error diffusion," *J. Electron. Imaging*, vol. 2, pp. 185–192, July 1993.
- [86] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, pp. 59–69, 1982.
- [87] —, "The self-organizing map," *Proceeding of the IEEE*, vol. 78, no. 9, pp. 1464–1480, Sept. 1990.
- [88] —, *Self-organizing Maps*. Berlin, Germany: Springer-Verlag, 2001.
- [89] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas, "Engineering applications of the self-organizing map," *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1358–1384, Oct. 1996.

Bibliography

-
- [90] P. Koikkalainen, "Progress with the tree-structured self-organizing map," in *11th European Conf. on Artificial Intelligence (ECAI94)*, Amsterdam, 1994, pp. 211–215.
 - [91] P. Koikkalainen and E. Oja, "Self-organizing hierarchical feature maps," in *Proc. Int. Joint Conf. on Neural Networks (IJCNN90)*, vol. 2, San Diego, CA., 1990, pp. 279–284.
 - [92] S. Kollias and D. Anastassiou, "A unified neural network approach to digital image halftoning," *IEEE Trans. Signal Processing*, vol. 39, no. 4, pp. 980–984, Apr. 1991.
 - [93] B. W. Kolpatzik and C. A. Bouman, "Optimized error diffusion for image display," *J. Electron. Imag.*, vol. 1, pp. 277–292, Jan. 1992.
 - [94] W. L. G. Koontz, P. M. Narendra, and K. Fukunaga, "A branch and bound clustering algorithm," *IEEE Trans. Comput.*, vol. 24, pp. 908–915, Sept. 1975.
 - [95] W. Kou, *Digital Image Compression, Algorithms and Standards*. Norwell, MA: Kluwer, 1995.
 - [96] A. K. Krishnamurthy, S. C. Ahalt, D. E. Melton, and P. Chen, "Neural networks for VQ of speech and images," *IEEE J. Select. Areas Commun.*, vol. 38, no. 1, pp. 25–29, Jan. 1992.
 - [97] P. Kuchi, P. Gabbur, S. Bhat, and S. David, "Human face detection and tracking using skin color modeling and connected component operators," *IETE Journal of Research, Special issue on Visual Media Processing*, vol. 38, no. 3&4, pp. 289–293, May-Aug. 2002.
 - [98] A. Lempel and J. Ziv, "Compression of two-dimensional data," *IEEE Trans. Inform. Theory*, vol. 32, no. 1, pp. 2–8, Jan. 1986.
 - [99] C. R. Lin and M. S. Chen, "A robust and efficient clustering algorithm based on cohesion self-merging," in *Proc. of the 8th ACM Int. Conf. on Knowledge discovery and data mining*, Edmonton, Alberta, Canada, 2002, pp. 582–587.
 - [100] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. 28, no. 1, pp. 84–95, Jan. 1980.
-

Bibliography

-
- [101] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Mathematical Statistics and Probability*, vol. 1, 1967, pp. 281–297.
 - [102] T. M. Martinez, S. G. Berkovich, and K. J. Schulten, "Neural-gas network for vector quantization and its application to time-series prediction," *IEEE Trans. Neural Networks*, vol. 4, no. 4, pp. 558–568, July 1993.
 - [103] B. Menser and M. Wien, "Segmentation and tracking of facial regions in color image sequences," *Proc. SPIE Visual Comm. and Image Processing*, vol. 4067, pp. 731–740, 2000.
 - [104] R. Miller, J. Sullivan, and G. Pios, "Image halftoning using a visual model in error diffusion," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 1, pp. 33–38, Jan./Feb. 1991.
 - [105] ———, "Image halftoning using a visual model in error diffusion," *J. Opt. Soc. Amer. A*, vol. 10, no. 8, pp. 1714–1724, Aug. 1993.
 - [106] F. Murtagh, "A survey of recent advances in hierarchical clustering algorithms which use cluster centers," *The Computer Journal*, vol. 26, no. 4, pp. 354–359, 1984.
 - [107] M. Murty and G. Krishna, "A computationally efficient technique for data clustering," *Pattern Recogn.*, vol. 12, no. 3, pp. 153–158, 1980.
 - [108] G. Nagy, "State of the art in pattern recognition," *Proceeding of the IEEE*, vol. 56, no. 5, pp. 836–862, 1968.
 - [109] A. Ng, M. Jordan, and Y. Weiss, "Spectral clustering: On spectral clustering: analysis and an algorithm," in *Proc. Neural Information Processing Systems*, 2001, pp. 849–856.
 - [110] M. T. Orchard, "A fast nearest neighbor search algorithm," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 1991, pp. 2297–2300.
 - [111] M. T. Orchard and C. A. Bouman, "Color quantization of images," *IEEE Trans. Signal Processing*, vol. 39, no. 12, pp. 2677–2690, Dec. 1991.
-

Bibliography

-
- [112] N. Papamarkos, A. E. Atsalakis, and C. P. Strouthopoulos, "Adaptive color reduction," *IEEE Trans. Syst., Man, Cybern. B*, vol. 32, no. 1, pp. 44–56, Jan. 2002.
 - [113] T. N. Pappas, "Model-based halftoning of color images," *IEEE Trans. Image Processing*, vol. 6, no. 7, pp. 1014–1024, July 1997.
 - [114] T. N. Pappas and D. L. Neuhoff, "Model-based halftoning," in *Proc. SPIE: Human Vision, Visual Processing and Digital Display II*, vol. 1453, 1991, pp. 244–255.
 - [115] T. Pappas, "Optimized error diffusion based on a human vision model," in *Proc. SPIE: Human Vision, Visual Processing and Digital Display III*, vol. 1666, 1992, pp. 152–164.
 - [116] S. C. Pei and Y. S. Lo, "Color image compression and limited display using self-organization kohonen map," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 2, pp. 191–205, Apr. 1998.
 - [117] S. L. Phung, A. Bouzerdoun, and D. Chai, "Skin segmentation using color pixel classification: Analysis and comparison," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 1, pp. 148–154, Jan. 2005.
 - [118] S. L. Phung, D. Chai, and A. Bouzerdoun, "A distribution-based face/non-face classifier using gaussian mixtures," in *Proc. Int. Conf. on Signal and Image Processing*, 2002, pp. 485–490.
 - [119] S. Phung, "Automatic human face detection in color images," Ph.D. dissertation, Edith Cowan University, Perth, Australia, 2003.
 - [120] J. Puzicha, M. Held, J. Ketterer, J. M. Buhmann, and D. W. Fellner, "On spatial quantization of color images," *IEEE Trans. Image Processing*, vol. 9, no. 4, pp. 666–682, Apr. 2000.
 - [121] L. R. Rabiner and B. H. Juang, *Fundamentals of speech recognition*. Englewood Cliffs, N.J.: PTR Prentice Hall, 1993.
 - [122] A. Rauber, D. Merkl, and M. Dittenbach, "The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data," *IEEE Trans. Neural Networks*, vol. 13, no. 6, pp. 1331–1341, Nov. 2002.
-

Bibliography

-
- [123] P. Roetling and R. P. Loce, "Digital halftoning," in *Image Processing: Fundamentals and Applications*, E. R. Dougherty, Ed. Ed. New York: Marcel Dekker, 1994.
 - [124] K. Rose, "A mapping approach to rate-distortion computation and analysis," *IEEE Trans. Inform. Theory*, vol. 40, no. 6, pp. 1939–1952, Nov. 1994.
 - [125] A. Rosenfeld and M. Thurtson, "Edge and curve detection for visual scene analysis," *IEEE Trans. Comput.*, vol. 20, pp. 562–569, May 1971.
 - [126] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, no. 1, pp. 23–38, Jan. 1998.
 - [127] M. R. Schroeder, "Images from computers," *IEEE Spectrum*, vol. 6, no. 3, pp. 66–78, Mar. 1969.
 - [128] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, July 1948.
 - [129] L. G. Shapiro and G. C. Stockman, *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 2001.
 - [130] G. Sharma and H. J. Trussell, "Digital color imaging," *IEEE Trans. Image Processing*, vol. 6, no. 7, pp. 901–973, Nov. 1997.
 - [131] G. Shepherd, *Neurobiology*, 3rd ed. New York: Oxford University Press, 1994.
 - [132] J. Shiau and Z. Fan, "A set of easily implementable coefficients in error diffusion with reduced worm artifacts," in *Proc. SPIE: Color Imaging-Device Independent Color, Color Hard Copy and Graphic Arts*, 1996, pp. 265–270.
 - [133] A. W. M. Smeulders, M. Worring, A. Gupta, and R. Jain, "Content-based image retrieval at the end of early years," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 12, pp. 1349–1380, Dec. 2000.
 - [134] P. Sneath and R. Sokal, *Numerical Taxonomy*. London, UK: Freeman, 1973.
 - [135] E. Soijananin, "Writing sequences on the plane," *IEEE Trans. Inform. Theory*, vol. 48, no. 6, pp. 1344–1354, June 2002.
-

Bibliography

-
- [136] F. Solina, P. Peer, B. Batagelj, S. Juvan, and J. Kovac, "Color-based face detection in the 15 seconds of fame art installation," in *Proc. of Mirage 2003*, INRIA Rocquencourt, France, Mar. 2003, pp. 38–47.
 - [137] S. Spors and R. Rabenstein, "A real-time face tracker for color video," in *Proc. IEEE Int. Conf. on Acoust., Speech and Signal Processing*, Salt Lake City, USA, May 2001, pp. 1–4.
 - [138] J. Sullivan, R. Miller, and G. Pios, "Image halftoning using a visual model in error diffusion," *Journal of the Optical Society of America A*, vol. 10, no. 8, pp. 1714–1724, Aug. 1993.
 - [139] J. Sullivan, L. Ray, and R. Miller, "Design of minimum visual modulation halftone patterns," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 1, pp. 33–38, Jan. 1991.
 - [140] N. Ta, Y. Attikiouzel, and C. Crebbin, "Vector quantization of images using the competitive networks," in *Proc. 2nd Australian Conf. Neural Networks*, Sydney, 1991, pp. 256–262.
 - [141] K. Takaya and K. Choi, "Detection of facial components in a video sequence by independent component analysis," in *Proc. 3rd Int. Conf. on Independent Component Analysis and Blind Signal Separation*, San Diego, California, Dec. 2001, pp. 266–271.
 - [142] B. Tian, H. X. Tian, and K. C. Yi, "A fast codebook design algorithm for vector quantization," in *Proc. 5th Int. Conf. on Signal Processing (WCCC-ICSP)*, vol. 3, 2000, pp. 2004–2007.
 - [143] S. B. Udin and J. W. Fawcett, "Formation of topographic maps," *Annual Review of Neuroscience*, vol. 11, pp. 289–327, 1988.
 - [144] R. A. Ulichney, "Dithering with blue noise," *Proceeding of the IEEE*, vol. 76, no. 1, pp. 56–79, Jan. 1988.
 - [145] R. Ulichney, *Digital Halftoning*. Cambridge, MA: MIT Press, 1981.
 - [146] N. D. Venkata, "Analysis and design and of vector error diffusion systems for image halftoning," Ph.D. dissertation, Univ. Texas, Austin, TX, 2000.
-

Bibliography

- [147] N. D. Venkata and B. L. Evans, "Design and analysis of vector color error diffusion halftoning systems," *IEEE Trans. Image Processing*, vol. 10, no. 10, pp. 1552–1565, Oct. 2001.
 - [148] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing map," *IEEE Trans. Neural Networks*, vol. 11, no. 3, pp. 586–600, May 2000.
 - [149] Y. Wang and B. Yuan, "A novel approach for human face detection from color images under complex background," *Pattern Recognition*, vol. 34, no. 10, pp. 1983–1992, Oct. 2001.
 - [150] J. H. J. Ward, "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association*, vol. 58, pp. 236–244, 1963.
 - [151] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
 - [152] R. Williams, *Electrical Engineering Probability*. St. Paul, MN: West, 1991.
 - [153] P. W. Wong, "Adaptive error diffusion and its application in multiresolution rendering," *IEEE Trans. Image Processing*, vol. 5, no. 7, pp. 1184–1196, July 1996.
 - [154] H. Wu, Q. Chen, and M. Yachida, "Face detection from color images using a fuzzy pattern matching method," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 21, no. 6, pp. 557–563, June 1999.
 - [155] X. Wu and N. Memon, "Context-based, adaptive, lossless image coding," *IEEE Trans. Commun.*, vol. 45, no. 4, pp. 437–444, Apr. 1997.
 - [156] Z. G. Xiang and G. Joy, "Color image quantization by agglomerative clustering," *IEEE Comput. Graph. Applicat.*, vol. 14, no. 3, pp. 44–48, May 1994.
 - [157] R. Xiao, C. H. Chang, and T. Srikanthan, "On the initialization and training methods for kohonen self-organizing feature maps in color image quantization," in *Proc. 1st Int. Workshop on Electronic Design, Test and Applications (DELTA)*, New Zealand, 2002, pp. 321–325.
-

- [158] P. Xu and C. H. Chang, "Self-organizing topological tree," in *Proc. IEEE Int. Sym. Circuits and Systems*, vol. 5, Vancouver, Canada, May 2004, pp. 732–735.
- [159] P. Xu, C. H. Chang, and A. Paplinski, "Self-organizing topological tree for online vector quantization and data clustering," *IEEE Trans. Syst., Man, Cybern. B*, vol. 35, no. 3, pp. 515–526, June 2005.
- [160] K.-M. Yang, L. Wu, and M. Mils, "Fractal based image coding scheme using peano scan," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, Espoo, Finland, June 1988, pp. 2301–2304.
- [161] M. Yang, D. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 1, pp. 34–58, Jan. 2002.
- [162] H. Yin and N. M. Allinson, "Bayesian learning for self-organizing maps," *Electron. Lett.*, vol. 33, no. 4, pp. 304–305, Feb. 1997.
- [163] —, "Self-organizing mixture networks for probability density estimation," *IEEE Trans. Neural Networks*, vol. 12, no. 2, pp. 405–411, Mar. 2001.
- [164] K. Zeger and A. Gersho, "Pseudo-gray coding," *IEEE Trans. Commun.*, vol. 38, no. 12, pp. 2147–2156, May 1990.
- [165] X. Zhu, J. Yang, and A. Waibel, "Segmenting hands of arbitrary color," in *Proc. IEEE Int. Conf. on Automatic Face and Gesture Recognition*, Grenoble, France, Mar. 2000, pp. 446–453.