İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

DİNAMİK ALGILAYICI ÖĞRENME ALGORİTMASI İLE KENAR SAPTAMANIN ÖĞRENİLMESİ

YÜKSEK LİSANS TEZİ

Müh. Filiz YOSMA TAŞKIN

Tezin Enstitüye Verildiği Tarih : 12 Haziran 1995
Tezin Savunulduğu Tarih : 10 Temmuz 1995

Tez Danışmanı : Doç. Dr. Cüneyt GÜZELİŞ
Diğer Jüri Üyeleri : Prof. Dr. Cem GÖKNAR
Prof. Dr. Uğur ÇİLİNGİROĞLU

TEMMUZ 1995

ÖNSÖZ

Bu çalışmanın hazırlanmasındaki değerli katkıları ve yapıcı tutumu dolayısıyla hocam Doç. Dr. Cüneyt Güzeliş 'e teşekkür ederim.

Ayrıca bana çalışmam boyunca gösterdiği anlayış ve fedakarlıktan ötürü eşim Müh. D.Ali TAŞKIN'a, manevi desteklerini esirgemeyen aileme ve tüm dostlarıma da teşekkürü bir borç bilirim.

Temmuz 1995
Filiz YOSMA TAŞKIN

İÇİNDEKİLER

ÖZET

Bu tezde, Hücresel Yapay Sinir Ağları için geliştirilen bir eğiticili öğrenme kuralı olan Dinamik Algılayıcı Öğrenme Kuralı (Recurrent Perceptron Learning Algorithm- RPLA) ile genel olarak görüntü işlemenin özel olarak da kenar saptamanın öğrenilebileceği gösterilmiştir. Bu amaçla RPLA kullanan HYSA'lar için şablon öğrenme yazılımı CNNSLATW (Cellular Neural Network Supervised Learning Algorithm Tool for Windows) geliştirilerek bir kişisel bilgisayar ortamında benzetim düzeni elde edilmiştir. Bellek sorununu ortadan kaldırmak için CNNSLATW programı Visual C++ dilinde yazılmış ve böylece Windows ortamında çalışılarak yüksek çözünürlüklü görüntülerin işlenebilmesi sağlanmıştır. Yazılım RPLA ile görüntü işlemeye yönelik genel bir yazılımdır ve çeşitli uygulamalar (köşe saptama, kenar saptama, boşluk doldurma v.b.) için kullanılabilir. Ancak pek çok görüntü işleme uygulamasının ilk aşamasını teşkil ettiğinden bu çalışmada kenar saptama üzerinde durulmuştur.

RPLA ile kenar saptama konusunda daha önce yapılan çalışmalarda küçük boyutlu, siyah-beyaz sentetik görüntüler kullanılmıştır. RPLA ilk kez bu çalışmada 256 x 256 görüntü birimi (pixel) boyutunda, gri seviyeli, gerçel görüntülerde kenar saptama için kullanılmıştır. Bulunan sonuçlar ile görsel örüntü tanıma ve anlamaya yönelik ara bir gösterilim elde edilmesi sağlanmıştır.

SUMMARY

LEARNING OF EDGE DETECTION USING RECURRENT PERCEPTRON LEARNING ALGORITHM

Analog circuits have played a very important role in the development of modern electronic technology. Even in our digital computer area, analog circuits still dominate such fields as communications, power, automatic control, audio and video electronics because of their 'real-time' signal processing capabilities.

Conventional digital computation methods have run into a serious speed battleneck due to their serial nature. To overcome this problem, a new computation model, called 'artificial neural networks' has been proposed, which is based on some aspects of neurobiology and adapted to integrated circuits. The key features of artificial neural networks (ANN) are parallel and real-time processing.

Although ANN becomes an important research area in the recent 10 years, the first works in this subject have been started at the beginnig of 1940's by McCullach, Pits, Hebb and Rosenblatt [1]. McCullach abd Pits showed it is possible to implement some functions like AND and OR using ANN. Then it gathers the interests of many researchers from various disciplines as well as from industrial intitutions, and almost all branches of engineering.

But, since in these days silicon technology was not good anough to implement circuits with complexity needed for ANNs and digital computers known as an alternative for ANNs were developed to be operated at high speeds and their capacity and reliability are higher, from mid 60s to the beginning of 80s, related design and investment on ANNs were decreased.

The advanced in silicon technology along with the fact that digital computers which have imcomparably high processing speeds in arithmetic operations could not show the same performance on such areas image processing, pattern recognation where the problems with incomplete data and/or not well defined have attracted a great deal of scientists from various disciplines to the field again.

In the past three decades a number of neural network architectures have been developed. The architectures have been inspired both by the principles governing

biological neural systems and well-established theories of engineering and fundamental sciences. The best known neural network models are Grossberg's Adaptive Resonance Theory: ART, Widrow's ADAptive LINEar Element: ADALINE, Hopfield Network, Multilayer Perceptron, and Kohonen's Self Organizing Feature Map.

Most of the widely applied neural networks fall into two main classes: 1) memoryless neural networks and 2) dynamical neural networks. From a circuit theoretical point of view, the memoryless neural networks are non-linear resistive circuits, while the dynamical neural networks are non-linear R-L-C circuits. A memoryless neural network defines a non-linear transformation from the space of input signals into the space of output signals. Such networks have been successfully used in pattern recognition and several problems which can be defined as a non-linear transformation between two spaces. As in the Hopfield network and Cellular Neural Network, dynamical neural networks have usually been designed as dynamical systems where the inputs are set of some constant values and each trajectory approaches one of the stable equilibrium points depending upon the initial state. Some useful application of these networks includes image processing, pattern recognition and optimization.

From the learning point of view, neural network models can also be classified into three main classes:

1) Supervised Learning
2) Unsupervised Learning
3) Non-Learning

One of the most important contemporary interest is developing neural network models which not only provide new features in information processing, but also are more appropriate to hardware realizations. The Cellular Neural Network (CNN) model propased by Chua and Yang in 1988 is one of the remarkable developments in neural network theory and is a significiont contribution to the development of a variaty of image processing applications.

The basic circuit unit of Cellular Neural Network is called a cell. It contains linear and nonlinear circuit element, which typically are linear capacitors, linear resistors, linear and nonlinear controlled sources, and independent sources. Any cell in a CNN is connected only to its neighbor cells. The neighborhood defined by following metric:

$$d(i,j;\hat{i},\hat{j}) = \max\left\{\left|i-\hat{i}\right|, \left|j-\hat{j}\right|\right\}$$

Where (i,j) is the vector of integers indexing the cell C(i,j) in the i *th* row j *th* column of the 2-dimensional array. The system of equations describing a CNN with the neighborhood size of one is given in (1)-(2).

$$\dot{x}_{i,j} = -A x_{i,j} + \sum_{m,n \in -1,0,1} w_{m,n}\, y_{i+m,j+n} + \sum_{m,n \in -1,0,1} z_{m,n}\, u_{i+m,j+n} + I \qquad (1)$$

$$y_{i,j} = f(x_{i,j}) = \frac{1}{2}\left|x_{i,j} + 1\right| - \left|x_{i,j} - 1\right| \qquad (2)$$

Where, A, I, $w_{p,l}$ and $z_{p,l} \in R$ are constants.

Theoretically, the network structure of CNN can be defined of any dimension, but in this thesis focuses the attention on one of the image processing problems, namely edge detection, so CNN is defined to be a single-layer, first order network made of regularly spaced cells inter connected so as to form a two dimensional grid.

A CNN is completely stable if the feedback connection weights $w_{p,l}$ are symmetric. Throughout the thesis, the input connection weights $z_{p,l}$ are chosen symmetric for reducing computational costs while the feedback connection weights $w_{p,l}$ are chosen symmetric for ensuring the complete stability, i.e.,

$$w_{-1,-1} = w_{1,1} \overset{\text{def}}{=} a_1, \quad w_{-1,0} = w_{1,0} \overset{\text{def}}{=} a_2, \quad w_{-1,1} = w_{1,-1} \overset{\text{def}}{=} a_3,$$

$$w_{0,-1} = w_{0,1} \overset{\text{def}}{=} a_4, \quad w_{0,0} \overset{\text{def}}{=} a_5;$$

$$z_{-1,-1} = z_{1,1} \overset{\text{def}}{=} b_1, \; z_{-1,0} = z_{1,0} \overset{\text{def}}{=} b_2, \; z_{-1,1} = z_{1,-1} \overset{\text{def}}{=} b_3, \; z_{0,-1} = z_{0,1} \overset{\text{def}}{=} b_4, \; z_{0,0} \overset{\text{def}}{=} b_5.$$

Digital image processing systems are designed to recover useful information from one or more images of a scene. To facilitate the analysis of image, image processing systems are considered to comprise three processing levels, which are commonly referred to as low level, indermadiate level, and high level. Edge detection is one of the most important parts of low level processing. This is obvious from human vision for which line drawing and cartoons provide sufficient information for object identification. By detecting edges, an image processing system (biological or digital) finds occluding countours of objects. The edge detection process serves simplify the analysis of images by drastically reducing amount of data to be processed, while at the same time serving useful structural information about object boundaries.

In digital image processing basically two methods are used to detect edges in gray scale images. The first approximates the gradient and uses its length to determine edges. The second method, known as parallel edge detection and used in high speed digital image processing consists of three steps:

i) Correlation of the image with a local difference operator, yielding a measure for the strength of a gray scale discontinuity.

ii) Thresholding of the correlated image in order to classify edges.

iii) Elimination of noise or bridging of gaps.

A difficulty of this procedure is to determine on appropriate operator and a corresponding threshold in order to get a relevant image. This was often done heuristically, or in some cases following an analytical approach proposed by Canny. As an alternative, neural networks have been suggested. By using neural networks, appropriate training images can be designed and the operator and threshold can be learned.

The foregoing studies show that CNN is capable of carring out some complicated information processing tasks, and is specifically very successful in relatively low-level image processing applications such as edge detection, corner detection, hole filling and noise filtering. The main concern in CNN applications is to determine the inter connection weights which perform the desired image processing. Various methods exist as solutions of this problem. Learning algorithms are an important part of these solutions. Learning is performed through the modification of the interconnection weights according to an optimality criterion, depending on the samples chosen from the training set which is constituted of some pre-determined input signals and the corresponding desired outputs (if known) as in the supervised learning.

The supervised learning of the steady-state outputs in completely stable CNNs is a constrained optimization problem, where the objective function represents the error between the actual and desired outputs. The constraints are originated from the basic operation principles of CNN and are due to the desired design requirements such as the bipolarity of the steady-state outputs and the complete stability.

Recently, a supervised learning algorithm has been proposed for the completely stable CNNs. It resembles the Perceptron learning algorithm, and hence is called the Recurrent Perceptron Learning Algorithm (RPLA), since applied to a dynamical neural network, CNN. RPLA is developed for finding the interconnection weights and threshold of CNN to realize an input- steady-state output map described by a set of training samples.

In this thesis, RPLA is used for finding the optimal template coefficients and the threshold for the edge detection. To achieve this, the input-desired output pairs in the training set are applied to the CNN in an order. The images to be processed input to the

network from the external inputs and initial states. Desired output is chosen as the image which represents the edges of the image to be processed. After each iteration, weight vector w is updated according to the error which shows the difference between desired and actual outputs.

In this thesis, it is showed that edge detection can be learned using Recurrent Perceptron Learning Algorithm (RPLA) which is recently proposed for supervised learning of the steady-state outputs in completely stable Cellular Neural Networks. To achieve this work, a template learning software CNNSLATW (Celluar Neural Network Supervised Learning Algorithm Tool for Windows) is developed and the images which are gray scale, 256x256 pixel size are processed for the first time in the literature. To overcome memory problem CNNSLATW was written in Visual C++.

After giving a general overview of the ANN concept in Chapter 2, the CNN are presented in Chapter 3 along with a brief discussion of its up-to-date applications to image processing. Chapter 4 includes a brief information about learning in ANNs and the Recurrent Perceptron Algorithm (RPLA) proposed recently which is used for edge detection in this thesis. In chapter 5, after giving information about some conventional edge detection algorithms, the learning of edge detection using RPLA is presented. This chapter also includes the simulation results of CNNSLATW program.

BÖLÜM 1

GİRİŞ

Analog devreler , elektronik teknolojisinin geliştirilmesinde çok önemli bir rol oynamışlardır. Bugün elektronik dünyası büyük oranda sayısallaşmış olmasına rağmen , iletişim, güç, otomatik kontrol, ses ve görüntü alanlarında analog devreler hala önemli bir yere sahiptirler. Bunun başlıca sebebi analog devrelerin gerçel-zamanda işaret işleme (real-time signal processing) yeteneklerine sahip olmalarıdır.

Alışılagelmiş sayısal bilgi işleme yöntemlerinde seri yapıdan kaynaklanan bir hız problemi ile karşı karşıya kalınmıştır. Bu problemi çözebilmek için biyolojik sistemlerden yola çıkılarak 'yapay sinir ağları' adı verilen yeni bir model ortaya atılmıştır. Yapay sinir ağlarının en önemli özellikleri paralel ve gerçek zamanda bilgi işlemeye uygun olmalarıdır.

Yapay sinir ağları konusu son 10 yılda çok önem kazanmış olmasına rağmen bu konudaki ilk çalışmalar 1940'larda McCulloch, Pits, Hebb ve Rosenblatt tarafından yapılmıştır[1]. Önceleri kuramsal düzeyde süren çalışmalar, sonrasında bazı prototipler gerçeklenerek denenmiştir. Bunlardan en önemlileri Frank Rosenblatt tarafından gerçekleştirilen Mark I Perceptronu ve Bernard Widrow'un ADALINE ("ADAptive LINeer Element") 'ıdır. Daha sonraki yıllarda pek çok araştırmacı bu konuyla ilgilenmiş ve hatta bu konuda ticari faaliyette bulunan şirketler dahi olmuştur.

Geçen yıllar içinde birçok yapay sinir ağı modeli geliştirilmiştir [2],[3]. Bunların en tanınmışları; Grossberg'in Uyarlanmalı Rezonans Kuramı ("Adaptive Resonance Theory :ART") modelleri, Kohonen'in Öz-Örgütlenmeli Haritası ("Self-Organizing Feature Map"), Hopfield Ağı, Çok-Katmanlı Algılayıcı ("Multilayer Perceptron") ve Widrow'un Uyarlamalı Doğrusal Hücre ("ADAptive LINeer Element : ADALINE") modelleridir. Hücresel Yapay Sinir Ağı (HYSA)(Celluar Neural Network : CNN) modeli de bunlardan biridir. Geniş çapta tümleşik devre

gerçeklemeye uygun ve işaret işlemede yeni olanaklar sağlayan model 1988 yılında Chua ve Yang tarafından önerilmiştir [4],[5]. HYSA 'nın karmaşık bilgi işlemede ve özellikle kenar, köşe saptama, gürültü süzme gibi alt-düzeyli görüntü işleme konularında çok başarılı olduğu yapılan araştırmalarla gösterilmiştir [5],[6].

Bu tezde amaçlanan, HYSA için geliştirilmiş bir eğiticili öğrenme kuralı olan Dinamik Algılayıcı Öğrenme Kuralı (Recurrent Perceptron Learning Algorithm: RPLA) ile genel olarak görüntü işlemenin özel olarak da kenar saptamanın öğrenilebileceğinin gösterilmesidir. Bu amaçla RPLA öğrenme kuralını kullanan CNNSLATW (Cellular Neural Network Supervised Learning Algorithm Tool for Windows) uygulama yazılımı geliştirilerek bir kişisel bilgisayar ortamında benzetim düzeni elde edilmiştir. RPLA belli kısıtlamalar altında bir hata fonksiyonunu enazlamaya çalışan bir eğiticili öğrenme kuralıdır. Kısıtlamalar, iki kutuplu çıkış özelliğinin (kalıcı durumda hücre çıkışının +1 ya da -1 değerlerinden birini alması) ve tam kararlılığın sağlanabilmesi için getirilmiştir. RPLA ile, verilen bir görüntü işleme görevini yerine getiren şablonlar bulunabilir ve böylece görüntü işleme öğrenilebilir. RPLA, tam kararlı HYSA'nın eğiticili öğrenmesi için geliştirilmiş bilinen en etkin öğrenme kuralıdır.

Bölüm 2'de yapay sinir ağları tanıtılmıştır. Tezde temel alınan hücresel yapay sinir ağı modeli ve uygulama alanları hakkında bilgi Bölüm 3'te verilmiştir. Bölüm 4'te yapay sinir ağlarında öğrenme hakkında bilgi verildikten sonra iki öğrenme kuralı, Algılayıcı ve Dinamik Algılayıcı Öğrenme kuralları tanıtılmıştır. Bölüm 5 'te önce klasik kenar saptama yöntemleri tanıtılmış, daha sonra Dinamik Algılayıcı Öğrenme Kuralı (RPLA) ile kenar saptamanın öğrenilmesi açıklanmıştır. Ayrıca bu bölümde RPLA ile kenar saptamanın öğrenilebileceğini göstermek amacıyla yazılmış CNNSLATW programı ile elde edilen benzetimler sunulmuştur. Bölüm 6'da CNNSLATW programının yapısı ve kullanılışı hakkında bilgi verilmiştir.

BÖLÜM 2

YAPAY SİNİR AĞLARI

1940'ların başında McCulloch, Rosenblatt, Hebb ve Pits YSA'larla ilgili ilk çalışmaları başlattılar. McCulloch ve Pits 1943 yılında geliştirdikleri YSA ile VE, VEYA gibi işlemlerin yapılabileceğini gösterdiler. Bundan sonra YSA farklı bilim dallarından pek çok araştırmacının birlikte çalıştığı bir bilim dalı haline geldi.

Ancak o yıllarda tümleşik devre teknolojisinin YSA'ların gerektirdiği karmaşıklıkta devre üretecek nitelikte olmaması ve YSA'ların alternatifi olan ve seri olarak çalışan hızlı birimlerden oluşan sayısal bilgisayarların aritmetik işlemlerde yüksek hız, kapasite ve güvenilirlik sağlanması 60'ların ortalarından 80'lerin başına kadar YSA'larla ilgili çalışmaların geri plana itilmesine neden olmuştur.

80'li yılların başlarında tümleşik devre teknolojisindeki gelişmeler ve sayısal bilgisayarların ses ve görüntü algılama, örüntü tanıma (pattern recognation) gibi bozulmuş ve tam tanımlı olmayan, yüksek veri yoğunluğuna sahip bilgileri işlemede yetersiz kaldığının anlaşılması YSA'ların tekrar güncellik kazanmasına neden olmuştur.

YSA ile ilgili önemli çalışmalar Tablo 2.1 'de verilmiştir [7].

YSA'larla ilgili çalışmalar üç ana grupta toplanabilir:

i - Teorik çalışmalar
ii - Uygulamalar
iii - Gerçeklemeler

Birinci grup çalışmalar, yeni ağ modellerinin bulunması, yeni veya var olan ağ mimarilerinin matematiksel olarak modellenmesi, analizi ve sentezi, özellikle de yeni öğrenme algoritmalarının geliştirilmesini kapsamaktadır.

Tablo 2.1.a Önemli 15 YSA Modeli

YSA 'nın Adı	Geliştiren	Yılı	Gerçekleştirilen	Sınırlar	Yorumlar
"Perceptron"	Rosenblatt (Cornell Univ.)	1957	Daktilo yazımı harflerin tanınması	Karmaşık harfleri tanıyamaz, dönüşümlere duyarlı	En eski YSA, donanım olarak gerçeklendi
"Madaline"	Widrow (Stanford Univ.)	1960-1962	Uyarlamalı kanal denkleyici	Giriş-çıkış arasında doğrusal ilişki varsayar	20 yıldır ticari kullanım, güçlü öğrenme kuralı
"Avalanche"	Grossberg (Boston Univ.)	1967	Ses tanıma,robot kolu kontrolü	Hız ve devinimi değiştir-menin basit yolu yok	Bütün işleri yapan bir ağ yok
"Cerebellatron"	Maar, Albus, Pellionez	1969-1982	Robot kolu ve motor hareketi kontrolü	Karmaşık kontrol girişi	"Avalanche" benzer
"Backpropagation"	Werbos, Parker, Rumelhart (Stanford Univ.)	1974-1985	Metinden sese dönüştürücü , robot kolu kontrolü	Sadece eğiticili öğrenme uygulanabilir	En yaygın ağ, iyi çalışıyor, öğrenmesi basit
"Brain state in a box"	Anderson (Brown Univ.)	1977	Veri tabanından bilgi çıkartılması	Tek-atış türü karar verme	Parçalardan bütünü elde edebilmektedir
"Neocognitron"	Fukushima (HNK Lab.)	1978-1984	Elyazımı harflerin tanınması	Çok sayıda hücre ve bağlantı	Geliştirilen en karmaşık devre, dönüşüme duyarsız
"Adaptive Resonance Theory"	Carpenter, Grossberg (Boston Univ.)	1978-1986	Örüntü tanıma (radar, sonar)	Geçişim, bozulmaya duyarlı	Karmaşık, kısıtlı bir alana uygulandı

Tablo 2.1.b Önemli 15 YSA Modeli

YSA 'nın Adı	Geliştiren	Yılı	Gerçekleştirilen	Sınırlar	Yorumlar
"Self-Organazing Feature Map"	Kohonen , (Helsinki Univ.)	1980	Bir geometrik bölgenin diğerine dönüştürülmesi	Uzun öğrenme zamanı	Sayısal Hava devinim akış hesaplamalarında etkin
"Hopfield"	Hopfield (AT&T Bell Lab.)	1982	Parçalardan tüm görüntüyü veya veriyi elde etme	Öğrenmez	Geniş çapta gerçeklenebilir
"Bidirectional Associative Memory"	Kosko (Southern California)	1985	İçerik sıralı bağlaşımsal bellek	Düşük saklama yoğunluğu, veri düzgün kodlanmalı	En kolay öğrenen ağ, parçalardan bütünü elde edebiliyor
"Boltzman and Chauchy Machine"	Hinton, Sejnıvski, Szu	1985-1986	Görüntü, sonar ve radar için örüntü tanımı	Uzun eğitim zamanı, gürültü	Mutlak en az noktayı bulabiliyor
"Counterpropagation "	H.Nielsen (Neurocomputer Corp.)	1986	Görüntü sıkıştırma ve istatiksel analiz	Çok sayıda hücre ve bağlantı	"Backpropagation"dan basit fakat güçsüz
"Cellular Neural Network" Hücresel YSA	Chua, Yang (Berkeley Univ.)	1988	Görüntü işleme	İki-Kutuplu çıkış	Geniş çapta tümleşik devre gerçeklemeye uygun
"Generalized Cellular Neural Network"	C.Güzeliş (İ.T.Ü), Chua (Berkeley Univ.)	1991	Karmaşık, kaotik bilgi ve özellikle lineer olmayan işaret işleme	Hücre ve ağ yapısı karmaşık olmasından dolayı gerçekleme zorluğu	Bilinen en genel YSA modeli, evrensel YSA modeli olarak alınabilir

İkinci grup çalışmalar YSA'ların çeşitli problemlerin çözümü için kullanılmasını içermektedir. Çok değişik bilim dallarından ve hatta endüstriyel kuruluşlardan pek çok araştırmacı bu konuda çalışmaktadır.

YSA'ların tümleşik devre olarak gerçeklenmesi üçüncü grup çalışmaları oluşturmaktadır.

2.1 Yapay Sinir Ağlarının Bir Tanımı

Kohonen YSA'yı birbirine yoğun bir şekilde paralel olarak bağlanmış, basit (genellikle adaptif) elemanlardan oluşmuş ve gerçek dünyadaki nesnelerle biyolojik sinir dizgesinin yaptığının aynı şekilde etkileşmek üzere hiyerarşik olarak düzenlenmiş bir ağ olarak tanımlamaktadır [7].

C.Güzeliş'in devre kuramsal bir bakış açısı ile yaptığı YSA tanımında ise "YSA, ii)-viii)'te tanımlanmış hücre denilen alt devrelerin , i), ix)-xii)'de tanımlı olan bir bağlantı geometrisi ile birbirlerine bağlanmasından oluşmuş bir elektrik devresidir " denilmektedir [2].

Bugüne kadar verilen çeşitli YSA modellerinin ortak özellikleri aşağıda sıralanmıştır [2].

i) Ağ, ii)-viii) 'de verilen, hücre denilen yapı taşlarının birbirlerine ix)-xii) 'de belirtilen bir geometri ile bağlanmasından oluşan bir sistemdir.

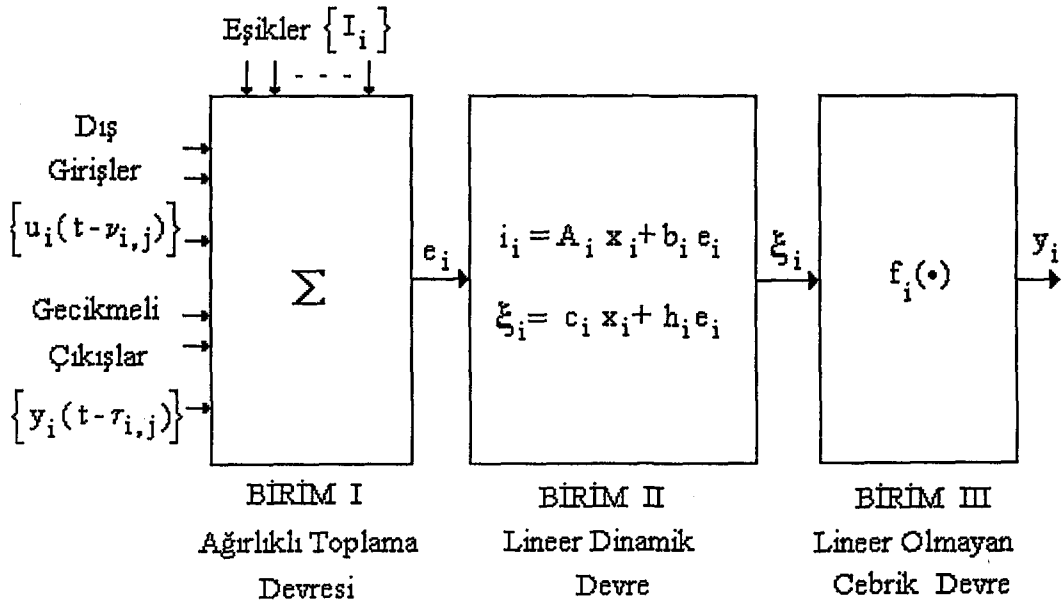
ii) Hücreler, genelde çok girişli ve tek çıkışlı, yüksek dereceden lineer olmayan dinamik alt devrelerdir.

iii) Hücrelerin yapısı Şekil 2.1 'de gösterildiği biçimdedir. (Genelde lineer toplama birimi yerine lineer olmayan bir toplama birimi vardır.)

iv) Tek olan hücre çıkış işareti, çoğullanarak dışarıya çıkış olarak alınabilir, veya diğer hücrelere giriş olacak şekilde (genelde gecikmeler ile) kullanılabilir.

v) Bir hücre; dış girişleri (gecikme ile ulaşmış olabilir), komşu hücre çıkışlarını, ve bir eşik değerini giriş olarak kabul eder.

vi) Hücrenin lineer dinamik birimi herhangi bir dereceden olabilir.



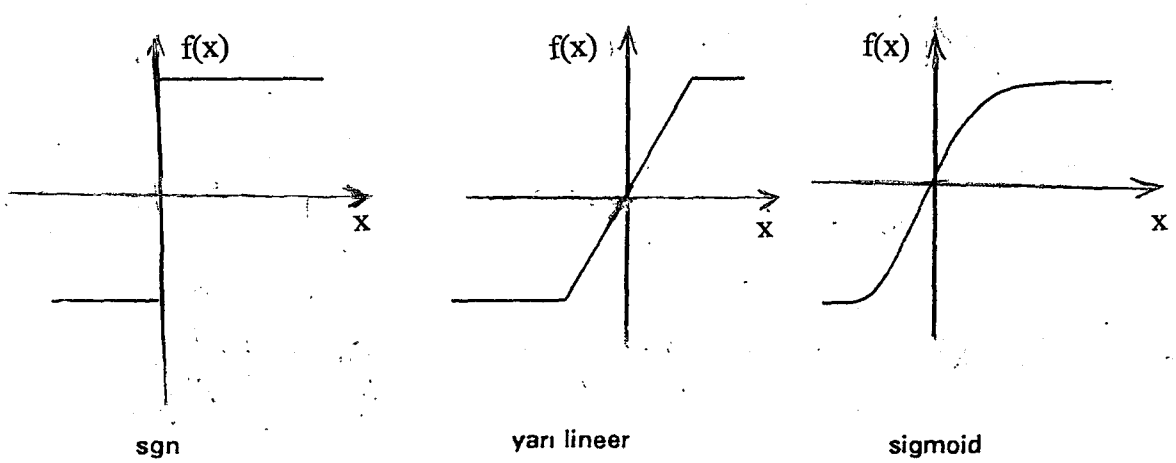
Şekil 2.1 Hücrelerin Genel Yapısı

vii) Hücre çıkışındaki lineer olmayan cebrik işlev 1-giriş 1-çıkışlıdır ve özel olarak lineer olabilen herhangi bir işlevdir.

viii) Hücre parametreleri hücreden hücreye değişebilir.

ix) Ağ, genelde çok boyutlu katmanların ileri veya geribeslemeli olacak şekilde bağlanmasından oluşmuştur.

x) Ağdaki herbir katman, hücrelerin çok boyutlu bir dizisidir.



Şekil 2.2 Genelde kullanılan eşik fonksiyonları.

xi) Katman içinde, herbir hücre komşularına (belirli bir metrik uyarınca) aynı şekilde bağlıdır. Böylece katman içi bağlantı geometrisi düzgündür.

xii) Katmanlar arası bağlantı bir metrik ile tanımlı düzgün bir geometriye sahiptir.

xiii) Ağda yalnızca bağlantı ağırlıkları değişebilir.

xiv) Bağlantı ağırlıkları; ya eğiticili bir öğrenme kuralı ile ya eğiticisiz bir öğrenme kuralı ile değiştirilir ya da önceden tasarlanan değerlerde sabit tutulur.

xv) Eğiticili bir öğrenme kuralı (eğer varsa); giriş-(istenen) çıkış örnek çiftlerinin alındığı, bilinmeyen bir giriş işlevine bu örnekler yardımıyla yaklaşılmasını sağlayan bir bağlantı ağırlık değişim kuralıdır.

xvi) Eğiticisiz öğrenme kuralı (eğer varsa); giriş örneklerinin kümelendirilmesini sağlayan bir bağlantı ağırlık değişim kuralıdır.

2.2 Yapay Sinir Ağlarının Üstünlükleri

YSA modelleri alışılagelmiş bilgi işleme yöntemlerine alternatif olarak ortaya atılmışlardır. Bu modellerin üstünlükleri aşağıdaki gibi özetlenebilir [1],[8]:

Paralellik :

Bir YSA'da her bir hücre kendi başına bir işlem elemanıdır. Hücreler arası ilişki sadece bir hücrenin çıkışının diğer bir hücreye giriş olarak gitmesinden ibarettir. Bu özellik YSA'ların paralel çalışmasına olanak sağlamaktadır. Zaten YSA'ların seri bilgi işleme yöntemlerine göre en büyük üstünlüğü de budur. Seri sistemlerde bir birimin yavaş olması bütün sistemin hızını düşürürken, paralel sistemlerde bu etki çok azdır.

Hata Toleransı :

Seri sistemlerde bir birimde meydana gelen hata bütün sistemin hatalı çalışmasına neden olmaktadır. Oysa YSA paralel çalışan bir sistem olduğundan bir kaç hücrede oluşacak problem (hatalı çalışma veya bozulma) tüm sistemin

çalışmasında önemli bir hataya neden olmayacak, sadece bu hücrelerin ağırlıkları oranında etkilenme olacaktır.

Yerel Bilgi İşleme :

YSA'larda bilgi işleme işlevi hücrelere dağıtılmıştır. Her bir hücre problemin bir parçasını çözmeye çalışmakta ve bunu yaparken de sadece bağlı olduğu hücrelerdeki bilgiye erişmektedir. Her bir hücrenin tek başına yaptığı işlem çok basit olmasına rağmen görev paylaşımı sayesinde çok karmaşık ve zor problemler çözülebilmektedir. Gerçekte beynin çalışmasının da bu biçimde olduğu düşünülmektedir.

Öğrenebilirlik :

YSA'ların en önemli özelliklerinden biri de öğrenebilme, yetenekleridir. Alışılagelmiş veri işleme yöntemlerinin çoğu programlama yoluyla hesaplamaya dayanmaktadır. Bu yöntemlerde bir problemi çözebilmek için o probleme yönelik algoritma geliştirilmekte ve veriden bağımsız olarak programlanmaktadır. Oysa YSA'lar bir problemi çözerken veriye bağlı olarak öğrenebilmektedirler. Bir YSA'nın öğrenmesi belirli bir amacı daha iyi yapabilmek için çevreden ve varsa eğiticiden aldığı girişlere göre iç yapısını uyarlayarak giriş-çıkış işlevini değiştirmesi olarak tanımlanır [7]. Öğrenme sırasında iç yapıdaki değişimler yeni bağlantılar oluşturmak, varolan bağlantıların kaybolması, varolan bağlantıların ağırlıklarının değişmesi ve eşik değerinin değişmesi şeklinde olabilmektedir. Öğrenme yeteneği, tam tanımlı olmayan problemlerin YSA'larla çözülebilmesine olanak sağlamıştır.

Gerçekleme Kolaylığı :

YSA'ların basit işlemler gerçekleyen, aynı yapıya sahip hücrelerden oluşması ve hücreler arası bağlantıların düzgün olması fiziksel olarak gerçeklenmelerini kolaylaştırmıştır.

BÖLÜM 3

HÜCRESEL YAPAY SİNİR AĞLARI

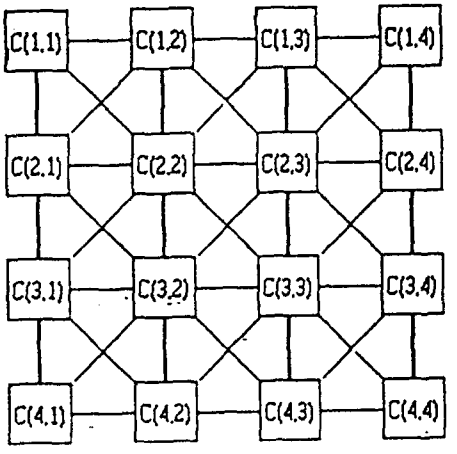
YSA'larla ilgili çalışmalara bakıldığında pek çok farklı ağ modelinin geliştirildiği görülmektedir. Model sayısının fazlalığının sebebi her uygulama alanı için diğerlerine üstün gelen bir ağ modeli olmamasıdır. İyi sonuçlar elde etmek ancak çözümü istenen problemin doğasına uygun bir model seçilmesiyle mümkündür.

Varolan modellere göre donanım olarak gerçeklemeye daha uygun ve işaret işlemede yeni olanaklar tanıyan YSA modellerinin geliştirilmesi günümüzde araştırmacıların önemli bir ilgi alanını oluşturmaktadır. 1988 yılında Chua ve Yang tarafından önerilen Hücresel Yapay Sinir Ağı (HYSA) modeli [4],[5] bu alanda atılmış bir adımdır. HYSA, gerçek zamanda işaret işleyen büyük çaplı doğrusal olmayan bir devredir.

HYSA'lar iki dünyanın en iyi özelliklerini yapılarında birleştirmişlerdir. Sürekli zaman özelliği gerçek zamanda bilgi işlemeye olanak sağlarken yerel bağlantı özelliği sayesinde geniş çaplı tümleşik devre üretimi için uygun hale gelmektedir. HYSA'lar yüksek hızlı paralel işaret işlemeye son derece uygundurlar.

3.1 Hücresel Yapay Sinir Ağının Mimarisi

Hücresel yapay sinir ağının temel yapı taşına hücre denir. Hücreler doğrusal ve doğrusal olamayan devre elemanlarından oluşmaktadır. Her bir hücre sadece yakın komşuluğunda bulunan hücrelere bağlıdır. Komşu olan hücreler birbirlerini doğrudan etkilerler. Komşu olmayan hücreler ise başlangıç durumdan kararlı duruma giderken zamanda yayılım ile birbirilerini etkilemektedirler. Kuramsal olarak, HYSA istenilen herhangi bir boyutta tanımlanabilir. Ancak bu çalışmada görüntü işleme ile ilgilenildiğinden HYSA yerel olarak birbirlerine bağlı, 2-boyutlu bir ızgara üzerinde düzgün olarak yinelenmiş hücrelerden oluşmuş tek katmanlı bir ağ olarak alınmıştır.



Şekil 3.1 4x4 2-boyutlu HYSA. Kareler hücreleri, çizgiler ise hücreler arası bağlantıları göstermektedir.

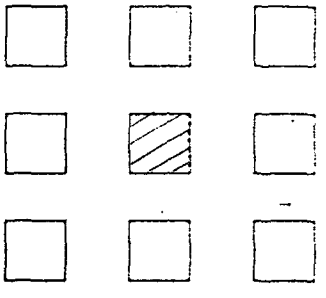
İki boyutlu MxN yani M satır N sütunluk bir HYSA 'da , i 'inci satır j 'inci sütundaki bir hücre C (i, j) ile gösterilir.

Tanım 1: r-komşuluğu

Bir hücresel sinir ağında bir C (i, j) hücresinin r komşuluğu, r bir tamsayı olmak üzere ;

$$N_r(i,j) = \left\{ C(k,l) \middle| \max\left\{ |k-i|, |l,j| \right\} \le r, 1 \le k \le M; 1 \le 1 \le N \right\} \qquad (3.1)$$

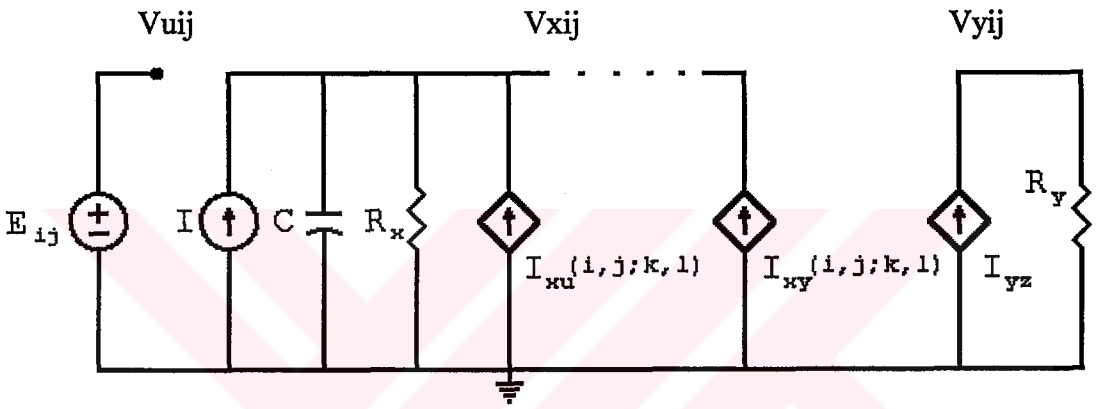
şeklinde tanımlanır. Şekil 3.2 r=1 durumunu göstermektedir. r=1 komşuluğu "3x3 komşuluğu" olarak da adlandırılır.



Şekil 3.2 Bir C(i,j) hücresinin r=1 komşuluğu.

Bir hücrenin yapısı Şekil 2.1 'deki gibi üç temel blok ile modellenebilir [2]. Birinci blok, yani giriş bloğu, girişlerin, komşu hücrelerden gelen çıkışların ve eşik değerinin ağırlıklı toplamını oluşturan bölümdür. İkinci blok, 1-giriş 1- çıkışlı doğrusal dinamik bir devredir. Hücrenin tek doğrusal olmayan bölümü üçüncü, yani çıkış bloğudur. İkinci bloğun çıkışını alıp ve doğrusal olmayan bir fonksiyondan geçirir.

HYSA'da bir hücrenin devre eş değeri Şekil 3.3'de verilmiştir. Burada u,x ve y indisleri sırasıyla girişi, durumu ve çıkışı belirtmektedir.



Şekil 3.3 Bir C (i, j) Hücresinin İç Yapısı.

Düğüm gerilimi V_{uij}, C(i,j) 'nin girişi olarak adlandırılır ve genliği küçük veya eşit 1 olan bir sabit kabul edilir. V_{xij} düğüm gerilimi ise C(i,j) 'nin durumu olarak isimlendirilir ve onun da başlangıç durumundaki genliği 1'e eşit veya küçüktür. V_{yij} düğüm gerilimi ise çıkıştır. İkinci blok, doğrusal dinamik birim bir bağımsız gerilim kaynağı Eij, bir bağımsız akım kaynağı I, bir doğrusal kapasite C, iki doğrusal direnç Rx, Ry ve m komşu sayısı olmak üzere en fazla 2m tane doğrusal gerilim kontrollü akım kaynağından oluşmaktadır. Doğrusal gerilim kontrollü akım kaynakları giriş gerilimi V_{ukl} ve komşu hücrelerin çıkış gerilimleri V_{ykl} ile kontrol edilmektedirler. $I_{xy}(i,j;k,l)$ ve $I_{xu}(i,j;k,l)$ kaynaklarının karakteristikleri $C(k,l) \in N_r(i,j)$ için

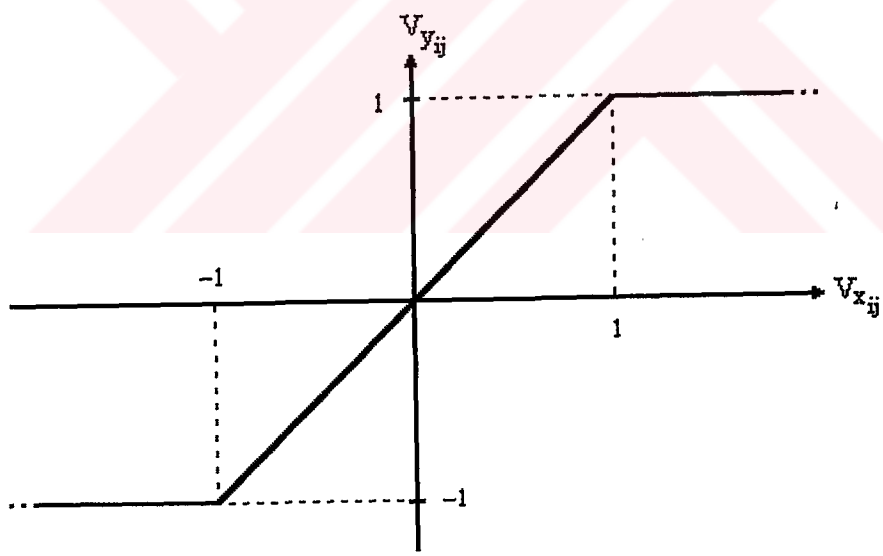
$$I_{xy}(i,j;k,l) = A(i,j;k,l)V_{ykl}$$

$$I_{xu}(i,j;k,l) = B(i,j;k,l)V_{ukl}$$

şeklindedir. Burada A(i,j;k,l) ve B(i,j;k,l) katsayıları C(i,j) ve C(k,l) hücreleri arasındaki bağlantı ağırlık katsayılarıdır.

HYSA hem çıkıştan geri besleme hem de giriş kontrol mekanizmalarına sahiptir. A(i,j;k,l) katsayısı çıkıştan geri beslemeyi , B(i,j;k,l) katsayısı ise giriş kontrolünü modeller, bu nedenle A(i,j;k,l)'ye geri besleme ağırlık katsayısı ve B(i,j;k,l)'ye de giriş ağırlık katsayısı denir.

Hücrenin son bloğu olan çıkış katı, parça parça doğrusal olan gerilim kontrollü bir akım kaynağından oluşur. $I_{yz} = (1/R_y)f(V_{xij})$ denkleminde f(•) karakteristiği Şekil 3.4'te verilmiştir.



Şekil 3.4 f(•) Karakteristiği.

Bir C (i, j) hücresi aşağıdaki devre denklemleriyle tanımlanır:

Durum denklemi:

$$C\frac{dV_{xij}(t)}{dt} = -\frac{1}{R_x}V_{xij}(t) + \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)\,V_{ykl}(t) + \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)\,V_{ukl}(t) + I$$

$$, \quad 1 \le i \le M \; ; \; 1 \le j \le N \qquad (3.2)$$

Çıkış denklemi:

$$V_{yij}(t) = \frac{1}{2}\left(\left|V_{xij}(t)+1\right| - \left|V_{xij}(t)-1\right|\right) \quad , \quad 1 \le i \le M \; ; \; 1 \le j \le N \qquad (3.3)$$

Giriş denklemi:

$$V_{uij} = E_{ij} \quad , \quad 1 \le i \le M \; ; \; 1 \le j \le N \qquad (3.4)$$

Sınır şartları:

$$\left|V_{xij}(0)\right| \le 1 \quad , \qquad 1 \le i \le M \; ; \; 1 \le j \le N \qquad (3.5)$$

$$\left|V_{uij}\right| \le 1 \quad , \qquad 1 \le i \le M \; ; \; 1 \le j \le N \qquad (3.6)$$

Simetri varsayımı:

$$A(i,j;k,l) = A(k,l;i,j) \quad , \qquad 1 \le i \le M \; ; \; 1 \le j \le N \qquad (3.7)$$

Parametre varsayımları:

$$C > 0 \quad , \quad R_x > 0 \qquad (3.8)$$

3.2 Hücresel Yapay Sinir Ağının Özellikleri

3.2.1 Tam Kararlılık

HYSA (3.5) ve(3.6) ile verilen sınır şartları ve (3.7)'de belirtilen simetri varsayımı altında tam kararlı bir ağdır. Yani zaman sonsuza giderken bütün yörüngeler denge noktalarından birine giderler [4].

3.2.2 Kararlı Hal Durum Değişkenleri ve Kararlı Hal Çıkışları

Geçici rejim sona erdikten sonra bir HYSA daima denge noktalarından birine gider. Yani zaman sonsuza giderken çıkış sabit bir değere erişir.

$$\lim_{t\to\infty} V_{xij}(t) = \text{constant}, \qquad 1\le i\le M,\ 1\le j\le N \qquad (3.9)$$

Eğer çıkıştan öz-geribesleme katsayısı A(i ,j; i ,j), durumdan öz-geribesleme katsayısı olan $1/CR_x$ 'den büyük seçilirse, mutlak değerce 1'den büyük iki tane kararlı denge noktası elde edilir.

$$\lim_{t\to\infty} V_{xij}(t) > 1, \qquad 1\le i\le M,\ 1\le j\le N \qquad (3.10)$$

Bu durumda Şekil 3.4 'deki çıkış karakteristiğinden görüleceği üzere çıkışlar mutlaka +1 veya -1 değerlerinden birini alacaktır, yani;

$$\lim_{t\to\infty} V_{yij}(t) = \pm 1, \qquad 1 \le i \le M,\ 1 \le j \le N \qquad (3.11)$$

Sonuç olarak HYSA, dış-girişlerin ve başlangıç koşullarının oluşturduğu sürekli uzaydan kararlı hal çıkışlarının ayrık uzayına doğrusal olmayan cebrik bir dönüşüm yapmaktadır. Bu dönüşüm

$$[-1,1]^{NxM} \times [-1,1]^{NxM} \xrightarrow{f} \{-1,1\}^{NxM} \qquad (3.12)$$

biçiminde ifade edilebilir.

3.2.3 Konumsal Değişmezlik

HYSA 'ların önemli bir alt kümesi konumsal-değişmez HYSA 'lardır [11]. Konumsal-değişmezlik

$$\begin{aligned} A(i, j ; k, l) &= A(k-i,\ l-j) \\ B(i, j ; k, l) &= B(k-i,\ l-j) \end{aligned} \qquad (3.13)$$

biçiminde ifade edilebilir. Burada i,j hücrenin yeri, A (i, j ; k, l) geri besleme katsayısı ve B (i, j ; k, l) de kontrol katsayısıdır. İfadeden görüleceği üzere tüm hücreler için A ve B katsayıları i,j'den yani hücrenin konumundan bağımsızdırlar. Bu nedenle bir şablon ile karakterize edilebilirler.

Konumsal-değişmez HYSA 'larda bağlantı ağırlık katsayıları matris olarak ifade edilebilirler. 1-komşuluklu bir HYSA 'da geribesleme bağlantı ağırlıkları ve giriş bağlantı ağırlıkları sırasıyla Şekil 3.5 'deki **A** ve **B** matrisleri biçiminde gösterilebilirler.

$$A=\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \quad \text{ve} \quad B=\begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix}$$

Şekil 3.5 1-komşuluk için geribesleme ve giriş bağlantı ağırlıklarının matrisel gösterilimi.

Burada

$$\begin{array}{lllllllll} a_1 & = & A(-1,-1) & a_2 & = & A(-1,0) & a_3 & = & A(-1,1) \\ a_4 & = & A(0,-1) & a_5 & = & A(0,0) & a_6 & = & A(0,1) \\ a_7 & = & A(1,-1) & a_8 & = & A(1,0) & a_9 & = & A(1,1) \end{array}$$

şeklindedir. B matrisi de benzer biçimde elde edilir. A matrisine geri besleme şablonu, B matrisine giriş şablonu ve I skalerine eşik şablonu denir.

HYSA 'yı tanımlayan (3.2) denklemi A, B ve I şablonları cinsinden

$$C\frac{dV_{xij}(t)}{dt} = -\frac{1}{R_x}V_{xij}(t) + A * V_{yij}(t) + B * V_{uij} + I \qquad (3.14)$$

biçiminde yazılabilir. Burada * sembolü konvolüsyonu ifade eder ve

$$T * V_{ij} = \sum_{C(k,l)\in N_1(i,j)} T(k-i,l-j)\, V_{k,l} \qquad (3.15)$$

şeklinde tanımlanır.

(3.7) 'deki simetri varsayımı, (3.13) 'deki konumsal-değişmezlik varsayımı ile birleştirildiğinde

$$A(k\text{-}i,\ l\text{-}j) = A(i\text{-}k,\ j\text{-}l) = A(|i\text{-}k|, |j\text{-}l|) \qquad (3.16)$$
$$B(k\text{-}i,\ l\text{-}j) = B(i\text{-}k,\ j\text{-}l) = B(|i\text{-}k|, |j\text{-}l|)$$

elde edilir. Dolayısıyla şablon katsayıları

$$a_1 = a_9\ ,\ a_2 = a_8\ ,\ a_3 = a_7\ ,\ a_4 = a_6 \qquad (3.17)$$
$$b_1 = b_9\ ,\ b_2 = b_8\ ,\ b_3 = b_7\ ,\ b_4 = b_6$$

şekline dönüşür.

3.3 Hücresel Yapay Sinir Ağının Görüntü İşleme Uygulamaları

HYSA uygulamaları görüntü işleme üzerinde yoğunlaşmıştır. Bunun nedeni HYSA 'nın 2-boyutlu ızgara yapısına sahip ve hücrelerinin yerel olarak birbirlerine bağlı olmalarından dolayı görüntünün sürekliliğinin sonucu olarak yerel bilginin korunabiliyor olmasıdır. Her bir görüntü elemanına (pixel) bir hücre karşı düşürülmektedir. Giriş görüntüsü ağa ya dış girişlerden ya başlangıç durumlarından ya da her ikisinden birden uygulanabilir. HYSA dış girişlerin ve başlangıç durumlarının oluşturduğu sürekli uzaydan karalı hal çıkışlarının ayrık uzayına bir dönüşüm tanımladığından giriş görüntüleri -1 ile +1 arasında herhangi bir değer alabilirken çıkış görüntüsü sadece -1 veya +1 'lerden oluşabilmektedir. Yani giriş görüntüsü renkli veya gri seviyeli bir görüntü olabilirken çıkış görüntüsü sadece iki seviyeli olabilmektedir. Bu iki seviye genellikle siyah (+1) ve beyaz (-1) olarak seçilir. Buradan anlaşılacağı üzere renkli veya gri seviyeli çıkış gerektiren görüntü işleme uygulamaları tek katmanlı HYSA ile gerçeklenememektedir.

HYSA 'ların görüntü işleme uygulamalarından bazıları kenar saptama, köşe saptama, örüntü tanıma, boşluk doldurma, Çin ve Japon karakterlerini tanıma, birleşik elemanların saptanması, hareketli ve duran cisimlerin saptanması, cisimlerin sayılması, gürültü süzme, görüntü inceltme, görüntü kalınlaştırma, gölgelendirme ve gölge saptamadır [5],[6].

BÖLÜM 4

YAPAY SİNİR AĞLARINDA ÖĞRENME VE DİNAMİK ALGILAYICI ÖĞRENME KURALI

Yapay Sinir Ağlarının üstünlüklerini anlatan bölümde de (Bölüm 2.2) ifade edildiği üzere öğrenebilirlik YSA 'ların çok önemli bir özelliğidir. Bu bölümde önce YSA 'larda öğrenme ile ilgili bilgi verilecek daha sonra HYSA için geliştirilmiş olan ve bu çalışmada kullanılan Dinamik Algılayıcı Öğrenme Kuralı (Recurrent Perceptron Learning Algorithm - RPLA) tanıtılacaktır.

4.1 Yapay Sinir Ağlarında Öğrenme

Genel olarak öğrenme, bir sistemin parametrelerini değiştirerek verilen giriş işaretlerine karşılık istenen çıkış işaretlerini üretmesini sağlayacak bir kendini uyarlama sürecidir.

Bir YSA 'nın öğrenmesi ise çevreden ve varsa eğiticiden aldığı girişler uyarınca iç yapısını uyarlayarak giriş çıkış işlevini değiştirmesi olarak tanımlanır [7].

YSA 'nın uyarlama sürecinde değiştirdiği parametreler hücreler arası bağlantıları tanımlayan bağlantı ağırlık katsayılarıdır. Ağırlık katsayılarındaki değişimler üç şekilde meydana gelir [7] :

1. Yeni bağlantıların oluşması,
2. Varolan bağlantıların kaybolması,
3. Varolan bağlantıların ağırlıklarının değişimi.

Ağırlık katsayılarının tek bir adımda değiştirildiği öğrenme kuralları olmasına rağmen, öğrenme genelde adım adım yapılan bir işlemdir. Öğrenme için YSA 'ya uygulanan girişler ve varsa bunlara karşı düşen istenen çıkışlar kümesine eğitim kümesi

denir. Her adımda eğitim kümesinden sırayla veya rastgele alınan örnekler ağa uygulanır ve adım sonunda belirli bir amaç ölçütü uyarınca bağlantı ağırlık katsayıları değiştirilir.

YSA 'larda iki tür öğrenme kuralı vardır:

1. Eğiticisiz öğrenme
2. Eğiticili öğrenme

4.1.1 Eğiticisiz Öğrenme

Eğiticiz öğrenmede istenen çıkış ağa verilmemektedir. Böylece ağ davranışını geliştirmek için kullanılabilecek bir hata bilgisi bulunamamaktadır. Ağın ürettiği çıkışın doğruluğu veya yanlışlığı hakkında herhangi bir bilgi olmadığından öğrenme daha önceden belirlenen bir amaç ölçütü uyarınca sadece girişlere bağlı olarak gerçekleştirilir. Bu tür öğrenme ancak sınırlı sayıda YSA modeline uygulanabilir.

4.1.2 Eğiticili Öğrenme

Eğiticili öğrenmede her bir giriş için ağın ürettiği çıkışın doğruluğu hakında eğitici tarafından ağa bilgi verilir. Eğiticinin verdiği bilginin tipine göre eğiticili öğrenme ikiye ayrılır. Birinci türde, mevcut girişe karşı düşen istenen çıkış eğitici tarafından ağa uygulanır. Ağın ürettiği çıkış ile istenen çıkış arasındaki fark belirli bir metriğe göre tanımlanır ve yapılan çıkış hatasını gösterir. Bu farkı enazlayacak biçimde ağırlık katsayıları değiştirilir. İkinci tür öğrenmede ise eğitici sadece üretilen çıkışın doğru veya yanlış olduğunu belirtir. Buna ödül-ceza türü eğiticili öğrenme denir.

4.2 Algılayıcı (Perceptron) Öğrenme Kuralı

Algılayıcı öğrenme kuralı, bir algılayıcıda (preceptron) istenen çıkışa karşı düşen bir bağlantı ağırlık katsayıları vektörü bulmaya çalışan eğiticili bir öğrenme kuralıdır. Algılayıcı, doğrusal cebirsel bir YSA 'dır. $x \in R^n$ giriş vektörü, $w \in R^n$ bağlantı ağırlık katsayıları vektörü ve $y \in R^n$ hücrenin çıkış vektörü olmak üzere algılayıcının giriş çıkış ilişkisi aşağıdaki gibidir.

$$y = w^T x \tag{4.1}$$

Algılayıcı öğrenme kuralı (4.2) 'deki fark denklemi ile tanımlanır.

$$w_i(n+1) = w_i(n) + \varepsilon \left[d_i(n) - \text{sgn}\left(w_i^T(n)\, x(n)\right)\right] x(n) \qquad (4.2)$$

Burada $w_i = \left[w_{i1}\ w_{i2} \ldots w_{in}\right]^T$ vektörü ağdaki i 'inci hücreye gelen n tane girişin bağlantı ağırlık katsayılarını, $x = \left[x_1\ x_2 \ldots x_n\right]^T$ vektörü ağa gelen n tane girişi, d_i i 'inci hücrenin istenen çıkışını ve ε da öğrenme oranını temsil etmektedir. Görüleceği üzere bu öğrenme kuralı ikili (binary) çıkışı olan hücrelere uygulanabilmektedir. Bu kurala göre, eğer ağın ürettiği çıkış istenen çıkıştan farklı ise bağlantı ağırlık katsayıları değiştirilmektedir.

4.3 Dinamik Algılayıcı Öğrenme Kuralı

Tam kararlı ve iki kutuplu HYSA 'da kararlı durum çıkışlarının eğiticili öğrenilmesi aslında kısıtlamalı bir enazlama problemidir. Kısıtlama ağın tam kararlı ve kararlı durum çıkışlarının iki kutuplu olmasından kaynaklanmaktadır.

HYSA 'nın eğiticili öğrenmesi için değişik öğrenme kuralları önerilmiştir. Kısa bir süre önce sunulan Dinamik Algılayıcı Öğrenme Kuralı (Recurrent Perceptron Learning Algorithm- RPLA) da bunlardan biridir [12]. Bu öğrenme kuralı bir önceki bölümde tanıtılan Algılayıcı Öğrenme Kuralı'nın (Perceptron Learning Algorithm) dinamik bir ağ olan HYSA 'na bir uyarlamasıdır.

4.3.1 Kullanılan Notasyon

HYSA 'yı tanımlayan eşitlikler (3.2) ve (3.3), 1-komşuluk için, $V_x \triangleq x$, $V_y \triangleq y$, $V_u \triangleq u$ alınarak aşağıdaki gibi düzenlenebilir.

$$\dot{x}_{i,j} = -A\, x_{i,j} + \sum_{m,n \in \{-1,0,1\}} w_{m,n}\, y_{i+m,j+n} + \sum_{m,n \in \{-1,0,1\}} z_{m,n}\, u_{i+m,j+n} + I \qquad (4.3)$$

$$y_{i,j} = f(x_{i,j}) = \frac{1}{2}\left\{\left|x_{i,j} + 1\right| - \left|x_{i,j} - 1\right|\right\} \qquad (4.4)$$

Bir HYSA 'nın tam kararlı olabilmesi için (Bölüm 3.2.1) 'den bilindiği üzere geribesleme bağlantı ağırlık katsayılarının ($w_{m,n}$) simetrik olması gerekmektedir. Geribesleme ve giriş bağlantı ağırlık katsayıları her adım sonunda değiştirilmektedir. Bu

öğrenme kuralında tam kararlılığı sağlamak için w 'lar ve hesaplamaları azaltmak için de z 'ler simetrik alınmıştır. Yani

$$w_{-1,-1} = w_{1,1} = a_1 \, , \, w_{-1,0} = w_{1,0} = a_2 \, , \, w_{-1,1} = w_{1,-1} = a_3 \, , \, w_{0,1} = w_{0,-1} = a_4 \, , \, w_{0,0} = a_5$$

$$z_{-1,-1} = z_{1,1} = b_1 \, , \, z_{-1,0} = z_{1,0} = b_2 \, , \, z_{-1,1} = z_{1,-1} = b_3 \, , \, z_{0,1} = z_{0,-1} = b_4 \, , \, z_{0,0} = b_5$$

şeklinde tanımlanmıştır.

Öğrenmeyi gerçekleştirmek için her adım sonunda değiştirilen bağlantı ağırlıkları vektörü de aşağıdaki gibi tanımlanmaktadır.

$$w = \left[a^T \; b^T \; I\right]^T = \left[a_1 \; a_2 \; a_3 \; a_4 \; a_5 \; b_1 \; b_2 \; b_3 \; b_4 \; b_5 \; I\right]^T \qquad (4.5)$$

Giriş vektörü $v = \left[v_u^T \; v_x^T\right]$ biçiminde ifade edilmektedir. Burada $v_u = \left[..., u_{i,j}, ...\right]^T$ dış girişleri ve $v_x = \left[..., x_{i,j}(0), ...\right]^T$ ise başlangıç durumlarını temsil etmektedir. $y(t) = \left[..., y_{i,j}(t), ...\right]^T$ ise belirli bir v giriş vektörü ve w bağlantı ağırlık katsayıları vektörü için tam kararlı bir çıkış vektörünü ifade etmektedir. Kararlı durumda bu vektör, kararlı durum çıkış vektörü olan $y(\infty)$ 'a dönüşmektedir. İstenen çıkış ise d ile ifade edilmektedir.

HYSA 'larda eğiticili öğrenmede enazlanmak istenen işlev ağın kararlı durum çıkışları ile istenen kararlı durum çıkışları arasındaki farkı gösteren hata fonksiyonudur. İstenen çıkışla ağın çıkışı ararsındaki Öklid uzaklığı metrik olarak alınırsa s 'inci giriş vektörü için hata $E^s(w)$

$$E^s(w) = \frac{1}{2}\sum_{i,j}(y_{i,j}^{\,s}(\infty) - d_{i,j}^{\,s})^2 \qquad (4.6)$$

şeklinde tanımlanır.

Enazlanmak istenen fonksiyon ise eğitim kümesindeki tüm giriş-çıkış çiftleri için yapılan hataların toplamıdır. O halde (4.6) 'dan toplam hata

$$E(w) = \sum_{s=1}^{N}\sum_{i,j} E^s(w) = \sum_{s=1}^{N}\sum_{i,j}(y^s_{i,j}(\infty) - d^s_{i,j})^2 \qquad (4.7)$$

biçiminde ifade edilebilir.

4.3.2 Öğrenme Kuralı

RPLA belli kısıtlamalar altında (4.9) eşitliğiyle verilen hata fonksiyonunu enazlamaya çalışan bir eğiticili öğrenme kuralıdır. Kısıtlamalar :

i) İki kutuplu çıkışların elde edilebilmesi için (3.2) ile verilen durum denkleminin normalize edilmiş hali, yani durumdan öz-geribesleme katsayısı $1/CR_x = 1$ için $a_5 \geq 1$ olması gerekmektedir. Bu nedenle bağlantı ağırlıklarının (4.8) ile verilen A kümesine izdüşümü alınmıştır.

$$w(n) \in A = \left\{ w(n) \in R^{11} \middle| a_5 > 1 \right\} \qquad (4.8)$$

ii. Tam kararlılığı sağlamak için geribesleme bağlantı ağırlık katsayıları (w) simetrik alınmıştır.

iii. Her adımda yapılması gereken işlem sayısını azalmak için giriş bağlantı ağırlık katsayıları (z) simetrik alınmıştır.

RPLA 'da enazlanmak istenen hata fonksiyonu aşağıdaki gibi tanımlanmıştır.

$$E(w) = \sum_{i,j} y_{i,j}(\infty)\,(y_{i,j}(\infty) - d_{i,j}) = \sum_{(i,j)\in D^+} y_{i,j}(\infty) - \sum_{(i,j)\in D^-} y_{i,j}(\infty) \qquad (4.9)$$

Burada, $D^+ = \left\{ (i,j) \middle| y_{i,j}(\infty) = -d_{i,j} = 1 \right\}$ yani, istenen çıkış -1 olmasına rağmen +1 kalıcı-durum çıkışı veren hücreler ve $D^- = \left\{ (i,j) \middle| y_{i,j}(\infty) = -d_{i,j} = -1 \right\}$ yani, istenen çıkış +1 olmasına rağmen -1 kalıcı-durum çıkışı veren hücreler kümesini ifade etmektedir. Görüleceği üzere hata, kalıcı durum çıkışı istenen çıkışla çakışmayan hücrelerin mutlak değerlerinin toplamıdır.

Hata fonksiyonunun hesaplanması için hatalı çıkış veren hücrelerin belirlenmesi gerekmektedir. Kararlı durumda $x_{i,j}(\infty) = 0$ olacağından (3.2) bağıntısı normalize halde (4.10) 'daki gibi yazılarak $x_{i,j}(\infty)$ elde edilmektedir.

$$x_{i,j}(\infty) = \sum_{m,n\in -1,0,1} w_{m,n}\, y_{i+m,j+n}(\infty) + \sum_{m,n\in -1,0,1} z_{m,n}\, u_{i+m,j+n} + I = \left[Y_{i,j}\right]^T w \qquad (4.10)$$

Bu denklemde $Y_{i,j}$ toplam girişi göstermektedir.

$$Y_{i,j} = \left[y_{i,j}\ u_{i,j}\ I\right]^T \qquad (4.11)$$

$$y_{i,j} = \left[y_{i-1,j-1}(\infty)+y_{i+1,j+1}(\infty) \quad y_{i-1,j}(\infty)+y_{i+1,j}(\infty)\right.$$
$$\left. y_{i-1,j+1}(\infty)+y_{i+1,j-1}(\infty) \quad y_{i,j+1}(\infty)+y_{i,j-1}(\infty) \quad y_{i,j}(\infty)\right] \qquad (4.12)$$

$$u_{i,j} = \left[u_{i-1,j-1}+u_{i+1,j+1} \quad u_{i-1,j}+u_{i+1,j} \quad u_{i-1,j+1}+u_{i+1,j-1} \quad u_{i,j+1}+u_{i,j-1} \quad u_{i,j}\right]^T \qquad (4.13)$$

(4.10) 'dan elde edilen $x_{i,j}(\infty)$ aşağıda verilen (4.14) bağıntısında yerine konularak kalıcı durum çıkışı elde edilmektedir.

$$y_{i,j}(\infty) = \frac{1}{2}\left\{\left|x_{i,j}(\infty) + 1\right| - \left|x_{i,j}(\infty) - 1\right|\right\} \qquad (4.14)$$

Kalıcı durum çıkışı belirlendikten sonra (4.9) 'da verilen E(w) hata fonksiyonu kolayca hesaplanabilmektedir.

Bağlantı ağırlık katsayılarının değişimi aşağıda verilmiştir.

$$w(n+1) = \left[w(n) - \varepsilon(n)\, Y(n)\right] \qquad (4.15)$$

Bu denklemde

$$Y(n) = \sum_{(i,j)\in D_l^+} Y_{i,j}(n) - \sum_{(i,j)\in D^-} Y_{i,j}(n) \qquad (4.16)$$

$$w(n) \in A = \{w(n) \in R^{11} \mid a_5 > 1\} \qquad (4.17)$$

ε (n) ise öğrenme oranıdır.

Eğer $w(n) \notin A$ ise $w(n) = K_n\, w(n)$ alınmaktadır.

$$K_n = \mu \frac{1}{a_5(n)}$$

şeklinde tanımlanmıştır. Burada μ sabit bir sayıdır ve genelde 1.5 alınmıştır.

BÖLÜM 5

KENAR SAPTAMANIN ÖĞRENİLMESİ

Görüntü işleme sistemleri görüntüyü uygun bir biçime dönüştürmek ve bu yeni görüntüden yararlı bilgileri elde etmek için tasarlanmışlardır. Bu sistemlerde görüntü analizi alt, orta ve yüksek-düzey olmak üzere üç işlem aşamasında gerçekleştirilmektedir. Alt-düzeyli görüntü işleme ile görüntünün bir ara gösterilimi elde edildikten sonra üst-düzey işlemler yapılarak istenen bilgilere ulaşılmaktadır.

Kenar saptama, sayısal görüntü işlemenin en önemli alt aşamalarından biridir. Pek çok görüntü işleme algoritmasında ilk adım olarak kullanılmaktadır. Örneğin, bir görüntü algılama sisteminde nesnelerin algılanmasında temel işlem görüntüyü, içinde bulunan farklı nesnelere ait bölgelere ayırmaktır. Kenar saptama algoritmaları bu bölütleme işleminin ilk basamağını oluşturur. İnsanın görme mekanizmasından da gözlendiği üzere çizgilerden oluşan taslaklar bir nesnenin tanımlanması için temel ayırdedici bilgiyi taşırlar. Biyolojik veya bilgisayar tabanlı görüntü işleme sistemleri kenarları saptamak yoluyla nesnenin hem derinliğini hem de şeklini bulurlar. Kenar saptama, işlenecek veri miktarını çok önemli oranda azaltarak görüntü analizini basitleştirirken aynı zamanda nesne sınırlarına ilişkin yararlı yapısal bilgiyi de korur [19].

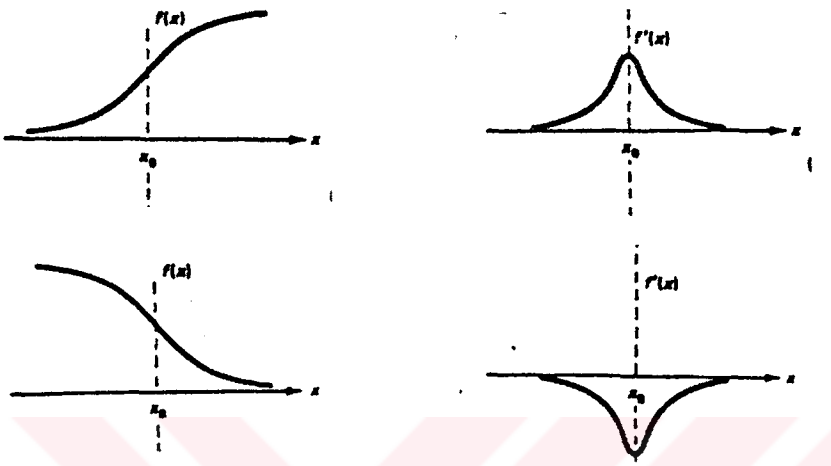
5.1 Kenar Saptamada Kullanılan Klasik Yöntemler

Literatürde kenar saptama için geliştirilmiş yöntemler temel olarak dört grupta incelenebilir [20], [21], [22]:

a) Türeve dayalı yöntemler
b) Şablon eşleştirme
c) Kenar uygunlaştırma
c) İstatiksel kenar saptama

5.1.1 Türeve Dayalı Yöntemler

Bir görüntüde kenar, gri-seviyeleri farklı iki bölge arasındaki geçiş olarak tanımlanabilir. Görüntünün gradyanı bu geçiş sınırında büyük değerler alır. Böylece gradyana (ilk türeve) dayalı kenar saptama yöntemlerinde görüntünün gradyanı alınıp uygun bir eşik değeri ile karşılaştırılarak kenarlar elde edilebilir.



(a) (b)

Şekil 5.1 (a) Gri seviyesi değişimleri. (b) Değişimlerin 1. türevleri

$\partial/\partial x$ ve $\partial/\partial y$ iki boyutlu görüntü fonksiyonunun x ve y doğrultularındaki değişim hızlarını göstermek üzere θ doğrultusundaki değişim

$$\nabla f = \frac{\partial f}{\partial x} \cos\theta + \frac{\partial f}{\partial y} \sin\theta \qquad (5.1)$$

biçiminde ifade edilir. Değişimin yönü

$$\theta = \arctan\left[\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right] \qquad (5.2)$$

ve büyüklüğü

$$|\nabla f| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2} \qquad (5.3)$$

denkleminden hesaplanır.

f(x,y)'nin gradyanı, (x,y) noktasında büyüklüğü ve yönü yukarıdaki eşitliklerle tanımlı bir vektördür. Bu nedenle eğer görüntü fonksiyonunun herhangi iki ortogonal doğrultudaki türevi biliniyorsa gradyan hesaplanabilir. Gradyana dayalı kenar saptayıcılar arasındaki temel fark operatörlerin kullandığı doğtultulardır. Kenar saptayıcılar görüntü fonksiyonunun bu doğrultulardaki bir boyutlu türevlerine yakınsarlar ve bu yakınsamaları birleştirerek gradyanın büyüklüğünü hesaplarlar.

Gradyan alma işlemi ayrık zamanda basit bir fark işlemine karşı düşer. İki boyutlu bir fonksiyonun x-doğrultusundaki türevi

$$\frac{\partial f}{\partial x} = f(x+n,y) - f(x,y) \qquad (5.4)$$

ve benzer şekilde y-doğrultusundaki türevi de

$$\frac{\partial f}{\partial y} = f(x,y+n) - f(x,y) \qquad (5.5)$$

biçiminde tanımlanır. Burada n bir tam sayıdır ve gradyanın, görüntü fonksiyonundaki değişimlere iyi yakınsamasını sağlayacak kadar küçük ve fonksiyonundaki küçük değişimlerin etkisini yenebilecek kadar büyük seçilmelidir.

Literatürde çeşitli gradyan operatörleri tanımlanmıştır. Roberts bir (x,y) görüntü noktasındaki gradyanın büyüklüğünü g(x,y), 2x2 komşuluğunda diagonal bir operatör kullanarak

$$g(x,y) \approx R(x,y) = \sqrt{[f(x,y)-f(x+1,y+1)]^2+[f(x,y+1)-f(x+1,y)]^2} \qquad (5.6)$$

biçiminde tanımlamıştır. R(x,y) genellikle Roberts çapraz operatörü olarak adlandırılır. Farklar, 'gradyan fonksiyonunun Roberts mutlak değer hesaplaması' olarak adlandırılan ve

$$g(x,y) \approx R(x,y) = |f(x,y)-f(x+1,y+1)|+|f(x,y+1)-f(x+1,y)| \qquad (5.7)$$

şeklinde ifade edilen hesaplanması daha kolay bir eşitlikle de bulunabilirler. Roberts operatörünün başka bir şekli de Rosenfeld ve Kak tarafından önerilmiştir. 'Roberts max' olarak adlandırılan ve

$$g(x,y) \approx R(x,y) = \text{Max}\left(|f(x,y)-f(x+1,y+1)|, |f(x,y+1)-f(x+1,y)|\right) \quad (5.8)$$

biçiminde tanımlanan bu operatör kenar doğrultusuna karşı duyarlı değildir. Eşit uzunlukta fakat farklı doğrultudaki kenarlara Roberts max operatörü uygulandığında elde edilen gradyanın büyüklüğündeki değişimin, çapraz operatörü uygulandığında elde edilenden daha az olduğu görülmüştür.

Roberts operatörünün temel problemi gürültüye karşı duyarlılığıdır. Bu sorunun üstesinden gelmek için Sobel ve Prewitt fark işlemini yerel ortalama ile birleştirmişlerdir. Sobel operatöründe f(x,y) merkezli 3x3 boyutlu bir alanda x-doğrultusundaki kısmî türev

$$\begin{aligned} S_x = &\left[f(x+1,y-1)+2f(x+1,y)+f(x+1,y+1)\right] \\ &-\left[f(x-1,y-1)+2f(x-1,y)+f(x-1,y+1)\right] \end{aligned} \quad (5.9)$$

biçiminde tanımlanır. Temel olarak bu işlem f(x,y) işlevinin her iki tarafında görüntünün gri seviyesinin ağılıklı ortalamasının farkını alır. Aynı şekilde y-doğrultusundaki türev de

$$\begin{aligned} S_y = &\left[f(x-1,y+1)+2f(x,y+1)+f(x+1,y+1)\right] \\ &-\left[f(x-1,y-1)+2f(x,y-1)+f(x+1,y-1)\right] \end{aligned} \quad (5.10)$$

biçiminde ifade edilir. Gradyan ise ya karelerin toplamının karekökü

$$g(x,y) \approx S = \sqrt{(S_x^2 + S_y^2)} \quad (5.11)$$

ya da mutlak değerlerin toplamı

$$g(x,y) \approx S = |S_x| + |S_y| \quad (5.12)$$

eşitliklerinden hesaplanır.

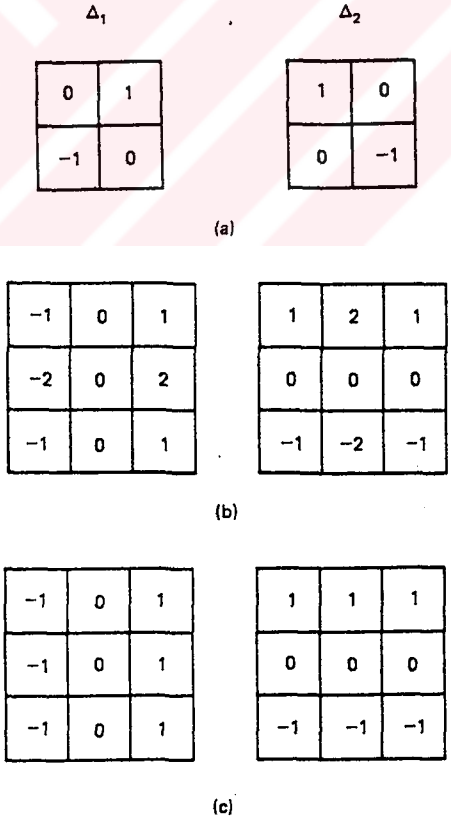
Benzer şekilde Prewitt operatöründe de kısmî türev

$$P_x = [f(x+1,y-1)+f(x+1,y)+f(x+1,y+1)] - [f(x-1,y-1)+f(x-1,y)+f(x-1,y+1)] \quad (5.13)$$

$$P_y = [f(x-1,y+1)+f(x,y+1)+f(x+1,y+1)] - [f(x-1,y-1)+f(x,y-1)+f(x+1,y-1)] \quad (5.14)$$

şeklinde tanımlanır ve gradyan yukarıdaki biçimde hesaplanır.

Genellikle farklar basit konvolüsyon maskeleri kullanılarak hesaplanırlar. x ve y kısmî türevleri için farklı birer maske kullanılır. Bunun sonucu iki kısmî türev görüntüsü oluşur. Daha sonra bu görüntüler, gradyanı hesaplamak için karelerin toplamının karekökü veya mutlak değerlerin toplamı yöntemlerinden biri kullanılarak nokta nokta birleştirilirler. Roberts, Sobel ve Prewitt operatörlerine ait maskeler Şekil 5.2'de verilmiştir.



Şekil 5.2 Konvolüsyon maskeleri (a) Roberts (b) Sobel (c) Prewitt

Gradyanın büyüklüğü hesaplandıktan sonra bu değer önceden belirlenmiş olan bir eşik değeri karşılaştırılarak burada bir kenar olup olmadığına karar verilir. Eğer gradyanın büyüklüğü eşik değerinden büyük ise bu görüntü elemanı (pixel) kenar olarak kabul edilir. Anlaşılacağı üzere eşik değerinin seçimi oldukça önemlidir. Özellikle gürültülü görüntülerde eşik değeri, kayıp geçerli kenarlarla gürültünün oluşturduğu yanlış kenarların birbirinden ayrılmasını sağlar.

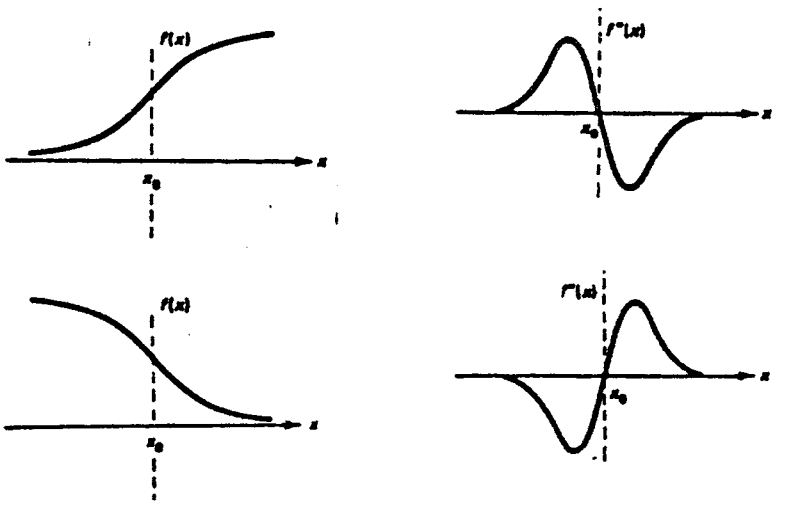
Şimdiye kadar anlatılan yöntemlerde birinci türeve dayalı yönlü operatörler kullanılmıştır. Diğer bir yaklaşım da ikinci türeve dayalı operatörler kullanmaktır. İki boyutlu görüntü fonksiyonunun ikinci türevine Laplasyen denir ve

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \qquad (5.15)$$

ifadesi ile tanımlanır. Laplasyenin fark denklemi olarak ifadesi

$$L(x,y) = f(x,y) - \tfrac{1}{4}\left[f(x,y+1) + f(x,y-1) + f(x+1,y) + f(x-1,y)\right] \qquad (5.16)$$

biçimindedir. Laplasyen gri seviyesindeki değişim noktalarında sıfır olur. Bu nedenle görüntünün Laplasyeni alınır ve sıfır geçiş noktaları izole edilirse kenar noktaları belirlenmiş olur. Bu yöntemin başlıca sorunu gürültüye karşı son derece duyarlı olmasıdır.



(a) (b)

Şekil 5.3 (a) Gri seviyesi değişimleri. (b) Değişimlerin 2. türevleri

Laplasyenin kenar saptamaya uygulanması konusunda Marr ve Hildreth 1980'de farklı ve daha etkili bir yaklaşım sunmuşlardır. Bu yöntemde görüntü değişik kesim frekanslarında sınırlandırılır ve herbir sınırlandırılmış görüntü için kenar tanıma işlemi uygulanarak değişik kenar görüntüleri elde edilir. Eğer bir kenar bu değişik çözünürlüklerdeki kenar görüntülerinde bulunuyorsa bu kenar bir gri seviyesi değişimini temsil eder.

Bu yöntemde görüntüyü değişik kesim frekanslarında sınırlandırmak için görüntünün iki boyutlu Gauss fonksiyonu ile konvolüsyonu alınır. Gauss fonksiyonu

$$G(x,y)=\frac{1}{2\pi\sigma^2}\exp\left[-(x^2+y^2)/2\sigma^2\right] \qquad (5.17)$$

biçiminde tanımlanır. Burada σ kesim frekansını belirler.

Konvolüsyonlu görüntünün Laplasyen kenar tanıma metodu kullanılarak kenarları saptanır. Bu yönteme Gauss'un Laplasyeni (the Laplacian of Gaussian : LoG) adı verilir. Gauss ve Laplasyen işlemleri için tek bir filtre tanımlanabilir:

$$\nabla^2\left[I(x,y)*G(x,y)\right]=\nabla^2G(x,y)*I(x,y) \qquad (5.18)$$

Burada I(x,y) görüntü fonksiyonun (x,y) noktasındaki gri seviyesini ve G(x,y) de iki boyutlu Gauss fonksiyonunu göstermektedir.

Gauss'un Laplasyeni yöntemi operatörü ayrıca sıfır geçişlerinin ince, sürekli ve kapalı hatlar olarak belirlenmesini sağlar. Bu özellik bir sonraki aşamada nesne sınırlarını bulmayı amaçlayan uygulamalarda son derece yararlı olmaktadır.

5.1.2 Şablon Eşleştirme Yöntemi

İdeal bir kenarın bir basamak fonksiyonu olmasından yola çıkan bu yaklaşımda ideal basamak kenarlarını temsil eden şablonlarla her bir nokta için görüntünün konvolüsyonu alınarak kenarlar bulunmaya çalışılır. ±45° açı yapan bütün kenarlar saptanabilir. Bir periyotta sekiz tane 45° olduğundan herbiri bir öncekinin 45° döndürülmüş hali olan sekiz ayrı şablon ile görüntü konvolüsyona sokulur ve herbir görüntü elemanı (pixel) için bu sekiz ayrı gradyandan en büyük olanı seçilir. Bu çıkış

gradyanının bir eşik değeri ile karşılaştırılması sonucu kenar elde edilir. Şablonlar kenar maskeleri olarak adlandırılırlar.

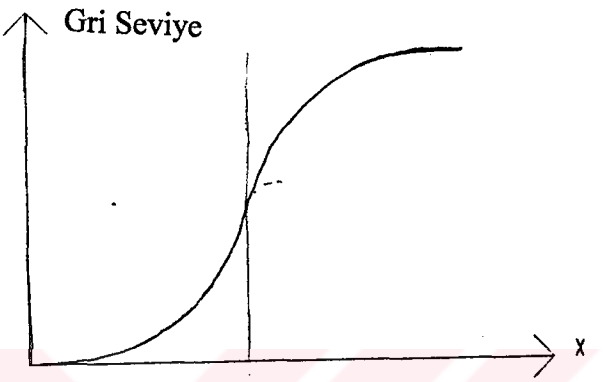
Literatürde tanımlanmış pek çok kenar maskesi vardır. En bilinenlerden üçü, Kirsch, Prewitt ve Sobel maskeleri Şekil 5.4'de verilmiştir.

Kirsch	Prewitt	Sobel
5 5 5	1 1 1	1 2 1
-3 0 -3	1 -2 1	0 0 0
-3 -3 -3	-1 -1 -1	-1 -2 -1
-3 5 5	1 1 1	2 1 0
-3 0 5	1 -2 -1	1 0 -1
-3 -3 -3	1 -1 -1	0 -2 -2
-3 -3 5	1 1 -1	1 0 -1
-3 0 5	1 -2 -1	2 0 -2
-3 -3 5	1 1 -1	1 0 -1
-3 -3 -3	1 -1 -1	0 -1 -2
-3 0 5	1 -2 -1	1 0 -1
-3 5 5	1 1 1	2 1 0
-3 -3 -3	-1 -1 -1	-1 -2 -1
-3 0 -3	1 -2 1	0 0 0
5 5 5	1 1 1	1 2 1
-3 -3 -3	-1 -1 1	-2 -1 0
5 0 -3	-1 -2 1	-1 0 1
5 5 -3	1 1 1	0 1 2
5 -3 -3	-1 1 1	-1 0 1
5 0 -3	-1 -2 1	-2 0 2
5 -3 -3	-1 1 1	-1 0 1
5 5 -3	1 1 1	0 1 2
5 0 -3	-1 -2 1	-1 0 1
-3 -3 -3	-1 -1 1	-2 -1 0
(a)	(b)	(c)

Şekil 5.4 Kenar saptama maskeleri (a) Kirsch (b) Prewitt (c) Sobel

Dikkat edilecek olursa bu maskelerin bazı ortak özellikleri olduğu görülür. Birinci özellik maskelerdeki sayıların toplamının sıfır olmasıdır. Eğer görüntünün 3x3 görüntü elemanı (pixel) boyutundaki bir parçası sabit bir değer içeriyorsa (hepsi 1 gibi) burada kenar yoktur ve bu alanın maske ile konvolüsyonu sıfır olmalıdır. Bu nedenle maskedeki sayıların toplamı sıfır seçilerek böyle sabit sayılardan oluşan

bölgelerde doğru sonuç olan sıfırı üretmesi sağlanmıştır. İkinci temel özellik maske kümesindeki herbir maskenin bir öncekinin 45° döndürülmüş hali olmasıdır. Üçüncü ortak özellik ise maskelerin kenar eğimini kuvvetlendirici nitelikte olmasıdır. Örneğin Şekil 5.5'teki bir boyutlu kenara bakılacak olursa kenara yakın noktalardaki gri seviyesi değerlerinin [3 5 7] gibi değerler olduğu kabul edilebilir. Bu üç nokta arasındaki eğim (7-3)/2 = 2'dir. Eğer bu üç nokta [-1 0 1] gibi bir maske ile konvolüsyona sokulursa -3+7=4 elde edilir. Konvolüsyon işlemi eğimi kuvvetlendirmiştir.



Şekil 5.5 Bir boyutlu kenar

5.1.3 Kenar Uygunlaştırma

Daha önce de belirtildiği üzere ideal bir kenar, görüntünün belli bir noktasında gri seviyesindeki basamak biçimli değişim olarak modellenebilir. Bir basamak fonksiyonu S(x,y) dairesel bir bölgede

$$S(x,y) = \begin{cases} b & (x\cos\theta + y\sin\theta) < \rho \\ b+h & (x\cos\theta + y\sin\theta) \geq \rho \end{cases} \qquad (5.19)$$

biçiminde tanımlanabilir. Burada

h adım büyüklüğü (gri seviyesindeki fark)
b temel gri seviyesi
ρ ve θ dairenin merkezine göre kenar çizgisinin yerini ve yönünü

belirtmektedir.

Bu yaklaşımda amaç, dairesel basamak fonksiyonu ile buna karşı düşen görüntü alanı arasındaki uzaklığı tanımlayan bir ölçütü enazlayan b, h, ρ ve θ değerlerini bulmaktır. Bu düşünce Hueckel operatörünün temelidir. Bir dairesel C

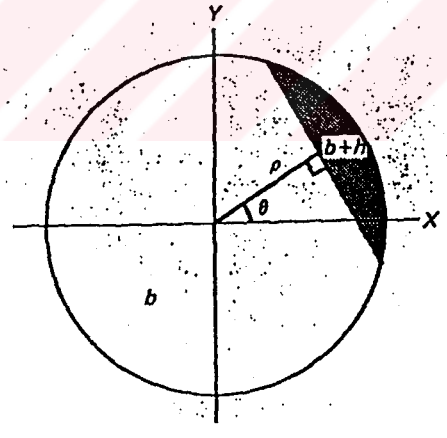
bölgesi üzerinde tanımlanan ideal basamak fonksiyonu $S(x,y,b,h,\rho,\theta)$ ve aynı boyuttaki dairesel bir alt görüntü f(x,y) arasındaki hata ε

$$\varepsilon^2 = \sum_{x,y \in C} (f(x,y) - S(x,y,b,h,\rho,\theta)) \qquad (5.20)$$

şeklinde tanımlanır. f(x,y), görüntü fonksiyonunun (x,y) noktasındaki değerini ve $S(x,y,b,h,\rho,\theta)$ de seçilmiş olan b, h, ρ ve θ değerleri için ideal basamak fonksiyonun (x,y) noktasındaki değerini göstermektedir. Toplama işlemi bütün C böldgesi üzerinde uygulanır ve E^2'yi enazlayacak b, h, ρ ve θ değerleri seçilir. Enazlama işlemi, f(x,y) ve $S(x,y,b,h,\rho,\theta)$) fonksiyonları Fourier fonksiyonlarına açılarak ve hata fonksiyonunda bu fonksiyonlar yerine katsayılar (f_i ve s_i) kullanılarak basitleştirilebilirler. Pratikte açınımın ilk sekiz terimi kullanılır. Böylece problem

$$E^2 = \sum_{i=0}^{7} (f_i - s_i) \qquad (5.21)$$

fonksiyonunun enazlanmasına indirgenmiş olur.



Şekil 5.6 Dairesel bir alanda basamak bir fonksiyonunu tanımlayan parametreler

5.1.4 Yerel İstatistik Yöntemi

Bu yöntemde yerel ortalama ve varyans değişimleri bulunarak kenarlar belirlenmektedir. Algoritma iki aşamalıdır. Birinci aşama pencereleme, ikinci aşama ise kenarların bulunmasıdır.

İlk aşamada görüntü (M x M) boyutlarında küçük pencerelere bölünür. Her bir pencerenin gri seviyesi ortalamaları ve gri seviyeleri arasındaki farklar hesaplanır. Gri seviyesi farkı belli bir eşik değerini geçen pencerelerde kenar olduğu kabul edilir. Burada pencere büyüklüğü çok önemlidir. Pencere büyüklüğü, içinden iki tane kenar geçmeyecek kadar küçük ve aynı zamanda rampa kenarları da içine alacak kadar büyük olmalıdır. Bütün kenarları saptayabilmek ve her bir pencerede sadece bir kenar bulunmasını sağlamak için pencereleme işlemi iki adımda yapılır. İlk adımda, belli kısımları komşu pencerelerle çakışacak şekilde büyük pencereler seçilerek gri seviyeleri ortalamaları komşu pencerelerinkilerle karşılaştırılır. Eğer fark belirli bir eşik değerini geçerse bu pencerede kenar vardır ve ikinci adım uygulanır. İkinci adımda, büyük pencere alt pencerelere bölünür ve bu küçük pencerelerin ortalama farkları hesaplanır. Böylece kenarlar küçük pencerelere indirgenmiş olur. Alt pencere ortalama farkları da belirli bir eşik değerini geçerse, o alt pencere bir kenar penceresi olarak saptanır ve kenarın tam yerini belirlemek için ikinci aşamaya geçilir.

İkinci aşamada kenar penceresinde sadece iki gri seviyesinin bulunduğu kabul edilir. Kenar bölgesi olduğu için bu iki gri seviyesi kenarın iki yanına aittir. Pencerenin gri seviyeleri ortalaması eşik değeri olarak seçilirse kenar geçişi saptanabilir. Ancak burada eşik değerinin üzerindeki tüm görüntü birimleri kenar olarak tanımlanacağından gerçek kenar noktalarını bulmak için yerel varyans testinin uygulanması gereklidir. Eşik varyans değerini geçen noktalar kenar olarak tanımlanır.

Bu yöntemle saptanan kenarlar gürültüsüz görüntüler için türev yöntemlerine göre daha kötüdürler. Ancak gürültülü görüntülerde daha iyi sonuçlar elde edilmektedir.

5.2 Dinamik Algılayıcı Öğrenme Kuralı ile Kenar Saptamanın Öğrenilmesi

Görüntü işlemede temel problem uygun bir operatör ve eşik değerinin bulunmasıdır. Bu sorunun çözümü için bir seçenek olarak yapay sinir ağları önerilmiştir [14]. YSA'larda uygun eğitim görüntüleri ile ağ eğitilerek operatör ve eşik değeri öğrenilebilmektedir.

Bu çalışmada, öğrenme işlemi Hücresel Yapay Sinir Ağları için geliştirilmiş olan Dinamik Algılayıcı Öğrenme Kuralı (RPLA) kullanılarak gerçekleştirilmiştir. Bölüm 4 'te tanıtılan bu algoritma ile istenilen herhangi bir görüntü işleme işlevini

öğrenmek mümkündür. Bu konuda daha önce yapılan çalışmalarda küçük boyutlu, siyah-beyaz, sentetik görüntüler işlenmiştir. RPLA ilk kez bu çalışmada 256 x 256 boyutunda, gri seviyeli, gerçel görüntülerde kenar saptamak için kullanılmıştır.

Yapılan çalışma iki grupta toplanabilir:

1- 256 x 256 boyutundaki, siyah-beyaz gerçel görüntülerde kenar saptama.
2- 256 x 256 boyutundaki, gri seviyeli gerçel görüntülerde kenar saptama.

Birinci gruptaki çalışmalar RPLA'nın büyük boyutlu, siyah-beyaz gerçel görüntülerde kenar saptama performansını ölçmeye yöneliktir. İlk olarak siyah-beyaz görüntülerden başlanılmasının nedeni daha önceki kenar saptama çalışmalarında bu tür görüntüler kullanılarak iyi sonuçlar alınmış olmasıdır. Siyah-beyaz gerçel görüntüler, gri-seviyeli görüntülerin bit görüntülerine ayrılması yoluyla elde edilmişlerdir. Çalışmada en anlamlı bitlerden oluşan bit-7 görüntüleri kullanılmıştır.

Bit görüntülerinin elde edilişi kısaca aşağıdaki gibi açıklanabilir.

Gri seviyeli görüntülerde her bir görüntü elemanı 8 bitten oluşan bir byte ile ifade edilir. Bir byte'ın alabileceği en büyük değer $2^8-1=1*2^0+1*2^1+1*2^2+...+1*2^7=255$'tir. Bu görüntüdeki en yüksek gri seviyesidir ve genellikle beyaza karşı düşer; siyah da sıfırla gösterilir. Gerçekte bir byte, iki tabanındaki sayıların bir dizisidir. Byte'ın en anlamlı biti bit-7 olarak adlandırılır ve diğerleri bitler bit-6, bit-5, bit-4, bit-3, bit-2, bit-1, bit-0 diye sıralanırlar. Bit-0 en az anlamlı bittir.

bit-7	bit-6	bit-5	bit-4	bit-3	bit-2	bit-1	bit-0

Şekil 5.7 Bir byte'ın gösterilimi

Bir byte 8 bitten oluştuğundan gri seviyeli bir görüntü, herbiri aynı seviyedeki bitlerden oluşan sekiz ikili (binary) görüntüye ayrılabilir. Anlaşılmayı kolaylaştırmak için 4x4'lük bir görüntüyü ele alalım. Bu görüntü P'ler bir görüntü elemanını göstermek üzere

P_1	P_2	P_3	P_4
P_5	P_6	P_7	P_8
P_9	P_{10}	P_{11}	P_{12}
P_{13}	P_{14}	P_{15}	P_{16}

şeklinde gösterilebilir. Görüntünün en anlamlı bitlerden oluşan 'bit-7' görüntüsü ise

$b_{1,7}$	$b_{2,7}$	$b_{3,7}$	$b_{4,7}$
$b_{5,7}$	$b_{6,7}$	$b_{7,7}$	$b_{8,7}$
$b_{9,7}$	$b_{10,7}$	$b_{11,7}$	$b_{12,7}$
$b_{13,7}$	$b_{14,7}$	$b_{15,7}$	$b_{16,7}$

biçiminde ifade edilebilir. Burada $b_{i,7}$ i'inci byte'ın 7'inci bitini gösterir. Benzer şekilde herhangi bir 'bit-N görüntüsü'

$b_{1,N}$	$b_{2,N}$	$b_{3,N}$	$b_{4,N}$
$b_{5,N}$	$b_{6,N}$	$b_{7,N}$	$b_{8,N}$
$b_{9,N}$	$b_{10,N}$	$b_{11,N}$	$b_{12,N}$
$b_{13,N}$	$b_{14,N}$	$b_{15,N}$	$b_{16,N}$

şeklinde gösterilebilir. Bit görüntülerini elde etmek için izlenecek yol istenilen bit dışında byte'ın diğer bitlerini maskelemektir. Örneğin bit-7 görüntüsü oluşturulurken byte'ın 6'dan 0'a kadar diğer bütün bitleri maskelenir ve görüntünün 7'inci bitlerinden

iki boyulu bir dizi oluşturulur. Böylece sadece 0 ve 1 değerlerinden oluşan bir görüntü elde edilir.

İkinci gruptaki çalışmalar gri seviyeli gerçel görüntüler üzerinde yoğunlaşmıştır.

Eğiticili öğrenme algoritması RPLA hem stokastik hem de deterministik olarak uygulanabilmektedir. Stokastik gerçeklemede her adımda eğitim çiftleri, eğitim kümesinden rastgele seçilerek ağa uygulanırlar ve ağırlık vektörü ω değiştirilir.

RPLA 'nın deterministik uygulamasında ise eğitim kümesindeki eğitim çiftleri sırayla işlenir. Bağlantı ağırlıkları tüm eğitim çiftleri ağa uygulandıktan sonra toplam hataya bakılarak değiştirilir. Bu çalışmada da bu yöntem kullanılmıştır.

İşlenecek görüntü, HYSA 'na ya dış girişlerden ya başlangıç durumlarından ya da her ikisinden birden uygulanabilir. Bu çalışmada her ikisinden birden verilmiştir.

HYSA'nın benzetiminde, diferansiyel denklemlerin çözümü için Açık Euler yöntemi kullanılmıştır.

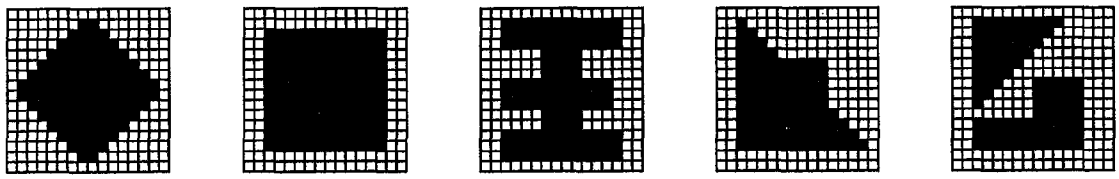
Öğrenme oranı başlangıç değeri olarak 0.0001 seçilmiş ancak eğitim süresince belirli bir kurala göre azaltılmış veya arttırılmıştır. Bu kural;

1- Eğer hata belli bir iterasyon süresinde sabit kalırsa öğrenme oranı arttırılır. Bu iterasyon sayısı 29 olarak belirlenmiştir.

2- Eğer değiştirilen ağırlık vektörünün bir önceki değeri ile arasındaki fark belli bir değeri geçerse öğrenme oranı azaltılır. Bu değer 0.1 olarak seçilmiştir.

Benzetim sonuçları aşağıda verilmiştir. Benzetimlerde iki şablon kullanılmıştır.

Öğrenme ile bulunan ve 1, 2, 3, 4 ve 5 numaralı deneylerde kullanılan ilk şablon eğitim çifti olarak Şekil 5.8'de verilen 5 tane 16x16 boyutlu eğitim çiftleri kullanılarak RPLA ile bulunmuştur. Bu eğitim kümesi dışından seçilen 16x16 boyutlu 50 farklı sentetik görüntü üzerinde %98 doğrulukla kenarları bulduğu tespit edilmiştir. Bu tezde, bulunan bu şablon eğitim kümesinde yer almayan Lenna, House ve satranç tahtası görüntülerinde denenmiş, oldukça başarılı olduğu görülmüştür.



Şekil 5.8 Şablon 1 eğitim kümesi

Şablon 1:

$$A=\begin{bmatrix}-0.096934 & -0.192281 & -0.088861\\ -0.179305 & 3.806794 & -0.179305\\ -0.088861 & -0.192281 & -0.096934\end{bmatrix}$$

$$B=\begin{bmatrix}-0.203938 & -0.203938 & -0.203938\\ -0.203938 & -0.101969 & -0.203938\\ -0.203938 & -0.203938 & -0.203938\end{bmatrix}$$

$$I=-0.193207$$

Yine öğrenme ile bulunan ve 6, 7, 8, 9 ve 10 numaralı deneylerde kullanılan ikinci şablon 256x256 boyutundaki gri seviyeli Lenna ve House görüntülerinden oluşan bir eğitim çifti kullanılarak bulunmuştur. Bu şablon istenen görüntü ile devrenin bulduğu görüntü arasındaki hatanın en az olduğu şablondur. Bu şablonun siyah-beyaz görüntülerde oldukça başarılı olduğu görülmüştür.

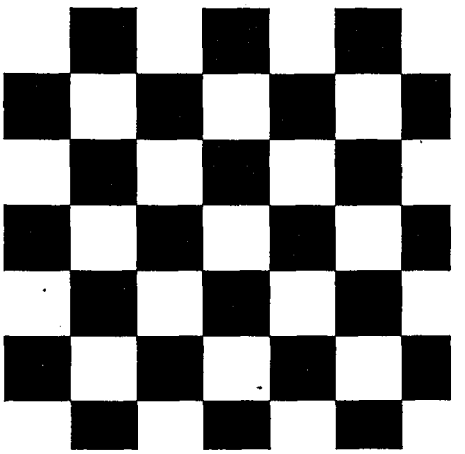
Şablon 2:

$$A=\begin{bmatrix}-0.005850 & -0.114069 & -0.043089\\ -0.094710 & 1.670608 & -0.094710\\ -0.043089 & -0.114069 & -0.005850\end{bmatrix}$$

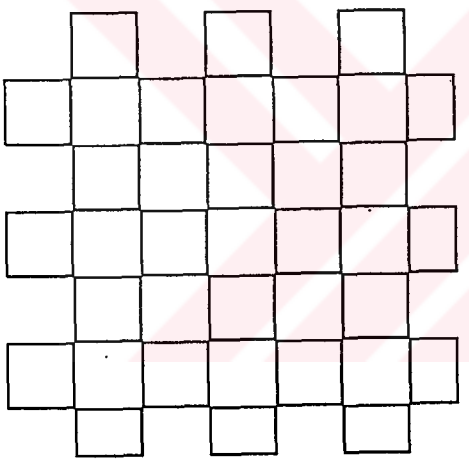
$$B=\begin{bmatrix}-0.200430 & -0.229714 & -0.211013\\ -0.216444 & 1.849995 & -0.216444\\ -0.211013 & -0.229714 & -0.200430\end{bmatrix}$$

$$I=-0.579392$$

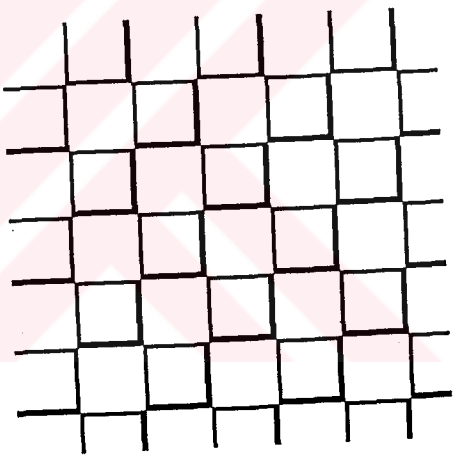
Deney 1:



(a)



(b)



(c)

(a) Orjinal siyah-beyaz satranç tahtası görüntüsü.
(b) RPLA algoritması ile kenar saptama uygulanmış satranç tahtası görüntüsü (Şalon 1).
(c) Canny algoritması ile kenar saptama uygulanmış satranç tahtası görüntüsü.

Deney 2:

(a)

(b)

(c)

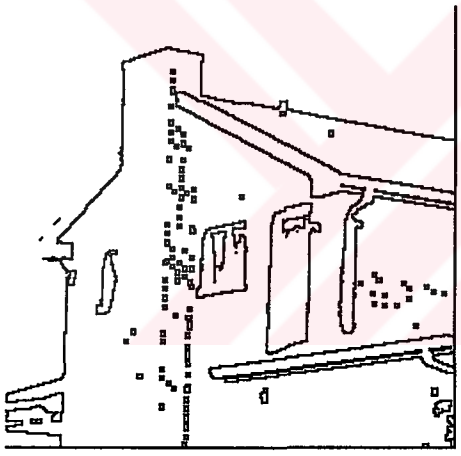
(a) Orjinal 'Lenna' görüntüsünün Bit-7 görüntüsü.

(b) 'Lenna' görüntüsünün RPLA algoritması ile kenar saptama uygulanmış Bit-7 görüntüsü (Şalon 1).

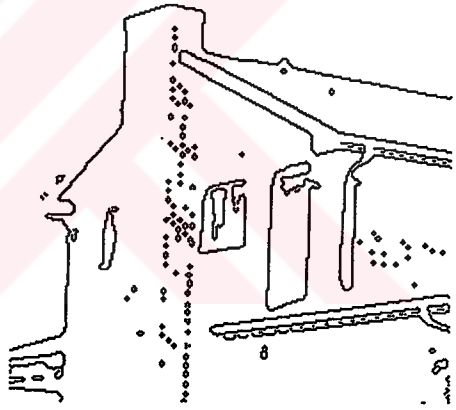
(c) 'Lenna' görüntüsünün Canny algoritması ile kenar saptama uygulanmış Bit-7 görüntüsü.

Deney 3:

(a)



(b)



(c)

(a) Orjinal ev görüntüsünün Bit-7 görüntüsü .

(b) Ev görüntüsünün RPLA algoritması ile kenar saptama uygulanmış Bit-7 görüntüsü (Şalon 1) .

(c) Ev görüntüsünün Canny algoritması ile kenar saptama uygulanmış Bit-7 görüntüsü .

Deney 4:

(a)

(b)

(c)

(a) Orjinal 'Lenna' görüntüsü (Gri seviyeli).

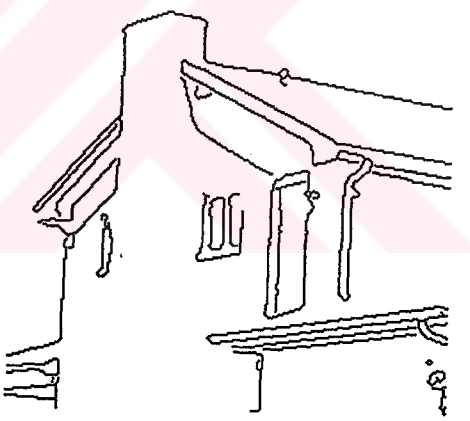
(b) RPLA algoritması ile kenar saptama uygulanmış 'Lenna' görüntüsü (Şalon 1).

(c) Canny algoritması ile kenar saptama uygulanmış 'Lenna' görüntüsü .

Deney 5:

(a)

(b)



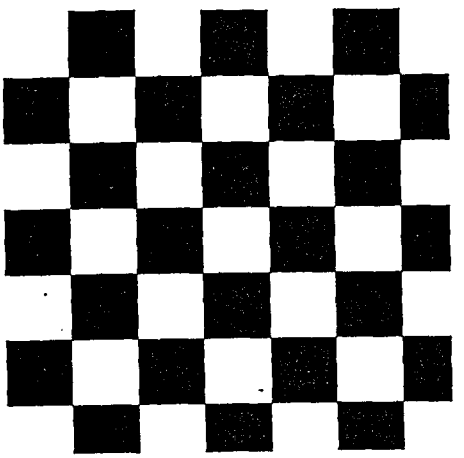
(c)

(a) Orjinal ev görüntüsü (Gri seviyeli).

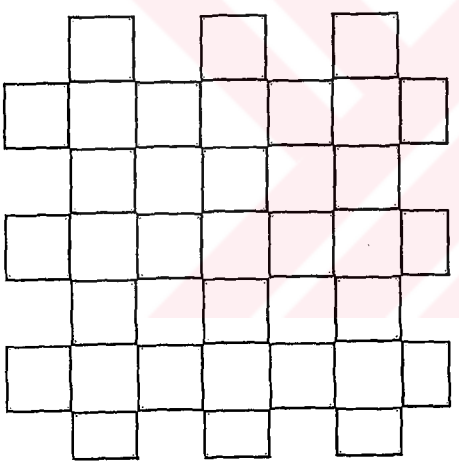
(b) RPLA algoritması ile kenar saptama uygulanmış ev görüntüsü (Şalon 1).

(c) Canny algoritması ile kenar saptama uygulanmış ev görüntüsü.

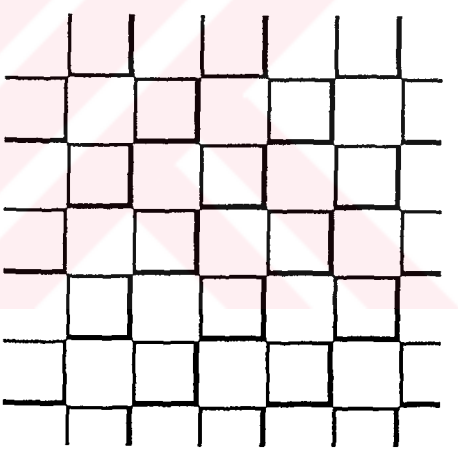
Deney 6:



(a)



(b)



(c)

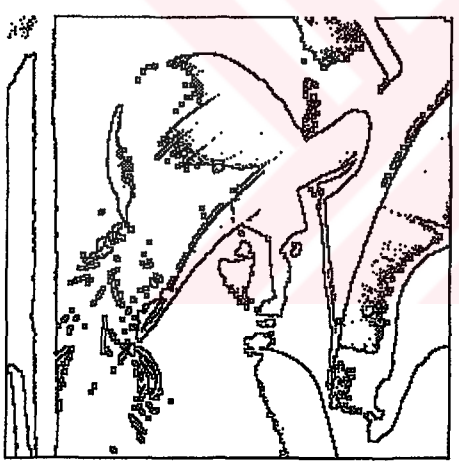
(a) Orjinal siyah-beyaz satranç tahtası görüntüsü .

(b) RPLA algoritması ile kenar saptama uygulanmış satranç tahtası görüntüsü (Şalon 2).

(c) Canny algoritması ile kenar saptama uygulanmış satranç tahtası görüntüsü .

Deney 7:

(a)



(b)

(c)

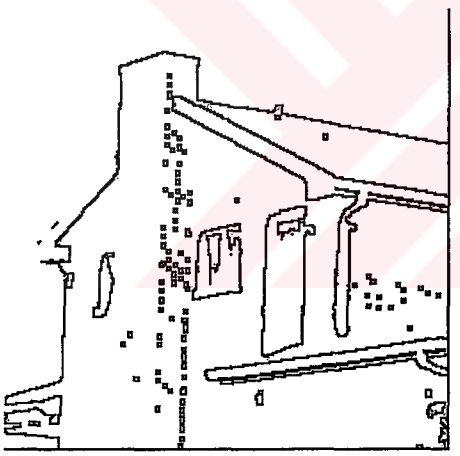
(a) Orjinal 'Lenna' görüntüsünün Bit-7 görüntüsü .

(b) 'Lenna' görüntüsünün RPLA algoritması ile kenar saptama uygulanmış Bit-7 görüntüsü (Şalon 2).

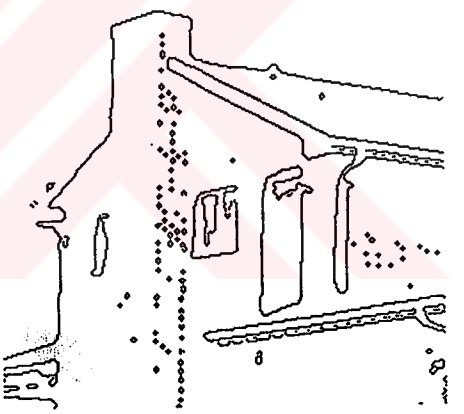
(c) 'Lenna' görüntüsünün Canny algoritması ile kenar saptama uygulanmış Bit-7 görüntüsü .

Deney 8:

(a)



(b)



(c)

(a) Orjinal ev görüntüsünün Bit-7 görüntüsü .

(b) Ev görüntüsünün RPLA algoritması ile kenar saptama uygulanmış Bit-7 görüntüsü (Şalon 2).

(c) Ev görüntüsünün Canny algoritması ile kenar saptama uygulanmış Bit-7 görüntüsü .

Deney 9:

(a)

(b)

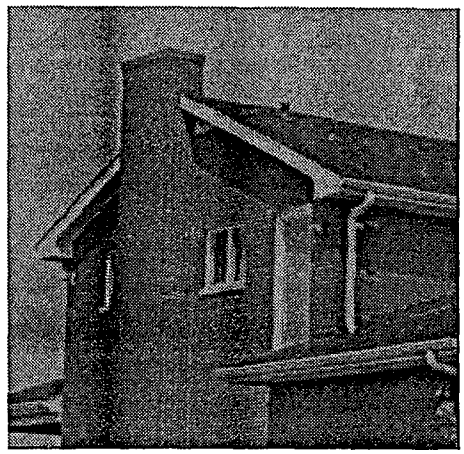
(c)

(a) Orjinal 'Lenna' görüntüsü (Gri seviyeli).

(b) RPLA algoritması ile kenar saptama uygulanmış 'Lenna' görüntüsü (Şalon 2).

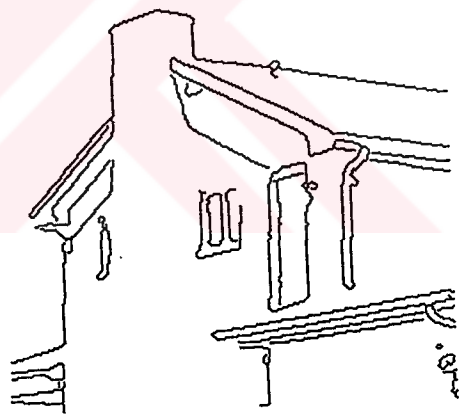
(c) Canny algoritması ile kenar saptama uygulanmış 'Lenna' görüntüsü .

Deney 10:



(a)

(b)



(c)

(a) Orjinal ev görüntüsü (Gri seviyeli).

(b) RPLA algoritması ile kenar saptama uygulanmış ev görüntüsü (Şalon 2).

(c) Canny algoritması ile kenar saptama uygulanmış ev görüntüsü .

BÖLÜM 6

CNNSLATW PROGRAMI KULLANIM KILAVUZU

6.1 Giriş

CNNSLATW (Celluar Neural Networks Supervised Learning Algorithm Tool for Windows), Hücresel Yapay Sinir Ağları için tasarlanmış Dinamik Algılayıcı Öğrenme Algoritması (Recurrent Perceptron Learning Algorithm : RPLA) eğiticili öğrenme algoritmasını gerçekleyen programdır. Program, belirli giriş görüntülerine ve başlangıç görüntülerine karşılık, istenen kalıcı-durum çıkış görüntülerini veren şablonları üretmektedir. Böylece, HYSA 'lar için değişik görüntü işlemeleri yapabilen şablonlardan oluşan bir kütüphane üretilebilmektedir.

Programda, HYSA benzetimi için Açık Euler analiz yöntemi kullanılmaktadır.

6.2 Gerekli Bilgisayar Donanımı

CNNSLATW programı Visual C++ programlama dilinde yazılarak Windows ortamında çalışması sağlanmış ve böylece benzeri programlarda görülen bellek problemi aşılmıştır. Program Windows yüklü olan IBM uyumlu bilgisayarlarda çalışabilmektedir. Programın yeterince hızlı çalışabilmesi için matematiksel işlemcisi olan bir 80386 veya daha gelişmiş bir bilgisayar tavsiye edilmektedir.

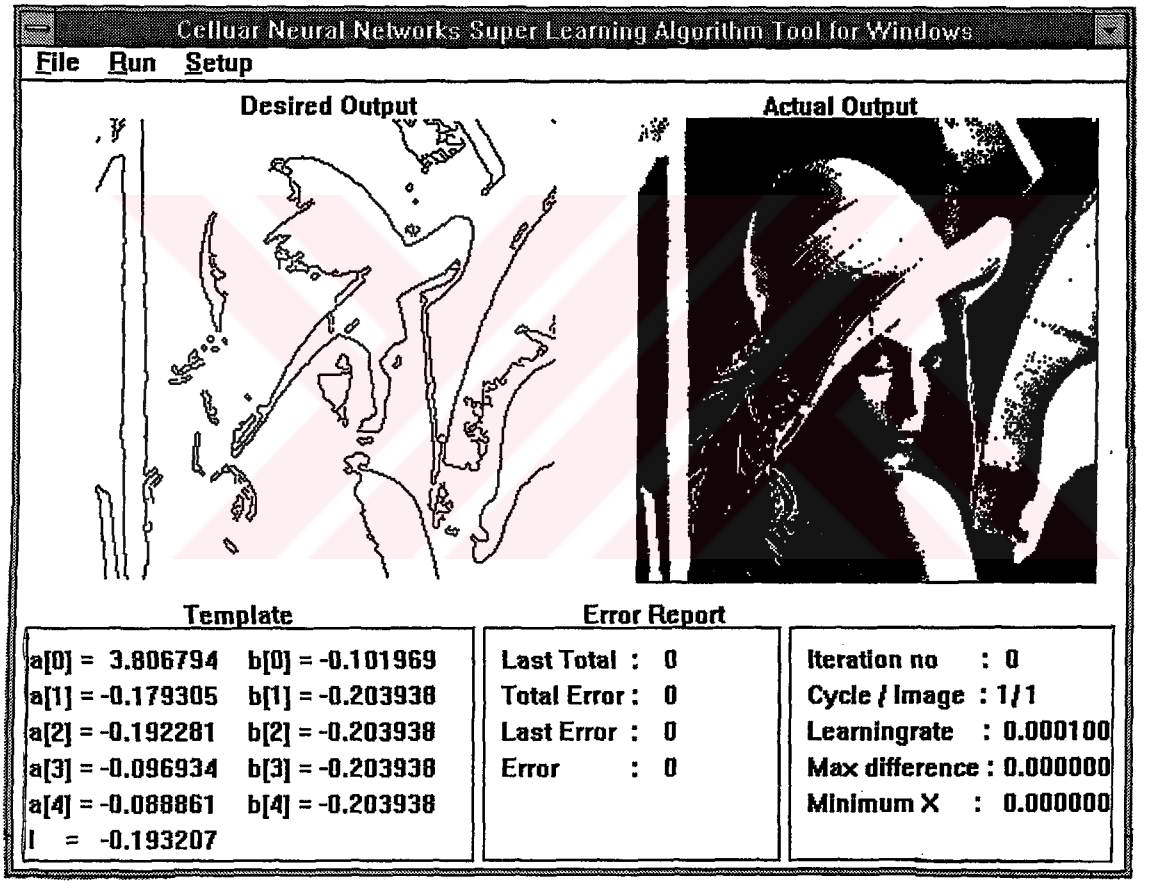
6.3 Programın İşleyişi

Program çalıştırıldıktan sonra eğitim sürecinin başlatılabilmesi için, dosya ("File") menüsünden eğitim kümesi ve başlangıç şablonu seçilmelidir. Sonra eğitim sürecinin başlatılması için ana menüde bulunan " Run " seçeneği aktif hale getirilir.

Öğrenme algoritması, toplam hata sıfır olduğunda eğitim sürecini durdurmaktadır. Eğitim süreci içinde kullanıcı, klavyedeki "S" tuşuna basarak programı durdurabilmekteedir. Eğitim süreci bittiğinde elde edilmiş olan şablonun kaydedilmesi gerekmektedir.

6.4 Ekran Görüntüsü

CNNSLATW programının ekranı Şekil 6-1 'de görüldüğü gibi dört bölümden oluşmaktadır. En üstteki bölüm menü, bunun altındaki bölüm görüntü, en altta soldaki bölüm şablon, ortadaki ve sağdaki bölümler de veri bloklarıdır.



Şekil 6.1 CNNSLATW Programının Ekran Görüntüsü

6.4.1 Menü Bloğu

Benzetim programının kullanıcı ile ilişkisi bir menü sistemi ile gerçeklenmiştir. Bu menü sistemi aracılığı ile kullanıcı istediği devre parametrelerini

tanımlayabilir, eğitim çiftlerini seçebilir ve benzetimi başlatabilir. Menü sisteminin yapısı bir ana menü ve onun altmenüleri şeklindedir. Bunları sırasıyla inceleyelim.

Ana Menü:

Ana menü en üst düzeyde gruplanmış menüleri içerir. Bunlar

1. File Menüsü
2. Run Seçeneği
3. Setup Menüsü

olarak sıralanmıştır. Menü seçeneklerine fare (mouse) veya klavye kullanarak erişilebilmektedir.

File Menüsü:

"File" menüsü içinde yapılabilecekler programın başlayabilmesi için gerek duyulan dosyaların yüklenmesi, benzetim sonucu elde edilen değerlerin dosyalarda saklanması, elde edilen çıkış görüntülerinin ve giriş görüntülerinin yazıcıya aktarılması ve programdan çıkılmasıdır.

Bu işlemler;

1. Load Menüsü
2. Save Menüsü
3. Print Menüsü
4. Exit Seçeneği

şeklinde gruplandırılmıştır.

"**Load**" menüsü seçildiğinde bu menüye ait seçenekler görünür. Bu seçenekler

1. Initial Template
2. Trainig set

şeklinde sıralanmıştır.

"**Initial Template**" seçildiğinde başlangıç şablonunu tanımlayan dosya yüklenmektedir. Bu dosyanın özellikleri 6.5.2'de verilmiştir.

"**Trainig set**" seçildiğinde eğitim kümesini tanımlayan dosya yüklenmektedir. Bu dosyanın yapısı 6.5.1'de verilmiştir.

"**Save**" menüsü seçildiğinde bu menüye ait seçenekler görünür. Bu seçenekler

1. Output Image
2. Template

şeklinde sıralanmıştır.

"**Output Image**" seçeneği seçildiğinde o anda devre çıkışında bulunan görüntü kullanıcı tarafından adı belirlenen bir dosyaya BMP (bitmap) olarak yazılır. Bu dosya açılamaz ya da herhangi bir hata nedeniyle kapatılamazsa kullanıcı uyarılarak bir önceki duruma dönülür.

"**Template**" seçeneği seçildiğinde devrenin o anki şablonu kullanıcı tarafından adı belirlenen bir dosyaya yazılır. Herhangi bir nedenle yazma işlemi gerçekleştirilemezse kullanıcı uyarılarak bir önceki duruma geri dönülür.

"**Print**" menüsü seçildiğinde bu menüye ait seçenekler görünür. Bu seçenekler

1. Input Image
2. Actual Output Image
3. Desired Output Image

"**Input Image**" seçeneği seçildiğinde devreye giriş olarak uygulanmış olan görüntü yazıcıya gönderilir.

"**Actual Output Image**" seçeneği seçildiğinde devrenin o andaki çıkış görüntüsü yazıcıya gönderilir.

"**Desired Output Image**" seçeneği seçildiğinde devreye istenen çıkış olarak uygulanmış olan görüntü yazıcıya gönderilir.

"**Exit**" seçeneği seçildiğinde açık olan bütün dosyalar kapatılır, dinamik olarak kullanılan bellekler serbest bırakılır ve programdan çıkılır.

Run Seçeneği:

Benzetimin başlayabilmesi için gerekli olan dosyalar "File" menüsü kullanılarak seçildikten sonra programın koşması için bu seçenek aktif hale getirilir. Benzetimi herhangi bir aşamada durdurmak için klavyedeki "S" tuşuna basılmalıdır.

Setup Menüsü:

"Setup" menüsü içinde yapılabilecekler bazı parametrelerinin ayarlanmasıdır. Bu menüde iki seçenek mevcuttur:

1. Learning rate
2. Cycles

"**Learning rate**" seçeneği seçildiğinde devrenin öğrenme oranı görülebilmekte ve eğer istenirse değiştirişebilmektedir.

"**Cycles**"seçeneği seçildiğinde benzetimin çevrim sayısı görülebilmekte ve eğer istenirse değiştirişebilmektedir.

6.4.2 Şablon Bloğu

Şablon bloğunun eğitim süreci boyunca şablonu göstermektedir. Eğitim sırasında meydana gelen değişimler buradan izlenebilmektedir.

6.4.3 Görüntü Bloğu

Görüntü bloğu iki parçadan oluşmaktadır. Sol tarafta istenen kalıcı-durum çıkış görüntüsü ve sağ tarafta da benzetim süreci boyunca oluşan gerçek çıkış görüntüsü gösterilmektedir. Çıkış kararlı duruma ulaştığında, gerçek çıkış görüntüsü siyah-beyaz duruma dönüşmektedir.

6.4.4 Veri Bloğu

Veri bloğu eğitim sürecinde değişen verileri göstermektedir. Bu bölümlerde benzetim sırasında oluşan hatalar ve çevrim sayısı gibi bilgiler verilmektedir.

6.5 Kütüphaneler

CNNSLATW programı eğitim kümesi kütüphanesini, şablon kütüphanesini ve görüntü kütüphanesini kullanmaktadır.

6.5.1 Eğitim Kümesi Kütüphanesi

Eğitim kümesi kütüphanesinde eğitim kümeleri bulunmaktadır. Bu kütüphane "TSE" alt dizininde bulunmaktadır. Eğitim kümeleri kullanıcı tarafından da yaratılabilmektedir. Bunun için text modda dosya oluşturan herhangi bir editör kullanılabilir. Uyulması gereken format Şekil 6.2 'de verilmiştir. İlk olarak eğitim kümesinde bulunan örneklerin sayısının verilmesi gerekmektedir. Her bir örnek sırasıyla, başlangıç görüntüsünün dosya ismi, giriş görüntüsünün dosya ismi ve istenen çıkış görüntüsünün dosya isminden oluşmaktadır. Dosya isimleri, varsa, uzantıları ile birlikte verilmelidir. Şekil 6.2 'de verilmiş olan eğitim kümesi örneğinde dört tane eğitim örneği bulunmaktadır. Kümenin ilk örneğinde, "image1" dosyası başlangıç görüntüsü ve giriş görüntüsü olarak, "dimage1" dosyası da istenen çıkış görüntüsü olarak tanımlanmıştır. Belirtilen görüntü dosyaları mutlaka " İMAGE " alt dizininde bulunmalıdır.

Seçilmiş olan eğitim kümesinde bulunan örnekler sırasıyla işlenmektedir. Her bir çevrim için eğitim kümesinin başından başlanılmaktadır.

```
4
image1 image1 dimage1
image2 image2 dimage2
image3 image3 dimage3
image4 image4 dimage4
```

Şekil 6.2 Eğitim Kümesi Örneği

6.5.2 Şablon Kütüphanesi

Şablon kütüphanesinde eğitim sürecinin ilk çevriminde kullanılması gereken başlangıç şablonları ve " SAVE " menüsünden kullanıcı tarafından kaydedilen şablonlar bulunmaktadır. Ayrıca CNNSLATW programı, eğitim süreci boyunca en az hatayı veren şablonu "MİN.TEM" adıyla bu kütüphaneye yüklemektedir. Bu kütüphane " TEMPLATE " alt dizininde bulunmaktadır.

Şekil 6.3 ile bir şablon örneği verilmiştir. " Neighborhood: " ile HYSA 'nın yakın komşuluk boyutu belirtilmektedir. CNNSLATW sadece 1 yakın komşuluk boyutu için çalışmaktadır. " Feedback: " ile geribesleme şablonu , " Control: " ile kontrol şablonu ve " Current: " ile eşik şablonu belirtilmektedir. Buralarda " : " 'dan sonra mutlaka bir boşluk bulunmalıdır. Kullanıcı aynı formatta kendi şablonunu oluşturabilir. Bu amaç için herhangi bir editör kullanılabilir. Geribesleme ve kontrol şablonları simetrik olmalıdır.

```
Neighborhood: 1

Feedback:  0 0 0
           0 4 0
           0 0 0

Current: -1

Control:  1 1 1
          1 3 1
          1 1 1
```

Şekil 6.3 Şablon Dosyası Örneği

6.5.3 Görüntü Kütüphanesi

Görüntü kütüphanesi " İMAGE " alt dizininde bulunmaktadır. Görüntü dosyaların formatı bitmap (BMP) formatıdır. Görüntüler 256x256 görüntü elemanı (pixel) boyutunda olmalıdır.

SONUÇ VE ÖNERİLER

Bu tezde, tam kararlı Hücresel Yapay Sinir Ağları için geliştirilen bir eğiticili öğrenme kuralı olan Dinamik Algılayıcı Öğrenme Kuralı (Recurrent Perceptron Learning Algorithm : RPLA) ile kenar saptamanın öğrenilebileceği gösterilmiş ve literatürde bilinen en iyi kenar saptama yöntemi olan Canny ile kıyaslanabilecek bir kenar saptayan şablon bulunmuştur. Bu öğrenme kuralını kullanan şablon öğrenme yazılımı CNNSLATW (Cellular Neural Network Supervised Learning Algorithm Tool for Windows) geliştirilerek bir kişisel bilgisayar ortamında benzetim düzeni elde edilmiştir.

RPLA kullanılarak eğitilen HYSA 'da siyah-beyaz ve gri seviyeli, 256 x 256 görüntü elemanı (pixel) boyutundaki gerçel görüntüler üzerinde kenar saptama işlemi uygulanmıştır. Siyah-beyaz görüntülerde oldukça başarılı sonuçlar elde edilmiştir. Ancak aynı başarı gri seviyeli görüntüler için elde edilememiştir.

Bu çalışmada RPLA' nın siyah-beyaz gerçel görüntülerde kenar saptamada kullanılabilecek en iyi yöntemlerden biri olduğu gösterilmiştir. Gri seviyeli gerçel görüntüler için ise yöntemin geliştirilmesine ihtiyaç vardır.

Tümleşik devre olarak gerçeklenmeye son derece uygun olan HYSA mimarisi RPLA ile birleştirildiğinde paralel olarak çalışan çok iyi bir kenar saptayıcı elde edilebilir.

KAYNAKLAR

[1] HECHT-NIELSEN, R., Neurocomputing, Addison-Wesley ,1990.

[2] GÜZELİŞ, C., "Genelleştirilmiş Hücresel Yapay Sinir Ağları", Elektrik Mühendisliği 5. Ulusal Kongresi, Trabzon, pp.7-12, 1993.

[3] DARPA Neural Network Study. Fairfax, VA: AFCEA International Press,1988.

[4] CHUA, L.O. ,and YANG, L., "Cellular Neural Networks : Theory", IEEE Transaction on Circuits and Systems, vol.35. pp.1257-1272,October 1988.

[5] CHUA, L.O.,and YANG, L., "Cellular Neural Networks : Application", IEEE Transaction on Circuits and Systems, vol.35. pp.1273-1290, October 1988.

[6] ROSKA, T., and KEK, L., Analogic CNN Program Library, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, December 1994.

[7] GÜZELİŞ, C., Yapay Sinir Ağları, Yüksek Lisans Ders Notları , İTÜ, 1991.

[8] SEVEN, A., Yapay Sinir Ağları ile Doku Sınıflandırma, Yüksek Lisans Tezi, İTÜ, Temmuz 1993.

[9] WUNSCH,G., and MATHIS, W.,"Toward A Theory of Cellular Systems",

[10] ROSKA, T., and CHUA, L., "Cellular Neural Networks With Nonlinear and Delay-Type Template Elements".

[11] GÜZELİŞ, C., and CHUA, L.,"Stability Analysis of Generalized Cellular Neural Networks", International Journal of Circuit Theory and Applications, Vol.21, pp.1-33, 1993.

[12] GÜZELİŞ, C.,and KARAMAHMUT, S.," Recurrent Perceptron Learning Algorithm for Completely Stable Cellular Neural Networks ", Submitted to IEEE Transaction on Circuits and Systems, Part I: Fundamental Theory and Applications.

[13] SHI, B., and CHUA, L., "A Generalized Cellular Neural Network Implementation of A Novel Edge Detection Algorithm", Electronics Research Laboratory, College of Engineering, University of California, Berkeley, 9 April 1991.

[14] NACHBAR, P., and STROBL, M., and NOSSEK, J., "Design of DTCNNs for Edge Detection", AEÜ, Vol.49, pp.12-17, 1995.

[15] LU,Y.,and JAIN,R., "Reasoning about Edges in Scale Space", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.14, pp.450-468, April 1992.

[16] CHU, C. and AGGARWAL, J., "The Integration of Image Segmentation Maps Using Region and Edge Information", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.15, pp.1241-1267,December 1993.

[17] WANI, A. and BATCHELOR, B., "Edge-Region-Based Segmentation of Range Images",IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.16, pp.314-326.

[18] GREGSON, P.H., "Using Angular Dispersion of Gradient Direction for Detecting Edge Ribbons", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.15, pp.682-695,July 1993.

[19] CANNY, J., "A Computational Approach to Edge Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, pp.679-698, 1986.

[20] MARR, D., Vision, W. H. Freeman and Company, 1982.

[21] ERÇİL, A., Image Processing and Machine Vision, Yüksek Lisans Ders Notları , B.Ü., 1994.

[22] PHILLIPS, D., Image Processing in C, R&D Publications Inc. , 1993.

EK:

CNNSLATW YAZILIMI KAYNAK KODU

```
/*****************************************************************************

   PROGRAM: cnnmain.c

*****************************************************************************/

#include <windows.h>
#include "cnnslatw.h"

#include <stdlib.h>
#include <commdlg.h>

char        achFileName[128] = "";
DWORD       dwOffset;
HANDLE      hInst ;
HWND        hWndApp;                /* Handle to app. window */
WORD        UpdateCount = 0;

BOOL tem_OK, tse_OK;
BOOL is_running;

extern char method_type[20];
extern char huge *bmp_U;
extern char huge *bmp_Y;
extern char huge *bmp_E;

/*****************************************************************************

   FUNCTION: WinMain(HANDLE, HANDLE, LPSTR, int)

   PURPOSE: calls initialization function, processes message loop

*****************************************************************************/

int PASCAL WinMain(hInstance, hPrevInstance, lpszCmdLine, nCmdShow)
HANDLE hInstance;
HANDLE hPrevInstance;
LPSTR lpszCmdLine;
int nCmdShow;
{

  MSG       msg ;

   /* Save the pointer to the command line */
   lstrcpy(achFileName, lpszCmdLine);

   hInst = hInstance ;

   if (!hPrevInstance)
   if (!InitApplication(hInstance))
     return (FALSE);

   if  (!InitInstance(hInstance, nCmdShow))
     return (FALSE);


   /* Enter message loop */
   while (GetMessage (&msg, NULL, 0, 0)) {
```

```
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }

    return msg.wParam ;
}


/****************************************************************************

    FUNCTION: InitApplication(HANDLE)

    PURPOSE: Initializes window data and registers window class

****************************************************************************/

BOOL InitApplication(hInstance)
HANDLE hInstance;
{
    WNDCLASS  wc;

    wc.style = NULL;
    wc.lpfnWndProc = MainWndProc;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    wc.hInstance = hInstance;
    wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground = COLOR_WINDOW+1;
    wc.lpszMenuName =  (LPSTR) "CNNMenu";
    wc.lpszClassName = (LPSTR) "CNNWClass";

    return (RegisterClass(&wc));
}


/****************************************************************************

    FUNCTION:  InitInstance(HANDLE, int)

    PURPOSE:  Saves instance handle and creates main window

****************************************************************************/

BOOL InitInstance(hInstance, nCmdShow)
    HANDLE          hInstance;
    int             nCmdShow;
{
    HWND            hWnd;

    hInst = hInstance;
```

```
   /* Create the app. window */

   hWnd = CreateWindow("CNNWClass",
                "Celluar Neural Networks Super Learning Algorithm Tool for Windows",

WS_OVERLAPPED|WS_MAXIMIZE|WS_SYSMENU|WS_MINIMIZEBOX|WS_THICKFRAME,
                0,
                0,
                640,
                480,
                (HWND) NULL,
                NULL,
                hInstance,
                (LPSTR) NULL
               );

   if (!hWnd)
      return (FALSE);


   ShowWindow (hWndApp = hWnd, nCmdShow) ;
   return (TRUE);

}

/****************************************************************************

   FUNCTION: MainWndProc(HWND, unsigned, WORD, LONG)

   PURPOSE:  Processes messages

   MESSAGES:

   WM_COMMAND    - application menu (About dialog box)
      WM_CREATE     - create window and objects
      WM_LBUTTONDOWN - begin selection
      WM_MOUSEMOVE   - keep track of mouse movement during selection
      WM_LBUTTONUP   - end selection, draw bitmap
      WM_RBUTTONUP   - draw bitmap without resizing
      WM_ERASEBKGND  - erase background and redraw
      WM_DESTROY     - destroy window

   COMMENTS:

      User may select a "normal" size bitmap by pressing the right mouse
      button, or set the size of the bitmap to display by using the left
      button to define a region.  The routines to display the selection
      region are defined in the library module select.exe.
   WM_DESTROY    - destroy window

****************************************************************************/

long FAR PASCAL __export MainWndProc(hWnd, message, wParam, lParam)
HWND    hWnd ;
UINT    message ;
WPARAM  wParam ;
LPARAM  lParam ;

{
```

```
PAINTSTRUCT     ps;
HDC             hDC;

switch (message) {
   case WM_DESTROY:
        /* Clean up and quit */
        FreeMemories();
        PostQuitMessage(0);
        break ;

   case WM_CREATE:

        tem_OK = FALSE;
                        tse_OK = FALSE;
                        is_running = FALSE;

                        /* Allocate memories */
                        AllocateMemories();

                        /* Read default configuration */
                        ReadConfiguration();
                        break;


   case WM_ACTIVATE:
        if (!wParam)  /* app. is being de-activated */
          break;
        /* If the app. is moving to the foreground, fall through and
         * redraw full client area with the newly realized palette,
         * if the palette has changed.
         */


   case WM_COMMAND:
        /* Process menu commands */
        return MenuCommand (hWnd, wParam);
        break;


   case WM_PAINT:
        /* If we have updated more than once, the rest of our
         * window is not in some level of degradation worse than
         * our redraw...  we need to redraw the whole area
         */
        if (UpdateCount > 1) {
           BeginPaint(hWnd, &ps);
           EndPaint(hWnd, &ps);
           UpdateCount = 0;
           InvalidateRect(hWnd, (LPRECT) (NULL), 1);
           break;
        }

        hDC = BeginPaint(hWnd, &ps);
        EndPaint(hWnd, &ps);
        break ;

    default:
      return (DefWindowProc(hWnd, message, wParam, lParam));

   }
```

```
  return 0L ;

}


/*************************************************************************
 *                                                                       *
 * FUNCTION   : MenuCommand ( HWND hWnd, WORD wParam)                    *
 *                                                                       *
 * PURPOSE    : Processes menu commands.                                 *
 *                                                                       *
 * RETURNS    : TRUE  - if command could be processed.                   *
 *              FALSE - otherwise                                        *
 *                                                                       *
 *************************************************************************/
BOOL MenuCommand (hWnd, id)
HWND hWnd;
WORD id;

{
  int           fh;
  OPENFILENAME    ofn;
        char          Name[40];
  int          xSize, ySize, xRes, yRes, dx, dy;
  RECT          Rect;
  HDC                           hDC;


  switch (id)
  {
    case ID_FILE_LOAD_INITIALTEMPLATE:


                    /* Bring up File/Open ... dialog */
                  achFileName[0] = NULL;
                  ofn.lStructSize = sizeof(OPENFILENAME);
                  ofn.hwndOwner = hWnd;
                  ofn.hInstance = hInst;
                  ofn.lpstrFilter = "Templates\0*.TMP\0";
                  ofn.lpstrCustomFilter = NULL;
                  ofn.nMaxCustFilter = 0L;
                  ofn.nFilterIndex = 1L;
                  ofn.lpstrFile = (LPSTR)achFileName;
                  ofn.nMaxFile = 128;
                  ofn.lpstrInitialDir = NULL;
                  ofn.lpstrTitle = NULL;
                  ofn.lpstrFileTitle = NULL;
                  ofn.lpstrDefExt = NULL;
                  ofn.Flags = OFN_HIDEREADONLY;

                  fh=GetOpenFileName((LPOPENFILENAME)&ofn);

                  /*  Load up the template if the user did not press cancel */
                  if (fh > 0) ReadTemplate();
                  break;

    case ID_FILE_LOAD_TRAININGSET:

          /* Bring up File/Open ... dialog */
```

```
        achFileName[0] = NULL;
                ofn.lStructSize = sizeof(OPENFILENAME);
                ofn.hwndOwner = hWnd;
                ofn.hInstance = hInst;
                ofn.lpstrFilter = "Trainingset\0*.TSE\0";
                ofn.lpstrCustomFilter = NULL;
                ofn.nMaxCustFilter = 0L;
                ofn.nFilterIndex = 1L;
                ofn.lpstrFile = (LPSTR)achFileName;
                ofn.nMaxFile = 128;
                ofn.lpstrInitialDir = NULL;
                ofn.lpstrTitle = NULL;
                ofn.lpstrFileTitle = NULL;
                ofn.lpstrDefExt = NULL;
                ofn.Flags = OFN_HIDEREADONLY;

                fh=GetOpenFileName((LPOPENFILENAME)&ofn);

                /*  Load up the training set file if the user did not press cancel */
                if (fh > 0) ReadTrainingset();
    break;

case ID_FILE_SAVE_OUTPUTIMAGE:

                /* Bring up File/Save ... dialog */
                achFileName[0] = NULL;
                ofn.lStructSize = sizeof(OPENFILENAME);
                ofn.hwndOwner = hWnd;
                ofn.hInstance = hInst;
                ofn.lpstrFilter = "Bitmaps\0*.BMP\0";
                ofn.lpstrCustomFilter = NULL;
                ofn.nMaxCustFilter = 0L;
                ofn.nFilterIndex = 1L;
                ofn.lpstrFile = (LPSTR)achFileName;
                ofn.nMaxFile = 128;
                ofn.lpstrInitialDir = NULL;
                ofn.lpstrTitle = NULL;
                ofn.lpstrFileTitle = NULL;
                ofn.lpstrDefExt = NULL;
                ofn.Flags = 0;

                fh=GetSaveFileName((LPOPENFILENAME)&ofn);

                /* Save the DIB if the user did not press cancel */
                if (fh > 0) SaveImage();
                break;

case ID_FILE_SAVE_TEMPLATE:

        /* Bring up File/Save ... dialog */
                achFileName[0] = NULL;
                ofn.lStructSize = sizeof(OPENFILENAME);
                ofn.hwndOwner = hWnd;
                ofn.hInstance = hInst;
                ofn.lpstrFilter = "Templates\0*.TMP\0";
                ofn.lpstrCustomFilter = NULL;
                ofn.nMaxCustFilter = 0L;
                ofn.nFilterIndex = 1L;
                ofn.lpstrFile = (LPSTR)achFileName;
```

```
            ofn.nMaxFile = 128;
            ofn.lpstrInitialDir = NULL;
            ofn.lpstrTitle = NULL;
            ofn.lpstrFileTitle = NULL;
            ofn.lpstrDefExt = NULL;
            ofn.Flags = 0;

            fh=GetSaveFileName((LPOPENFILENAME)&ofn);

            /* Save the template if the user did not press cancel */
            if (fh > 0) SaveTemplate();
    break;

case ID_FILE_PRINT_INPUTIMAGE:

    GetWindowText(hWnd, Name, sizeof(Name));

    /* Initialise printer stuff */
    if (!(hDC = GetPrinterDC()))
        break;

    xSize = GetDeviceCaps(hDC, HORZRES);
    ySize = GetDeviceCaps(hDC, VERTRES);
    xRes  = GetDeviceCaps(hDC, LOGPIXELSX);
    yRes  = GetDeviceCaps(hDC, LOGPIXELSY);

    /* Use half inch margins on left and right
     * and one inch on top. Maintain the same aspect ratio.
     */

    dx = xSize - xRes;
    dy = (int)dx;

    /* Fix bounding rectangle for the picture .. */
    Rect.top    = yRes;
    Rect.left   = xRes / 2;
    Rect.bottom = yRes + dy;
    Rect.right  = xRes / 2 + dx;

    /* ... and inform the driver */
    Escape(hDC, SET_BOUNDS, sizeof(RECT), (LPSTR)&Rect, NULL);

    if (InitPrinting(hDC, hWnd, hInst, Name)) {

        PrintDIB(hDC, (LPBITMAPINFO)bmp_U, xRes/3, yRes, dx/3, dy/3, 0,0,256,256);

        /* Signal to the driver to begin translating the drawing
         * commands to printer output...
         */
        Escape (hDC, NEWFRAME, NULL, NULL, NULL);

        TermPrinting(hDC);
    }

    DeleteDC(hDC);
    break;

case ID_FILE_PRINT_ACTUALOUTPUTIMAGE:

    GetWindowText(hWnd, Name, sizeof(Name));
```

```
        /* Initialise printer stuff */
        if (!(hDC = GetPrinterDC()))
            break;

        xSize = GetDeviceCaps(hDC, HORZRES);
        ySize = GetDeviceCaps(hDC, VERTRES);
        xRes  = GetDeviceCaps(hDC, LOGPIXELSX);
        yRes  = GetDeviceCaps(hDC, LOGPIXELSY);

        /* Use half inch margins on left and right
         * and one inch on top. Maintain the same aspect ratio.
         */

        dx = xSize - xRes;
        dy = (int)dx;

        /* Fix bounding rectangle for the picture .. */
        Rect.top    = yRes;
        Rect.left   = xRes / 2;
        Rect.bottom = yRes + dy;
        Rect.right  = xRes / 2 + dx;

        /* ... and inform the driver */
        Escape(hDC, SET_BOUNDS, sizeof(RECT), (LPSTR)&Rect, NULL);

        if (InitPrinting(hDC, hWnd, hInst, Name)) {

            PrintDIB(hDC, (LPBITMAPINFO)bmp_Y, xRes/3, yRes, dx/3, dy/3, 0,0,256,256);

            /* Signal to the driver to begin translating the drawing
             * commands to printer output...
             */
            Escape (hDC, NEWFRAME, NULL, NULL, NULL);

            TermPrinting(hDC);
        }

        DeleteDC(hDC);

        break;

    case ID_FILE_PRINT_EXPECTEDOUTPUTIMAGE:

        GetWindowText(hWnd, Name, sizeof(Name));

        /* Initialise printer stuff */
        if (!(hDC = GetPrinterDC()))
            break;

        xSize = GetDeviceCaps(hDC, HORZRES);
        ySize = GetDeviceCaps(hDC, VERTRES);
        xRes  = GetDeviceCaps(hDC, LOGPIXELSX);
        yRes  = GetDeviceCaps(hDC, LOGPIXELSY);

        /* Use half inch margins on left and right
         * and one inch on top. Maintain the same aspect ratio.
         */

        dx = xSize - xRes;
```

```
        dy = (int)dx;

        /* Fix bounding rectangle for the picture .. */
        Rect.top    = yRes;
        Rect.left   = xRes / 2;
        Rect.bottom = yRes + dy;
        Rect.right  = xRes / 2 + dx;

        /* ... and inform the driver */
        Escape(hDC, SET_BOUNDS, sizeof(RECT), (LPSTR)&Rect, NULL);

        if (InitPrinting(hDC, hWnd, hInst, Name)) {

            PrintDIB(hDC, (LPBITMAPINFO)bmp_E, xRes/3, yRes, dx/3, dy/3, 0,0,256,256);

            /* Signal to the driver to begin translating the drawing
             * commands to printer output...
             */
            Escape (hDC, NEWFRAME, NULL, NULL, NULL);

            TermPrinting(hDC);
        }

        DeleteDC(hDC);

        break;

    case ID_FILE_EXIT:
        PostMessage(hWnd, WM_SYSCOMMAND, SC_CLOSE, 0L);
        break;

    case ID_RUN:
        if (is_running == FALSE)
        {
                if ((tse_OK == FALSE) && (tem_OK == FALSE))
                {
                        ErrMsg("Please load trainig set and template");
                        break;
                }
                else if (tse_OK == FALSE)
                {
                        ErrMsg("Please load trainig set");
                        break;
                }
                else if (tem_OK == FALSE)
                {
                        ErrMsg("Please load template");
                        break;
                }
                else
                {
                                                Deterministic();
                }
        }
        break;

    case ID_SETUP_LEARNNGRATE:
        ChangeLearningrate(hWndApp, hInst);
        break;
```

```
        case ID_SETUP_CYCLES:
            ChangeCycles(hWndApp, hInst);
            break;

        default:
            break;
    }

    return TRUE;
}
```

```
/*****************************************************************************
        Module : print.c

        Includes printing functions

*****************************************************************************/

#include <windows.h>
#include <string.h>
#include <stdlib.h>
#include "resource.h"

FARPROC  lpfnAbortProc   = NULL;
FARPROC  lpfnPrintDlgProc = NULL;
HWND     hWndParent      = NULL;
HWND     hDlgPrint       = NULL;
BOOL     bError;
BOOL     bUserAbort;
FARPROC  lpfnBetaProc    = NULL;
FARPROC  lpfnLearningProc = NULL;
FARPROC  lpfnCyclesProc   = NULL;


BOOL FAR PASCAL __export AbortProc (HDC, short);
BOOL FAR PASCAL __export PrintDlgProc (HWND, unsigned, WORD, DWORD);
BOOL FAR PASCAL __export LearningrateProc (HWND, unsigned, WORD, DWORD);
BOOL FAR PASCAL __export CyclesProc (HWND, unsigned, WORD, DWORD);

#pragma alloc_text(_PRINT, AbortProc, PrintDlgProc)

extern float learningrate;
extern int   cycle;

/*****************************************************************************
 *                                                                           *
 * FUNCTION   : GetPrinterDC()                                               *
 *                                                                           *
 * PURPOSE    : Read WIN.INI for default printer and create a DC for it.     *
 *                                                                           *
 * RETURNS    : A handle to the DC if successful or NULL otherwise.         *
 *                                                                           *
 *****************************************************************************/
HDC PASCAL GetPrinterDC()
{
   static char szPrinter [80];
   char   *szDevice, *szDriver, *szOutput;

   GetProfileString ("windows", "device", "", szPrinter, sizeof(szPrinter));

   if ((szDevice = strtok (szPrinter, "," )) &&
      (szDriver = strtok (NULL,     ", ")) &&
      (szOutput = strtok (NULL,     ", ")))

      return CreateDC (szDriver, szDevice, szOutput, NULL) ;

   return NULL;
}

/*****************************************************************************
```

```
*                                                                    *
* FUNCTION   : InitPrinting(HDC hDC, HWND hWnd, HANDLE hInst, LPSTR msg)  *
*                                                                    *
* PURPOSE    : Makes preliminary driver calls to set up print job.       *
*                                                                    *
* RETURNS    : TRUE  - if successful.                                *
*              FALSE - otherwise.                                    *
*                                                                    *
*********************************************************************/
BOOL PASCAL InitPrinting(HDC hDC, HWND hWnd, HANDLE hInst, LPSTR msg)
{

   bError    = FALSE;    /* no errors yet */
   bUserAbort = FALSE;    /* user hasn't aborted */

   hWndParent = hWnd;     /* save for Enable at Term time */

   lpfnPrintDlgProc = MakeProcInstance (PrintDlgProc, hInst);
   lpfnAbortProc    = MakeProcInstance (AbortProc, hInst);

   hDlgPrint = CreateDialog (hInst, "PRTDLG", hWndParent, lpfnPrintDlgProc);

   if (!hDlgPrint)
   return FALSE;

   SetWindowText (hDlgPrint, msg);
   EnableWindow (hWndParent, FALSE);       /* disable parent */

   if ((Escape (hDC, SETABORTPROC, 0, (LPSTR)lpfnAbortProc, NULL) > 0) &&
      (Escape (hDC, STARTDOC, lstrlen(msg), msg, NULL) > 0))
      bError = FALSE;
   else
      bError = TRUE;

   /* might want to call the abort proc here to allow the user to
    * abort just before printing begins */
   return TRUE;
}
/*********************************************************************
*                                                                    *
* FUNCTION   : TermPrinting(HDC hDC)                                 *
*                                                                    *
* PURPOSE    : Terminates print job.                                 *
*                                                                    *
*********************************************************************/
void PASCAL TermPrinting(HDC hDC)
{
   if (!bError)
      Escape(hDC, ENDDOC, 0, NULL, NULL);

   if (bUserAbort)
      Escape (hDC, ABORTDOC, 0, NULL, NULL) ;
   else {
      EnableWindow(hWndParent, TRUE);
      DestroyWindow(hDlgPrint);
   }

   FreeProcInstance(lpfnAbortProc);
   FreeProcInstance(lpfnPrintDlgProc);
}
```

```
/*************************************************************************
 *                                                                       *
 * FUNCTION   :PrintDlgProc (HWND, unsigned , WORD , DWORD )            *
 *                                                                       *
 * PURPOSE    :Dialog function for the "Cancel Printing" dialog. It sets *
 *             the abort flag if the user presses <Cancel>.              *
 *                                                                       *
 *************************************************************************/
BOOL FAR PASCAL __export PrintDlgProc (HWND hDlg, unsigned iMessage, WORD wParam,
DWORD lParam)
{

  switch (iMessage) {
  case WM_INITDIALOG:

       EnableMenuItem (GetSystemMenu (hDlg, FALSE), SC_CLOSE, MF_GRAYED);
       break;

  case WM_COMMAND:
       bUserAbort = TRUE;
       EnableWindow (hWndParent, TRUE);
       DestroyWindow (hDlg);
       hDlgPrint = 0;
       break;

  default:
       return FALSE;
  }
  return TRUE;

}

/*************************************************************************
 *                                                                       *
 * FUNCTION   :AbortProc (HDC hPrnDC, short nCode)                       *
 *                                                                       *
 * PURPOSE    :Checks message queue for messages from the "Cancel Printing"*
 *             dialog. If it sees a message, (this will be from a print  *
 *             cancel command), it terminates.                           *
 *                                                                       *
 * RETURNS    :Inverse of Abort flag                                     *
 *                                                                       *
 *************************************************************************/
BOOL FAR PASCAL __export AbortProc (HDC hPrnDC, short nCode)
{
  MSG  msg;

  while (!bUserAbort && PeekMessage (&msg, NULL, 0, 0, PM_REMOVE)) {
    if (!hDlgPrint || !IsDialogMessage(hDlgPrint, &msg)) {
       TranslateMessage (&msg);
       DispatchMessage (&msg);
    }
  }
  return !bUserAbort;
}

/*************************************************************************
 *                                                                       *
 * FUNCTION   : PrintDIB(HWND hWnd, HDC hDC, int x, int y, int dx, int dy)*
 *                                                                       *
```

```
* PURPOSE   : Set the DIB bits to the printer DC.                    *
*                                                                    *
*********************************************************************/
void PrintDIB (hdc, lpbi, x, y, dx, dy, x0, y0, dx0, dy0)
HDC hdc;
LPBITMAPINFO lpbi;
int x, y;
int dx, dy;
int x0, y0;
int dx0, dy0;

{
  LPSTR     pBuf;
  BOOL      f;

    /* Stretch the DIB to printer DC */

  pBuf = (LPSTR)lpbi + 1064;

  f = StretchDIBits ( hdc,
              x, y,
              dx, dy,
              x0, y0,
              dx0, dy0,
              pBuf, (LPBITMAPINFO)lpbi,
              DIB_RGB_COLORS,
              SRCCOPY);

}


/*********************************************************************
*                                                                    *
* FUNCTION  :LearningrateProc (HWND, unsigned , WORD , DWORD )      *
*                                                                    *
*********************************************************************/

BOOL FAR PASCAL __export LearningrateProc (HWND hDlg, unsigned iMessage, WORD wParam,
DWORD lParam)
{
        char *learn_val;
        int  ret_val;

 switch (iMessage)
 {
  case WM_INITDIALOG:           /* message: initialize dialog box */
   SetDlgItemText (hDlg,ID_LEARNING , ftoa(learningrate));
   return (TRUE);

  case WM_COMMAND:       /* message: received a command */
   switch (wParam)
   {
    case IDOK:
     ret_val = GetDlgItemText (hDlg,ID_LEARNING ,learn_val , 20);
     learningrate = (float)atof(learn_val);
    case IDCANCEL:
     EndDialog(hDlg, TRUE);  /* Exits the dialog box */
     break;
   }
   break;
```

```
   default:
    return FALSE;
  }

}

/*************************************************************************
 *                                                                       *
 * FUNCTION   :CyclesProc (HWND, unsigned , WORD , DWORD )                *
 *                                                                       *
 *************************************************************************/

BOOL FAR PASCAL __export CyclesProc (HWND hDlg, unsigned iMessage, WORD wParam,
DWORD lParam)
{
        char *cycle_val;
        char *buf;
        int  dec, sign, ret_val;

 switch (iMessage)
 {
  case WM_INITDIALOG:          /* message: initialize dialog box */
      buf = _fcvt(cycle, 6, &dec, &sign);
      strncpy(cycle_val, buf, dec);
      strcpy((cycle_val + dec), '\0');
      SetDlgItemText (hDlg,ID_CYCLE , cycle_val);
      return (TRUE);

  case WM_COMMAND:       /* message: received a command */
    switch (wParam)
    {
     case IDOK:
      ret_val = GetDlgItemText (hDlg,ID_CYCLE ,cycle_val , 20);
      cycle = atoi(cycle_val);
     case IDCANCEL:
      EndDialog(hDlg, TRUE);  /* Exits the dialog box */
      break;
    }
    break;
  default:
    return FALSE;
 }

}

BOOL ChangeLearningrate (HWND hwnd, HANDLE hInst)
{

 BOOL   bRet;

 lpfnLearningProc = MakeProcInstance (LearningrateProc, hInst);
 bRet = (BOOL) DialogBox(hInst, "LEARNINGDLG", hwnd, lpfnLearningProc);
 FreeProcInstance (lpfnLearningProc);
 return bRet;
}

BOOL ChangeCycles (HWND hwnd, HANDLE hInst)
{

 BOOL   bRet;
```

```
 lpfnCyclesProc = MakeProcInstance (CyclesProc, hInst);
 bRet = (BOOL) DialogBox(hInst, "CYCLEDLG", hwnd, lpfnCyclesProc);
 FreeProcInstance (CyclesProc);
 return bRet;
}
```

```
/*****************************************************************************
        Module : slatfunc.c

        Includes general functions

*****************************************************************************/

#include <windows.h>
#include "cnnslatw.h"

#include <time.h>
#include <math.h>
#include <string.h>
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>


float a[6],b[6],current,t_a[6],t_b[6],t_current;;
char algorithm_type[20],method_type[20],x_type[20],y_type[20];
char temp_path[20], image_path[20], result_path[20];
float learningrate;
int cycle;
float run_cycle,
last_cycle=1;

int Error[100],OLD_Error[100],Totalerror,OLD_Totalerror,Minerror=1000;
unsigned int Error_counter=0;
char save_file_name[30];
time_t first, second;
float time_step = (float)0.4,MAXDIF= (float)0.01;
float Treshold=(float)1.5;
int steady_state;
float beta;

struct str
                { char   ixdosya[25],
                         iudosya[25],
                         edosya[25];
                } training[100];

unsigned int ts_entry_quantity;
float minX;

char huge *bmp_U;
char huge *bmp_Xe;
char huge *bmp_Xy;
char huge *bmp_Y;
char huge *bmp_E;

char huge *bmp_Ubit;
char huge *bmp_Xebit;
char huge *bmp_Xybit;
char huge *bmp_Ybit;
char huge *bmp_Ebit;
char huge *bmp_fileheader;

float huge *U1;
float huge *U2;
float huge *U3;
```

```
float huge *U4;
float huge *U5;
float huge *U6;
float huge *U7;
float huge *U8;

float huge *Xe1;
float huge *Xe2;
float huge *Xe3;
float huge *Xe4;
float huge *Xe5;
float huge *Xe6;
float huge *Xe7;
float huge *Xe8;

float huge *Xy1;
float huge *Xy2;
float huge *Xy3;
float huge *Xy4;
float huge *Xy5;
float huge *Xy6;
float huge *Xy7;
float huge *Xy8;

float huge *Y1;
float huge *Y2;
float huge *Y3;
float huge *Y4;
float huge *Y5;
float huge *Y6;
float huge *Y7;
float huge *Y8;

float huge *E1;
float huge *E2;
float huge *E3;
float huge *E4;
float huge *E5;
float huge *E6;
float huge *E7;
float huge *E8;


HANDLE hdib_bmpU;
HANDLE hdib_bmpXe;
HANDLE hdib_bmpXy;
HANDLE hdib_bmpY;
HANDLE hdib_bmpE;
HANDLE hdib_fileheader;

HANDLE hU1;
HANDLE hU2;
HANDLE hU3;
HANDLE hU4;
HANDLE hU5;
HANDLE hU6;
HANDLE hU7;
HANDLE hU8;

HANDLE hXe1;
```

```
HANDLE hXe2;
HANDLE hXe3;
HANDLE hXe4;
HANDLE hXe5;
HANDLE hXe6;
HANDLE hXe7;
HANDLE hXe8;

HANDLE hXy1;
HANDLE hXy2;
HANDLE hXy3;
HANDLE hXy4;
HANDLE hXy5;
HANDLE hXy6;
HANDLE hXy7;
HANDLE hXy8;

HANDLE hY1;
HANDLE hY2;
HANDLE hY3;
HANDLE hY4;
HANDLE hY5;
HANDLE hY6;
HANDLE hY7;
HANDLE hY8;

HANDLE hE1;
HANDLE hE2;
HANDLE hE3;
HANDLE hE4;
HANDLE hE5;
HANDLE hE6;
HANDLE hE7;
HANDLE hE8;

unsigned int iteration_no;
BOOL user_stop=FALSE;
BOOL new_temp_tse = FALSE;
unsigned int image_no;
float maxdif_calc;

extern BOOL is_running;
extern HWND hWndApp;
extern BOOL tem_OK, tse_OK;
extern char achFileName[128];

#define DESIREDMESX  120
#define DESIREDMESY  5
#define DESIREDIMX   40
#define DESIREDIMY   20

#define ACTUALMESX   410
#define ACTUALMESY   5
#define ACTUALIMX    340
#define ACTUALIMY    20

#define TEMPMESX     100
#define TEMPMESY     285

#define TEMPRECT_L   5
```

```
#define TEMPRECT_U  300
#define TEMPRECT_R  TEMPRECT_L+255
#define TEMPRECT_D  TEMPRECT_U+130

#define TEMPPOSX_A  TEMPRECT_L+10
#define TEMPPOSY_A  TEMPRECT_U+10
#define TEMPPOSX_B  TEMPPOSX_A+120
#define TEMPPOSY_B  TEMPPOSY_A

#define ERRORMESX   320
#define ERRORMESY   285

#define ERRORRECT_L  TEMPRECT_R+5
#define ERRORRECT_U  TEMPRECT_U
#define ERRORRECT_R  ERRORRECT_L+165
#define ERRORRECT_D  TEMPRECT_D

#define LASTTOTX    ERRORRECT_L+10
#define LASTTOTY    ERRORRECT_U+10

#define STATRECT_L  ERRORRECT_R+5
#define STATRECT_U  TEMPRECT_U
#define STATRECT_R  ERRORRECT_R+195
#define STATRECT_D  TEMPRECT_D

#define STATX       STATRECT_L+10
#define STATY       STATRECT_U+10

/*****************************************************************************
 *                                                                           *
 * FUNCTION   : Deterministic(void)                                          *
 *                                                                           *
 * PURPOSE    : Runs deterministic calculation method                        *
 *                                                                           *
 *****************************************************************************/

int Deterministic(void)
{
  int simu_time, dec, sign;
  char *buf;

  if ( (user_stop == FALSE) || ((user_stop == TRUE) && (new_temp_tse == TRUE)))
  {
                iteration_no = 0;
                image_no    = 1;
  }

  is_running = TRUE;
  first = time(NULL);

  run_cycle=last_cycle;
  while ((run_cycle<=cycle) && (is_running == TRUE))
  {
   image_no = 1;

        if ( (OLD_Totalerror==Totalerror)&&(Totalerror) ) Error_counter=Error_counter+1 ;
        else  Error_counter=0;

        if( (run_cycle>=2) && (Totalerror<=Minerror) )
        {
```

```
            Minerror=Totalerror;
            auto_tempsave();
        }
        OLD_Totalerror=Totalerror;
        Totalerror=0;

        if( Error_counter==29 )
        {
          multiply_learningrate();
          Error_counter=0;
        }

show_stat();
        init_template();

        while ((image_no >0) && (image_no<=ts_entry_quantity) && (is_running == TRUE))
        {

          if ((main_loop() == FALSE) && (is_running == TRUE))
          {
            InfoMsg("Program Stopped");
            is_running = FALSE;
            tse_OK = FALSE;
                return FALSE;
          }
          if (is_running == TRUE)
          {
                if(update_template())
                {
                                init_template();
                                image_no=0;
                                decrease_learningrate();
                                Totalerror=0;
                        }

                        image_no++;
                }
        }

        if (is_running == TRUE)
        {
                last_template();
                show_template();
                write_error();

                if ((Totalerror==0))
                {
                                if( time_step > 1 )
                                {
                                time_step= (float)0.01;
                                InfoMsg("Test Mode Time-step = 0.01" );
                                }
                                else
                                {
                                second = time(NULL);
                                        simu_time = (int)difftime(second,first);
                                        buf = _fcvt(simu_time, 6, &dec, &sign);
                                        strcpy((buf + dec), "");
```

```
                                        PrintMsg(hWndApp, "Simu time (sec): ", LASTTOTX,
LASTTOTY+80);
                                        PrintMsg(hWndApp, buf, LASTTOTX+110,
LASTTOTY+80);

                        last_cycle = run_cycle;
                        is_running = FALSE;
                                }
                }
        }
        else last_cycle=run_cycle;

        if (is_running==TRUE) run_cycle++;
 }

 is_running = FALSE;
 InfoMsg("Program Stopped");
}

/*****************************************************************************
 *                                        *
 * FUNCTION   : main_loop (void)                                    *
 *                                        *
 * PURPOSE    :
*
 *                                        *
 *****************************************************************************/

int main_loop(void)
{
  HDC hDC;    // handle to device context

        if ( (user_stop == FALSE) || ((user_stop == TRUE) && (new_temp_tse == TRUE)))
  {
                user_stop = FALSE;
                new_temp_tse = FALSE;

                if (FileInputU(image_no) == FALSE) return FALSE ;
                if (FileInputX(image_no) == FALSE) return FALSE;
                if (FileInputE(image_no) == FALSE) return FALSE;
        }


        /* Display expected output */

        PrintMsg(hWndApp, "Desired Output",DESIREDMESX,DESIREDMESY);
        hDC = GetDC(hWndApp);
        SetDIBitsToDevice(hDC,
DESIREDIMX,DESIREDIMY,256,256,0,0,0,256,bmp_Ebit,(BITMAPINFO FAR
*)bmp_E,DIB_RGB_COLORS);
        ReleaseDC(hWndApp,hDC);

  show_template();
  show_error();
  show_stat();

  switch(x_type[0])
        {
                case 'e': stepya();
                                        TransferActualBits();
```

```
                                        PrintMsg(hWndApp, "Actual
Output",ACTUALMESX,ACTUALMESY);
                                        hDC = GetDC(hWndApp);
                                SetDIBitsToDevice(hDC,
ACTUALIMX,ACTUALIMY,256,256,0,0,0,256,bmp_Ybit,(BITMAPINFO FAR
*)bmp_Y,DIB_RGB_COLORS);
                                ReleaseDC(hWndApp,hDC);
                                do {
                                                stepx();
                                                if (is_running == FALSE) return
FALSE;
                                                exchange();
                                        stepya();
                                        TransferActualBits();

                                        iteration_no++;
                                                show_stat();

                                                /* Display actual output */
                                                hDC = GetDC(hWndApp);
                                        SetDIBitsToDevice(hDC,
ACTUALIMX,ACTUALIMY,256,256,0,0,0,256,bmp_Ybit,(BITMAPINFO FAR
*)bmp_Y,DIB_RGB_COLORS);
                                                ReleaseDC(hWndApp,hDC);

                                        } while ( (steady_state==0)||(minX<1)) ;
                                break;
                        default : break;
                }

    unstable_x();

  /* Display actual output */
        hDC = GetDC(hWndApp);
        TransferActualBits();
        SetDIBitsToDevice(hDC,ACTUALIMX,ACTUALIMY,256,256,0,0,0,256,bmp_Ybit,(BITM
APINFO FAR *)bmp_Y,DIB_RGB_COLORS);
        ReleaseDC(hWndApp,hDC);
  finderror();
        show_error();
}

/*************************************************************************
 *                                      *
 * FUNCTION   : stepx (void)                                            *
 *                                      *
 * PURPOSE    :
*
 *                                      *
 *************************************************************************/

int stepx(void)
{

    float dif;                          /*  ---> i  */
    unsigned int i,j;           /*  |       */
    steady_state=1;              /*  |       */
    minX=1;                     /*  v j     */
```

```
    maxdif_calc=0;

          for (j=1;j<33;j++)
    {
         for (i=1;i<=256;i++)
         {
                    *(Xy1+i+j*258)=  *(Xe1+i+j*258)
                                                +time_step*(  a[0]*(*(Y1+i+j*258))
                                                + a[1]*(*(Y1+i-1+j*258) + *(Y1+i+1+j*258))
                                                + a[2]*(*(Y1+i+(j-1)*258) +
*(Y1+i+(j+1)*258))
                                                + a[3]*(*(Y1+i-1+(j-1)*258) +
*(Y1+i+1+(j+1)*258))
                                                + a[4]*(*(Y1+i-1+(j+1)*258) + *(Y1+i+1+(j-
1)*258))
                                                + b[0]*(*(U1+i+j*258))
                                                + b[1]*(*(U1+i-1+j*258) + *(U1+i+1+j*258))
                                                + b[2]*(*(U1+i+(j-1)*258) +
*(U1+i+(j+1)*258))
                                                + b[3]*(*(U1+i-1+(j-1)*258) +
*(U1+i+1+(j+1)*258))
                                                + b[4]*(*(U1+i-1+(j+1)*258) + *(U1+i+1+(j-
1)*258))
                                                - *(Xe1+i+j*258) + current );

                    if (*(Xe1+i+j*258) != 0)
                    {
                      dif=(float)fabs(*(Xy1+i+j*258)- *(Xe1+i+j*258));
                      if ( dif> (maxdif_calc/fabs(*(Xe1+i+j*258))) )
maxdif_calc=(float)(dif/fabs(*(Xe1+i+j*258)));
                    }
                    if( (fabs(*(Xy1+i+j*258))-minX)<0 ) minX=(float)fabs(*(Xy1+i+j*258));

           }
           if (CheckMsgQueue()) return FALSE;
          }

    for (i=1;i<=256;i++)
    {
      *(Xe2+i) = *(Xe1+i+32*258);
    }

          for (j=1;j<33;j++)
    {
         for (i=1;i<=256;i++)
         {
                    *(Xy2+i+j*258)=  *(Xe2+i+j*258)
                                                +time_step*(  a[0]*(*(Y2+i+j*258))
                                                + a[1]*(*(Y2+i-1+j*258) + *(Y2+i+1+j*258))
                                                + a[2]*(*(Y2+i+(j-1)*258) +
*(Y2+i+(j+1)*258))
                                                + a[3]*(*(Y2+i-1+(j-1)*258) +
*(Y2+i+1+(j+1)*258))
                                                + a[4]*(*(Y2+i-1+(j+1)*258) + *(Y2+i+1+(j-
1)*258))
                                                + b[0]*(*(U2+i+j*258))
                                                + b[1]*(*(U2+i-1+j*258) + *(U2+i+1+j*258))
                                                + b[2]*(*(U2+i+(j-1)*258) +
*(U2+i+(j+1)*258))
```

```
                                                    + b[3]*(*(U2+i-1+(j-1)*258) +
*(U2+i+1+(j+1)*258))
                                                    + b[4]*(*(U2+i-1+(j+1)*258) + *(U2+i+1+(j-
1)*258))
                                                    - *(Xe2+i+j*258) + current );

                        if (*(Xe2+i+j*258) != 0)
                {
                  dif=(float)fabs(*(Xy2+i+j*258)- *(Xe2+i+j*258));
                  if ( dif> (maxdif_calc/fabs(*(Xe2+i+j*258))) )
maxdif_calc=(float)(dif/fabs(*(Xe2+i+j*258)));
                }
                if( (fabs(*(Xy2+i+j*258))-minX)<0 ) minX=(float)fabs(*(Xy2+i+j*258));

              }
              if (CheckMsgQueue()) return FALSE;
            }


    for (i=1;i<=256;i++)
    {
      *(Xe1+i+33*258) = *(Xe2+i+258);
    }

    for (i=1;i<=256;i++)
    {
      *(Xe3+i) = *(Xe2+i+32*258);
    }

          for (j=1;j<33;j++)
    {
        for (i=1;i<=256;i++)
        {
                  *(Xy3+i+j*258)=  *(Xe3+i+j*258)
                                                    +time_step*(  a[0]*(*(Y3+i+j*258))
                                                    + a[1]*(*(Y3+i-1+j*258) + *(Y3+i+1+j*258))
                                                    + a[2]*(*(Y3+i+(j-1)*258) +
*(Y3+i+(j+1)*258))
                                                    + a[3]*(*(Y3+i-1+(j-1)*258) +
*(Y3+i+1+(j+1)*258))
                                                    + a[4]*(*(Y3+i-1+(j+1)*258) + *(Y3+i+1+(j-
1)*258))
                                                    + b[0]*(*(U3+i+j*258))
                                                    + b[1]*(*(U3+i-1+j*258) + *(U3+i+1+j*258))
                                                    + b[2]*(*(U3+i+(j-1)*258) +
*(U3+i+(j+1)*258))
                                                    + b[3]*(*(U3+i-1+(j-1)*258) +
*(U3+i+1+(j+1)*258))
                                                    + b[4]*(*(U3+i-1+(j+1)*258) + *(U3+i+1+(j-
1)*258))
                                                    - *(Xe3+i+j*258) + current );

                if (*(Xe3+i+j*258) != 0)
                {
                  dif=(float)fabs(*(Xy3+i+j*258)- *(Xe3+i+j*258));
                  if ( dif> (maxdif_calc/fabs(*(Xe3+i+j*258))) )
maxdif_calc=(float)(dif/fabs(*(Xe3+i+j*258)));
                }
                if( (fabs(*(Xy3+i+j*258))-minX)<0 ) minX=(float)fabs(*(Xy3+i+j*258));
```

```
            }
            if (CheckMsgQueue()) return FALSE;
          }


    for (i=1;i<=256;i++)
    {
      *(Xe2+i+33*258) = *(Xe3+i+258);
    }

    for (i=1;i<=256;i++)
    {
      *(Xe4+i) = *(Xe3+i+32*258);
    }

          for (j=1;j<33;j++)
    {
        for (i=1;i<=256;i++)
        {
                  *(Xy4+i+j*258)=  *(Xe4+i+j*258)
                                          +time_step*(  a[0]*(*(Y4+i+j*258))
                                          + a[1]*(*(Y4+i-1+j*258) + *(Y4+i+1+j*258))
                                          + a[2]*(*(Y4+i+(j-1)*258) +
*(Y4+i+(j+1)*258))
                                          + a[3]*(*(Y4+i-1+(j-1)*258) +
*(Y4+i+1+(j+1)*258))
                                          + a[4]*(*(Y4+i-1+(j+1)*258) + *(Y4+i+1+(j-
1)*258))
                                          + b[0]*(*(U4+i+j*258))
                                          + b[1]*(*(U4+i-1+j*258) + *(U4+i+1+j*258))
                                          + b[2]*(*(U4+i+(j-1)*258) +
*(U4+i+(j+1)*258))
                                          + b[3]*(*(U4+i-1+(j-1)*258) +
*(U4+i+1+(j+1)*258))
                                          + b[4]*(*(U4+i-1+(j+1)*258) + *(U4+i+1+(j-
1)*258))
                                          - *(Xe4+i+j*258) + current );

                  if (*(Xe4+i+j*258) != 0)
                  {
                    dif=(float)fabs(*(Xy4+i+j*258)- *(Xe4+i+j*258));
                    if ( dif> (maxdif_calc/fabs(*(Xe4+i+j*258))) )
maxdif_calc=(float)(dif/fabs(*(Xe4+i+j*258)));
                  }
                  if( (fabs(*(Xy4+i+j*258))-minX)<0 ) minX=(float)fabs(*(Xy4+i+j*258));

            }
            if (CheckMsgQueue()) return FALSE;
          }


    for (i=1;i<=256;i++)
    {
      *(Xe3+i+33*258) = *(Xe4+i+258);
    }

    for (i=1;i<=256;i++)
    {
      *(Xe5+i) = *(Xe4+i+32*258);
    }
```

```
        for (j=1;j<33;j++)
    {
        for (i=1;i<=256;i++)
        {
                *(Xy5+i+j*258)=  *(Xe5+i+j*258)
                                        +time_step*(  a[0]*(*(Y5+i+j*258))
                                        + a[1]*(*(Y5+i-1+j*258) + *(Y5+i+1+j*258))
                                        + a[2]*(*(Y5+i+(j-1)*258) +
*(Y5+i+(j+1)*258))
                                        + a[3]*(*(Y5+i-1+(j-1)*258) +
*(Y5+i+1+(j+1)*258))
                                        + a[4]*(*(Y5+i-1+(j+1)*258) + *(Y5+i+1+(j-
1)*258))
                                        + b[0]*(*(U5+i+j*258))
                                        + b[1]*(*(U5+i-1+j*258) + *(U5+i+1+j*258))
                                        + b[2]*(*(U5+i+(j-1)*258) +
*(U5+i+(j+1)*258))
                                        + b[3]*(*(U5+i-1+(j-1)*258) +
*(U5+i+1+(j+1)*258))
                                        + b[4]*(*(U5+i-1+(j+1)*258) + *(U5+i+1+(j-
1)*258))
                                        - *(Xe5+i+j*258) + current );

                if (*(Xe5+i+j*258) != 0)
                {
                  dif=(float)fabs(*(Xy5+i+j*258)- *(Xe5+i+j*258));
                  if ( dif> (maxdif_calc/fabs(*(Xe5+i+j*258))) )
maxdif_calc=(float)(dif/fabs(*(Xe5+i+j*258)));
                }
                if( (fabs(*(Xy5+i+j*258))-minX)<0 ) minX=(float)fabs(*(Xy5+i+j*258));

          }
          if (CheckMsgQueue()) return FALSE;
        }


    for (i=1;i<=256;i++)
    {
      *(Xe4+i+33*258) = *(Xe5+i+258);
    }

    for (i=1;i<=256;i++)
    {
      *(Xe6+i) = *(Xe5+i+32*258);
    }


        for (j=1;j<33;j++)
    {
        for (i=1;i<=256;i++)
        {
                *(Xy6+i+j*258)=  *(Xe6+i+j*258)
                                        +time_step*(  a[0]*(*(Y6+i+j*258))
                                        + a[1]*(*(Y6+i-1+j*258) + *(Y6+i+1+j*258))
                                        + a[2]*(*(Y6+i+(j-1)*258) +
*(Y6+i+(j+1)*258))
                                        + a[3]*(*(Y6+i-1+(j-1)*258) +
*(Y6+i+1+(j+1)*258))
```

```
                                                + a[4]*(*(Y6+i-1+(j+1)*258) + *(Y6+i+1+(j-
1)*258))
                                                + b[0]*(*(U6+i+j*258))
                                                + b[1]*(*(U6+i-1+j*258) + *(U6+i+1+j*258))
                                                + b[2]*(*(U6+i+(j-1)*258) +
*(U6+i+(j+1)*258))
                                                + b[3]*(*(U6+i-1+(j-1)*258) +
*(U6+i+1+(j+1)*258))
                                                + b[4]*(*(U6+i-1+(j+1)*258) + *(U6+i+1+(j-
1)*258))
                                                - *(Xe6+i+j*258) + current );

                    if (*(Xe6+i+j*258) != 0)
                    {
                      dif=(float)fabs(*(Xy6+i+j*258)- *(Xe6+i+j*258));
                      if ( dif> (maxdif_calc/fabs(*(Xe6+i+j*258))) )
maxdif_calc=(float)(dif/fabs(*(Xe6+i+j*258)));
                    }
                    if( (fabs(*(Xy6+i+j*258))-minX)<0 ) minX=(float)fabs(*(Xy6+i+j*258));

                }
                if (CheckMsgQueue()) return FALSE;
              }


    for (i=1;i<=256;i++)
    {
      *(Xe5+i+33*258) = *(Xe6+i+258);
    }

    for (i=1;i<=256;i++)
    {
      *(Xe7+i) = *(Xe6+i+32*258);
    }


          for (j=1;j<33;j++)
    {
        for (i=1;i<=256;i++)
        {
                    *(Xy7+i+j*258)=  *(Xe7+i+j*258)
                                                +time_step*(  a[0]*(*(Y7+i+j*258))
                                                + a[1]*(*(Y7+i-1+j*258) + *(Y7+i+1+j*258))
                                                + a[2]*(*(Y7+i+(j-1)*258) +
*(Y7+i+(j+1)*258))
                                                + a[3]*(*(Y7+i-1+(j-1)*258) +
*(Y7+i+1+(j+1)*258))
                                                + a[4]*(*(Y7+i-1+(j+1)*258) + *(Y7+i+1+(j-
1)*258))
                                                + b[0]*(*(U7+i+j*258))
                                                + b[1]*(*(U7+i-1+j*258) + *(U7+i+1+j*258))
                                                + b[2]*(*(U7+i+(j-1)*258) +
*(U7+i+(j+1)*258))
                                                + b[3]*(*(U7+i-1+(j-1)*258) +
*(U7+i+1+(j+1)*258))
                                                + b[4]*(*(U7+i-1+(j+1)*258) + *(U7+i+1+(j-
1)*258))
                                                - *(Xe7+i+j*258) + current );

                    if (*(Xe7+i+j*258) != 0)
```

```
                {
                  dif=(float)fabs(*(Xy7+i+j*258)- *(Xe7+i+j*258));
                  if ( dif> (maxdif_calc/fabs(*(Xe7+i+j*258))) )
maxdif_calc=(float)(dif/fabs(*(Xe7+i+j*258)));
                }
                if( (fabs(*(Xy7+i+j*258))-minX)<0 ) minX=(float)fabs(*(Xy7+i+j*258));

            }
            if (CheckMsgQueue()) return FALSE;
          }


    for (i=1;i<=256;i++)
    {
      *(Xe6+i+33*258) = *(Xe7+i+258);
    }

    for (i=1;i<=256;i++)
    {
      *(Xe8+i) = *(Xe7+i+32*258);
    }

          for (j=1;j<33;j++)
    {
        for (i=1;i<=256;i++)
        {
                  *(Xy8+i+j*258)=  *(Xe8+i+j*258)
                                          +time_step*(  a[0]*(*(Y8+i+j*258))
                                          + a[1]*(*(Y8+i-1+j*258) + *(Y8+i+1+j*258))
                                          + a[2]*(*(Y8+i+(j-1)*258) +
*(Y8+i+(j+1)*258))
                                          + a[3]*(*(Y8+i-1+(j-1)*258) +
*(Y8+i+1+(j+1)*258))
                                          + a[4]*(*(Y8+i-1+(j+1)*258) + *(Y8+i+1+(j-
1)*258))
                                          + b[0]*(*(U8+i+j*258))
                                          + b[1]*(*(U8+i-1+j*258) + *(U8+i+1+j*258))
                                          + b[2]*(*(U8+i+(j-1)*258) +
*(U8+i+(j+1)*258))
                                          + b[3]*(*(U8+i-1+(j-1)*258) +
*(U8+i+1+(j+1)*258))
                                          + b[4]*(*(U8+i-1+(j+1)*258) + *(U8+i+1+(j-
1)*258))
                                          - *(Xe8+i+j*258) + current );

                if (*(Xe8+i+j*258) != 0)
                {
                  dif=(float)fabs(*(Xy8+i+j*258)- *(Xe8+i+j*258));
                  if ( dif> (maxdif_calc/fabs(*(Xe8+i+j*258))) )
maxdif_calc=(float)(dif/fabs(*(Xe8+i+j*258)));
                }
                if( (fabs(*(Xy8+i+j*258))-minX)<0 ) minX=(float)fabs(*(Xy8+i+j*258));

            }
            if (CheckMsgQueue()) return FALSE;
          }


    for (i=1;i<=256;i++)
    {
```

```
        *(Xe7+i+33*258) = *(Xe8+i+258);
    }


  if (maxdif_calc>MAXDIF) steady_state=0;

}


/*****************************************************************************
 *                                                                           *
 * FUNCTION   : stepya (void)                                                *
 *                                                                           *
 * PURPOSE    :
*
 *                                                                           *
 *****************************************************************************/

void stepya(void)
{
    unsigned int i,j;

  for (j=1;j<33;j++)
  {
          for (i=1;i<=256;i++)
          {
                  *(Y1+i+j*258) = (float)0.5*( (float)fabs(*(Xe1+i+j*258)+1)-
(float)fabs(*(Xe1+i+j*258)-1) );
                  *(Y2+i+j*258) = (float)0.5*( (float)fabs(*(Xe2+i+j*258)+1)-
(float)fabs(*(Xe2+i+j*258)-1) );
                  *(Y3+i+j*258) = (float)0.5*( (float)fabs(*(Xe3+i+j*258)+1)-
(float)fabs(*(Xe3+i+j*258)-1) );
                  *(Y4+i+j*258) = (float)0.5*( (float)fabs(*(Xe4+i+j*258)+1)-
(float)fabs(*(Xe4+i+j*258)-1) );
                  *(Y5+i+j*258) = (float)0.5*( (float)fabs(*(Xe5+i+j*258)+1)-
(float)fabs(*(Xe5+i+j*258)-1) );
                  *(Y6+i+j*258) = (float)0.5*( (float)fabs(*(Xe6+i+j*258)+1)-
(float)fabs(*(Xe6+i+j*258)-1) );
                  *(Y7+i+j*258) = (float)0.5*( (float)fabs(*(Xe7+i+j*258)+1)-
(float)fabs(*(Xe7+i+j*258)-1) );
                  *(Y8+i+j*258) = (float)0.5*( (float)fabs(*(Xe8+i+j*258)+1)-
(float)fabs(*(Xe8+i+j*258)-1) );

                  if ((*(Y1+i+j*258)) == -1) *(Y1+i+j*258) = (float) -0.992188;
                  if ((*(Y2+i+j*258)) == -1) *(Y2+i+j*258) = (float) -0.992188;
                  if ((*(Y3+i+j*258)) == -1) *(Y3+i+j*258) = (float) -0.992188;
                  if ((*(Y4+i+j*258)) == -1) *(Y4+i+j*258) = (float) -0.992188;
                  if ((*(Y5+i+j*258)) == -1) *(Y5+i+j*258) = (float) -0.992188;
                  if ((*(Y6+i+j*258)) == -1) *(Y6+i+j*258) = (float) -0.992188;
                  if ((*(Y7+i+j*258)) == -1) *(Y7+i+j*258) = (float) -0.992188;
                  if ((*(Y8+i+j*258)) == -1) *(Y8+i+j*258) = (float) -0.992188;
   }
  }

  for (i=1; i<=256; i++)
        {
          *(Y1+i+33*258) = *(Y2+i+258);
          *(Y2+i)        = *(Y1+i+32*258);

          *(Y2+i+33*258) = *(Y3+i+258);
```

```
            *(Y3+i)      = *(Y2+i+32*258);

            *(Y3+i+33*258) = *(Y4+i+258);
            *(Y4+i)      = *(Y3+i+32*258);

            *(Y4+i+33*258) = *(Y5+i+258);
            *(Y5+i)      = *(Y4+i+32*258);

            *(Y5+i+33*258) = *(Y6+i+258);
            *(Y6+i)      = *(Y5+i+32*258);

            *(Y6+i+33*258) = *(Y7+i+258);
            *(Y7+i)      = *(Y6+i+32*258);

            *(Y7+i+33*258) = *(Y8+i+258);
            *(Y8+i)      = *(Y7+i+32*258);
    }

}

/*************************************************************************
 *                                                                       *
 * FUNCTION   : exchange (void)                                          *
 *                                                                       *
 * PURPOSE    :
*
 *                                                                       *
 *************************************************************************/

void exchange(void)
{
    unsigned int i,j;

    for (j=1;j<33;j++)
        {
            for (i=1;i<=256;i++)
            {
                *(Xe1+i+j*258)= *(Xy1+i+j*258);
                *(Xe2+i+j*258)= *(Xy2+i+j*258);
                *(Xe3+i+j*258)= *(Xy3+i+j*258);
                *(Xe4+i+j*258)= *(Xy4+i+j*258);
                *(Xe5+i+j*258)= *(Xy5+i+j*258);
                *(Xe6+i+j*258)= *(Xy6+i+j*258);
                *(Xe7+i+j*258)= *(Xy7+i+j*258);
                *(Xe8+i+j*258)= *(Xy8+i+j*258);
            }
        }


  for (i=1; i<=256; i++)
      {
        *(Xe1+i+33*258) = *(Xe2+i+258);
        *(Xe2+i)      = *(Xe1+i+32*258);

        *(Xe2+i+33*258) = *(Xe3+i+258);
        *(Xe3+i)      = *(Xe2+i+32*258);

        *(Xe3+i+33*258) = *(Xe4+i+258);
        *(Xe4+i)      = *(Xe3+i+32*258);
```

```
            *(Xe4+i+33*258) = *(Xe5+i+258);
            *(Xe5+i)      = *(Xe4+i+32*258);

            *(Xe5+i+33*258) = *(Xe6+i+258);
            *(Xe6+i)      = *(Xe5+i+32*258);

            *(Xe6+i+33*258) = *(Xe7+i+258);
            *(Xe7+i)      = *(Xe6+i+32*258);

            *(Xe7+i+33*258) = *(Xe8+i+258);
            *(Xe8+i)      = *(Xe7+i+32*258);
    }
}

/*****************************************************************************
 *                                                                           *
 * FUNCTION   : TransferActualBits (void)                                    *
 *                                                                           *
 * PURPOSE    :
*
 *                                                                           *
 *****************************************************************************/

void TransferActualBits(void)
{
        unsigned int                h,w,k;
   unsigned char huge *temp;
   float                            value;
   int which;

        for (h=256; h>224; h--)
   {
     temp = bmp_Ybit + (h-1)*256;
     k = 257 - h;
                for (w=1; w<=256; w++)
           {

             value =(float) ( 128.0 * (*(Y1+w+k*258)));

                    if (value<=0) which = (int) floor(value);
                    else        which = (int) ceil(value);

                    if (which == -128) which = -127;

                    *temp = 128-which;
          temp++;

        }
   }

        for (h=224; h>192; h--)
   {
     temp = bmp_Ybit + (h-1)*256;
     k = 225 - h;
                for (w=1; w<=256; w++)
           {

             value =(float) ( 128.0 * (*(Y2+w+k*258)));

                    if (value<=0) which = (int) floor(value);
```

```
                else        which = (int) ceil(value);

                if (which == -128) which = -127;

                *temp = 128-which;
        temp++;

        }
}

for (h=192; h>160; h--)
{
   temp = bmp_Ybit + (h-1)*256;
   k = 193 - h;
                for (w=1; w<=256; w++)
          {

            value =(float) ( 128.0 * (*(Y3+w+k*258)));

                if (value<=0) which = (int) floor(value);
                else        which = (int) ceil(value);

                if (which == -128) which = -127;

                *temp = 128-which;
        temp++;

        }
}

for (h=160; h>128; h--)
{
   temp = bmp_Ybit + (h-1)*256;
   k = 161 - h;
                for (w=1; w<=256; w++)
          {

            value =(float) ( 128.0 * (*(Y4+w+k*258)));

                if (value<=0) which = (int) floor(value);
                else        which = (int) ceil(value);

                if (which == -128) which = -127;

                *temp = 128-which;
        temp++;

        }
}

for (h=128; h>96; h--)
{
   temp = bmp_Ybit + (h-1)*256;
   k = 129 - h;
                for (w=1; w<=256; w++)
          {

            value =(float) ( 128.0 * (*(Y5+w+k*258)));

                if (value<=0) which = (int) floor(value);
```

```
                else        which = (int) ceil(value);

                if (which == -128) which = -127;

                *temp = 128-which;
        temp++;

        }
}

for (h=96; h>64; h--)
{
   temp = bmp_Ybit + (h-1)*256;
   k = 97 - h;
                for (w=1; w<=256; w++)
          {

           value =(float) ( 128.0 * (*(Y6+w+k*258)));

                if (value<=0) which = (int) floor(value);
                else        which = (int) ceil(value);

                if (which == -128) which = -127;

                *temp = 128-which;
        temp++;

        }
}

for (h=64; h>32; h--)
{
   temp = bmp_Ybit + (h-1)*256;
   k = 65 - h;
                for (w=1; w<=256; w++)
          {

           value =(float) ( 128.0 * (*(Y7+w+k*258)));

                if (value<=0) which = (int) floor(value);
                else        which = (int) ceil(value);

                if (which == -128) which = -127;

                *temp = 128-which;
        temp++;

        }
}

for (h=32; h>0; h--)
{
   temp = bmp_Ybit + (h-1)*256;
   k = 33 - h;
                for (w=1; w<=256; w++)
          {

           value =(float) ( 128.0 * (*(Y8+w+k*258)));

                if (value<=0) which = (int) floor(value);
```

```
                else        which = (int) ceil(value);

                if (which == -128) which = -127;

                *temp = 128-which;
          temp++;

        }
   }
}

/*****************************************************************************
 *                                                *
 * FUNCTION   : auto_tempsave(void)                          *
 *                                                *
 * PURPOSE    : Saves template file
                        *
 *                                                *
 *****************************************************************************/


void auto_tempsave(void)
{


    strcpy(save_file_name,"min.tem");
    write_template();
    if ( Totalerror==0 )
          {
          strcpy(save_file_name,"last.tem");
          write_template();
          }
}

/*****************************************************************************
 *                                                *
 * FUNCTION   : write_template(void)                         *
 *                                                *
 * PURPOSE    : Writes template file
                        *
 *                                                *
 *****************************************************************************/


void write_template(void)
{
   FILE *image;
   char buffer[100];

   strcpy(buffer,temp_path);
   strcat(buffer,save_file_name);
   if ((image = fopen(buffer,"w")) == NULL)
   {
            ErrMsg("Can not open '%ls'", save_file_name);
   }

    fprintf(image,"Neighborhood: 1\n\n ");

    fprintf(image,"Feedback: \n");
    fprintf(image,"%8.6f %8.6f %8.6f\n",a[3],a[2],a[4]);
```

```
    fprintf(image,"%8.6f %8.6f %8.6f\n",a[1],a[0],a[1]);
    fprintf(image,"%8.6f %8.6f %8.6f\n\n",a[4],a[2],a[3]);

    fprintf(image,"Current: %8.6f\n\n",current);

    fprintf(image,"Control: \n");
    fprintf(image,"%8.6f %8.6f %8.6f\n",b[3],b[2],b[4]);
    fprintf(image,"%8.6f %8.6f %8.6f\n",b[1],b[0],b[1]);
    fprintf(image,"%8.6f %8.6f %8.6f\n",b[4],b[2],b[3]);

    second = time(NULL);
    fprintf(image," \n\n Simulation time:  %d seconds", (int)difftime(second,first));
    fprintf(image," \n\n Totalerror:  %d ", Totalerror);

    fclose(image);
}

/*****************************************************************************
 *                                                                           *
 * FUNCTION   : multiply_learningrate(void)                                  *
 *                                                                           *
 * PURPOSE    : Multiplies learning rate by 10
                *
 *                                                                           *
 *****************************************************************************/


void multiply_learningrate(void)
{
    //InfoMsg("Learningrate multiplication occured");
    learningrate=learningrate*10;
    if (learningrate > 1 ) learningrate=(float)ceil(learningrate);
}

/*****************************************************************************
 *                                                                           *
 * FUNCTION   : decrease_learningrate(void)                                  *
 *                                                                           *
 * PURPOSE    : Multiplies learning rate by 0.1
         *
 *                                                                           *
 *****************************************************************************/

void decrease_learningrate(void)
{
    //InfoMsg("Template variation overflow");
    learningrate=(float)(learningrate*0.1);
}
```

ÖZGEÇMİŞ

Filiz YOSMA TAŞKIN, 1968 İstanbul doğumludur. 1986 yılında Yalova Lisesi'ni bitirerek İstanbul Teknik Üniversitesi Elektrik Elektronik Fakültesi Elektronik ve Haberleşme Bölümü'nde lisans eğitimine başlamıştır. 1990 yılında bu bölümden mezun olarak Erde Elektrik A.Ş.'de yazılım mühendisi olarak çalışmaya başlamıştır. 1991'de İ.T.Ü. Fen Bilimleri Enstitüsü'nde yüksek lisans eğitimine başlamıştır. Nisan 1991'den Kasım 1994'e kadar Alcatel Teletaş A.Ş.'de araştırma-geliştirme bölümünde yazılımcı olarak çalışmıştır. Kasım 1994'ten bu yana Türk Hava Yolları A.O. Simülator Bölümünde simülatör mühendisi olarak çalışmaktadır.