T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MODEL TABANLI YAPAY SİNİR AĞLARI KULLANILARAK GÖRÜNTÜ KENARININ YAKALANMASI

Fatih TALU

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ELAZIĞ, 2004

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MODEL TABANLI YAPAY SİNİR AĞLARI KULLANILARAK GÖRÜNTÜ KENARININ YAKALANMASI

Fatih TALU

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Bu tez, 09 / 02 / 2004 tarihinde, aşağıda belirtilen jüri tarafından oybirliği /~~ oyçokluğu~~ ile başarılı / ~~başarısız~~ olarak değerlendirilmiştir.

Danışman:Yrd.Doç.Dr. Yetkin TATAR
Üye: Doç.Dr. Erhan AKIN
Üye: Yrd.Doç.Dr. İbrahim TÜRKOĞLU
Bu tezin kabulü, Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 13../ 02../ 2004 tarih ve ..... 64 ..... sayılı kararıyla onaylanmıştır.

TEŞEKKÜR

Bu tez çalışmam boyunca, ilgi ve yardımlarını esirgemeyen danışmanım Sayın Yrd.Doç.Dr. Yetkin TATAR'a, çalışmam esnasında değerli fikir ve yardımlarından yararlandığım Yrd.Doç.Dr. İbrahim TÜRKOĞLU'na ve Arş.Gör. Bilal Alataş'a, ve tezin düzenlenmesindeki yardımlarından ötürü Arş.Gör. Ahmet TEKİN'e teşekkürlerimi ve şükranlarımı sunarım.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

SİMGELER LİSTESİ

W	:Maskeleme katsayısı
u	:Yatay maskeleme sonucu
v	:Düşey maskeleme sonucu
b(i,j)	:Euclidian uzaklığı
d(i,j)	:Eğim yönü
z	:Noktanın gri seviye değeri
T	:Eşik değeri
F(j,k)	:Orijinal görüntü
Gr(j,k)	:Yatay farklılık yakalayıcı
Gc(j,k)	:Düşey farklılık yakalayıcı
G(j,k)	:Farklılık birleştirici
p(x,y)	:Güncel nokta değeri
Ç	:Klasik kenar yakalayıcı sonucu
R	:Referans değeri
α	:Filtre katsayısı
θ	:Değişim açısı
$\nabla^2 f$	:Laplacian operatörü
SRN	:Gürültü sinyal ortalaması
I	:Görüntü matrisi
Σ	:Gaussian fonksiyonun yayılım değeri
G(σ)	:Gaussian yumuşatma filtresi
S[i,j]	:Yumuşatılmış görüntü
P[i,j]	:Yatay kısmi türev
Q[i,j]	:Düşey kısmi türev
M[i,j]	:Eğilimin büyüklüğü
θ[i,j]	:Eğilimin yönü
nms(.)	:Non-maksimum baskısı
N[i,j]	:Non-maksimum baskısı sonucu
x	:Yüksek boyutlu giriş vektörü
x^p	:Düşük boyutlu giriş vektörü

p	:Büyük boyutlu ağırlık vektörü
z	:Küçük boyutlu ağırlık vektörü
$fs(x,z)$	:Doğrusal olmayan fonksiyon
$p_{s^*_r}$	:Kazanan hücrenin indeksi
p_r	:Ağın parlaklık değeri
$\phi(x, p_r)$	:Alt ağın çıkışı
$\bar{x}$	:Pencerenin ortalama değeri
m_1	:Ortalamadan büyük olan noktaların ortalama değeri
m_2	:Ortalamadan küçük olan noktaların ortalama değeri
$\bar{m}$	:m_1 ve m_2'nin ortalaması
U_{r^*}	:*r.* alt ağ
$N_{r,s}$	:Ağ içerisindeki hücrenin indeksi
$w_{r,s}$	:*r.* ağın *s.* hücresinin ağırlık vektörü
$w^u{}_{r^*}$	:Kuvvet kenar örneğinin ağırlığı
$w^l{}_{r^*}$	:Zayıf kenar örneğinin ağırlığı
V_d	:Dinamik kenar izleme hücresi
w_d	:Dinamik ağırlık vektörü
p_d	:Dinamik parlaklık değeri
b	:İkili kenar örneği
$Q(x, w_{r^*s^*})$	:İki seviyeye dönüştürme fonksiyonu
C	:Kenar konfigürasyon kümesi
S_r	:Alt ağın hücre sayısı
$R\pi/4$	:45^0'lik döndürme operatörü
t_f	:Toplam iterasyon sayısı
$\eta(t)$	:*t.* andaki öğrenme oranı
m_p	:Dinamik hücre girişi
itn	:İterasyon sayısı
lr	:Öğrenme oranı

KISALTMALAR LİSTESİ

YSA	:Yapay sinir ağı
MYSA	:Model tabanlı yapay sinir ağı
KYA	:Kenar yakalama algoritması
YYA	:Yakalanan yön açısı
KBA	:Kullanılan vektörün başlangıç açısı
LOG	:Gaussian Filtreli Laplacian Operatörü

ÖZET

Yüksek Lisans Tezi

MODEL TABANLI YAPAY SİNİR AĞLARI KULLANILARAK GÖRÜNTÜ KENARININ YAKALANMASI

Fatih TALU

Fırat Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

2004, Sayfa:71

Bu çalışmada, iki seviyeli ve gri seviyeli görüntülerde arzu edilen kenar sonucuna ulaşabilmek için Matlab 6.5 programlama dili kullanılarak iki tane paket program geliştirilmiştir. Her iki paket programa giriş görüntüsü olarak yapay ve doğal görüntüler verilmiştir. İki seviyeli görüntüler için geliştirilen paket program, klasik kenar yakalama algoritmalarından daha hızlı ve daha az karşılaştırma yaparak arzu edilen kenar sonucuna ulaşabilmektedir. Gri seviyeli görüntüler için hazırlanmış paket program ise model tabanlı yapay sinir ağı mimarisini kullanarak kenar yakalama problemine etkili ve makul bir çözüm önerisi getirmektedir.

Anahtar Kelimeler: Kenar yakalama, görüntü işleme, örüntü tanıma, model tabanlı yapay sinir ağı

ABSTRACT

IMAGE EDGE DETECTION USING MODEL BASED ARTIFICIAL NEURAL NETWORK

Fatih TALU

Firat University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering
2004, Page:71

In this thesis, two package programs have been developed for reaching to the desired edge results in binary and gray level image using Matlab 6.5 programming language. Artificial and natural images might be selected for input to both of the package programs. Package program developed for binary level image finds the desired edges more rapidly by making less comparison than the traditional edge detection algorithms do. Package program developed for gray level image presents an efficient and reasonable solution to the edge detection problem by using model based artificial neural network architecture.

Key Words: Edge Detection, image processing, object recognition, model-based artificial neural network

1. GİRİŞ

Hızla gelişmekte olan teknoloji, bilim adamlarının kusursuz bir model olan insanın düşünebilme ve yorumlayabilme özelliklerini eksiksiz bir şekilde taklit edebilme ümidini arttırmaktadır. Bu nedenle, insandan başka hiçbir canlıda olmayan bu özelliklerin gelişimi, onun doğumundan ölümüne kadarki her evrede ilgi ile izlenmesini ve yıllarca benzerini yapma konusunda bir çok çalışmaların yapılmasını sağlamıştır.

İnsan görüntü algılama sistemi, görüntü işleme açısından ele alındığında; görüntü yakalama, gruplama ve analiz konusunda bilinen en karmaşık sistemdir. İnsanın çevresindeki nesnelerin geometrik şekillerini tanıması, bu nesneler hakkındaki belirgin ilişkiyi kavraması ve duruma uygun cevap vermesi için nesnelere ait görüntü bilgisine ihtiyacı vardır. "Bir resim bin kelimeden daha kıymetlidir" sözü, görüntü bilgisinin diğer bilgilere oranla daha kalıcı ve düşünebilme ve yorumlayabilme fonksiyonlarının daha etkili kullanıldığını ifade eden bir örnektir.

Görüntü bir nesnenin sunumudur. Görüntü bilgisi günümüzde bir çok alanda kullanılır. Örneğin; tıp alanında, biyolojik materyallerde, endüstride, parmak izi veya yüz tanıma gibi kriminolojik uygulamalarda, uzaysal çalışmalarda ve askeri uygulamalarda görüntü bilgilerinin sıkça kullanıldığını görmek mümkündür. Tarayıcılar, kameralar v.b. gibi algılayıcı sistemler kullanılarak görüntü bilgisi sayısal sistemlere aktarılır ve matris formatında depolanır.

Bir görüntünün insan tarafından algılanması ve yorumlanması birkaç aşamadan meydana gelir. Öncelikle göze gelen ışık ışınları korneadan, gözbebeğinden ve ardından da mercekten geçer. Saydam tabakanın bükümlü üst yüzeyi ve mercek, ışınları kırar ve nesnenin (resmin) görüntüsü ters çevrildikten sonra retinaya ulaşır. Daha sonra ışığa duyarlı hücreler (reseptörler; koni ve çubuk hücreler) ışığı elektrik sinyallerine çevirir ve sinir uçlarına uyarı olarak yollarlar. Retinadan gelen görüntü orijinaline göre baş aşağı durumda ve ters taraftadır. Bu elektriksel uyarılar beyne nesnenin çeşidi, büyüklüğü, rengi, uzaklığı hakkında bilgi götürürler ve tüm bu dizi işlemler saniyenin onda biri kadar bir sürede gerçekleşir. Ancak beyin, kendisine gelen bu sinyallerin genliğini, fazını, frekansını değerlendirip görüntü hakkında tüm bilgiyi elde eder ve

insanın o nesneyi doğru bir şekilde tanıması sağlanır. Bu olay görüntü işleminin ta kendisidir [1].

Görüntüyü algılama ve yorumlama işlemi gerçekleşirken bir saniyede meydana gelen işlem sayısı şu an mevcut hiçbir bilgisayarın yapamayacağı kadar yüksektir. Bu kadar hızlı olmasının yanı sıra görmenin en şaşırtıcı ve mucizevi yanı, ağ tabakaya düşen ters görüntünün beynin optik merkezinde düzeltilmesidir [2].

Görüntünün meydana gelmesinde bir ışık kaynağına ihtiyaç duyulur. Bu ışık kaynağı sayesinde sürekli zamanlı bir görüntünün her hangi bir (x,y) noktasındaki (pikselindeki) parlaklık yoğunluğu (değeri) λ olarak ifade edilir. Parlaklık yoğunluğu ve ışık fonksiyonu gerçek ve pozitif niceliklerdir. Tüm görüntü işleme uygulamalarında azda olsa arka planda bir ışık kaynağı mevcuttur.

Görüntüyü seviye olarak üçe ayırabiliriz. Sadece siyah ve beyaz renkleri içeren ikili görüntü, gri seviyeli görüntü ve renkli görüntü. Her üç seviyedeki görüntünün sunumu da matris formundadır. Matrisin her hangi bir değeri görüntünün o koordinatlardaki noktanın parlaklık, ısı veya buna benzer farklı niteliklerdeki değeri olabilir [3].

Görüntü işleme teknikleri, görüntünün insan veya bilgisayar tarafından daha iyi anlaşılması, yorumlanması, kullanılması, depolanması ve transfer edilmesini sağlamak amacıyla, uzaysal alandaki bir görüntünün parlaklık, çözünürlük, zıtlık gibi nicel değerleri değiştirilmesini veya frekans alanındaki bir görüntünün filtrelenmesi, genlik veya faz değerlerinin değiştirilmesini sağlayarak yeni bir görüntünün oluşmasını sağlayan tekniklerdir. Görüntü bilgisi, Şekil 1.1'de gösterilen görüntü işleme teknikleri sayesinde istenilen forma dönüştürülür. Bu teknikler görüntü dönüştürme, onarma, iyileştirme, bölümleme, sıkıştırma, sunma ve algılama olarak verilebilir.

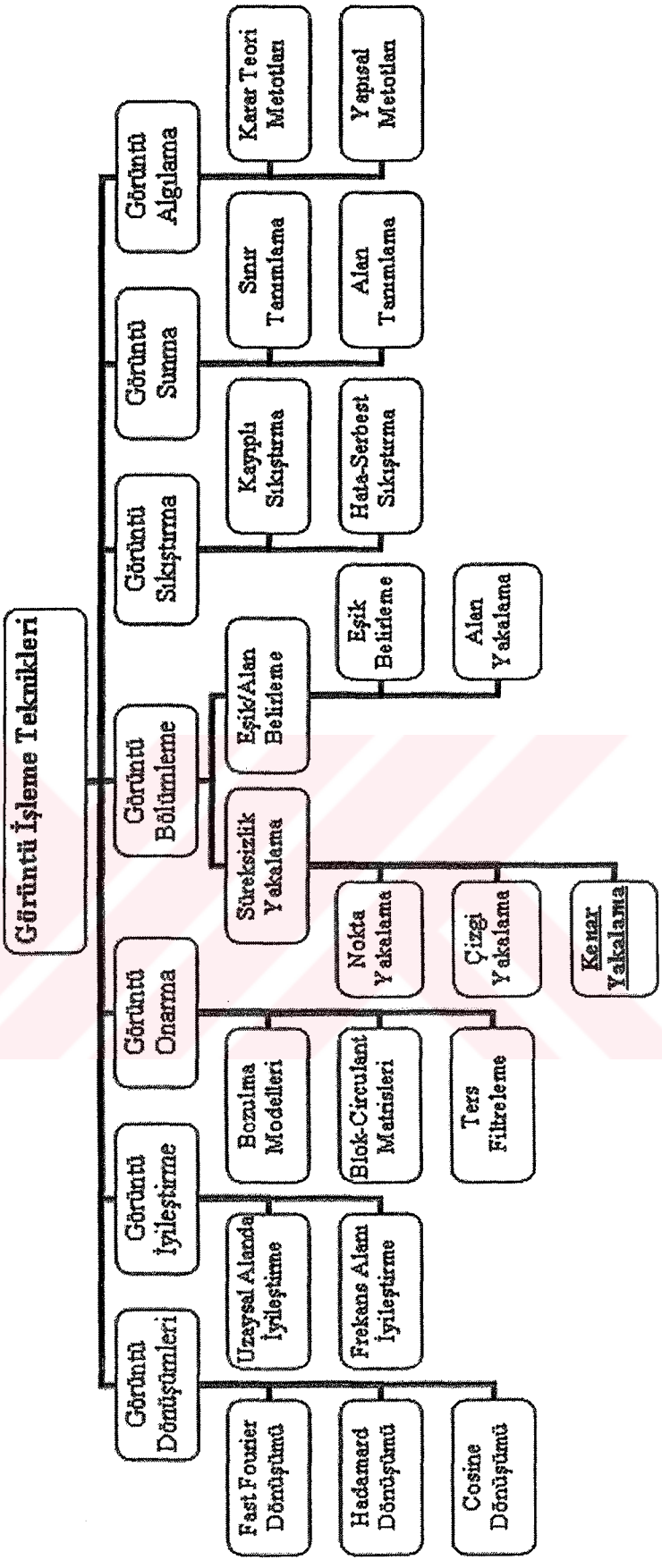
Tüm görüntü işleme teknikleri uzaysal ve frekans alanı olmak üzere iki farklı ortamda çalışır. Görüntü dönüştürme tekniği kullanılarak uzaysal alanda bulunan bir görüntü frekans alanına aktarılabilir veya işlemin tersi yapılabilir. Görüntü iyileştirme tekniği, kaynak görüntüyü daha anlaşılır ve yorumlanabilir bir biçime dönüştürmek için kullanılır. Eğer kaynak görüntü uzaysal alanda ise histogram eşitleme, görüntü çıkarma gibi operasyonlar kullanılır. Eğer kaynak görüntü frekans alanında ise düşük frekans geçiren, yüksek frekans geçiren, homomorphik filitreleme gibi operasyonlar kullanılır. Görüntü onarma teknikleri, bozulmuş görüntüleri düzeltme amacına yöneliktir.

Genellikle kaynak görüntü frekans alanında işlenir. Görüntü sıkıştırma tekniği, yüksek hacimli bir görüntüyü daha düşük bir hacme indirgeyerek, görüntünün depolanması işleminde hafızanın optimal kullanılmasını, görüntünün transfer edilmesi işleminde ise aktarım için harcanan süresinin daha az olmasını sağlar. Görüntü bölümleme tekniği, kaynak görüntü içerisindeki süreksizlik (nokta, çizgi ve kenar) ve benzerlik (alan) özelliklerinin tespit edilmesini amaçlar. Görüntü bölümleme tekniği ile parçalara ayrılmış bir görüntüyü tekrar sayısal ortamda kullanabilmek için uygun bir biçimde birleştirmek gerekir. Bu işlem için uygun bir görüntü sunma tekniği kullanılabilir. Görüntü işleme hiyerarşisi içerisindeki en önemli tekniklerden biri olan görüntü algılama ve yorumlama tekniği kullanılarak, görüntü içerisinde bölümlenmiş her bir nesnenin önceden veri tabanına kaydedilmiş nesne boyutlarına karşılık gelip gelmediği kontrol edilir. Böylece aranan nesnenin görüntü içerisinde olup olmadığına karar verilebilir [4, 5, 6, 7].

Kenar yakalama (edge detection) metodu, görüntü işleme tekniklerinden görüntü bölümleme konusunun bir alt başlığı olarak incelenir. Görüntünün bütününü oluşturan parçaların bir birinden ayrılması işlemine görüntü bölümleme denir. Görüntü bölümleme işlemi, görüntüden tüm nesneler ayrıldığı zaman son bulur. Parçalanmış görüntü, diğer görüntü işleme teknikleri tarafından işlenebilir. Görüntü analizinin başarılı olup olmaması, görüntü bölümlemenin başarılı olup olmamasına bağlıdır. Başka bir deyişle görüntüdeki aynı karakteristiğe sahip alanların bir birinden ayrılması işlemine bağlıdır. Bu işlemi gerçekleştirebilmek için, görüntü içerisindeki nesnelerin zemin ile kesiştikleri alanda nokta, çizgi ve kenar özelliklerinin yakalanması gerekir [8].

Görüntü içerisindeki nesnelerin algılanması ve geometrik özelliklerini tanıma süreci, insan görme sisteminde, nesnelerin dış çizgilerine göz gezdirilmesi ile başlar. Bu yaklaşımı yapay görme sistemlerine uyarladığımızda, şayet nesnenin sınırları başarılı bir şekilde izlenirse nesne tanımada çok daha iyi sonuçlar alınabilir. Böylece tanıma olayında nesneye ait kenarların yakalanması işlemi önemli bir rol oynar [9].

Kenar yakalama algoritmaları (KYA) üç farklı yapıya sahip görüntü üzerinde çalıştırılabilir. Bunlardan birincisi, iki seviyeli (siyah-beyaz) görüntüdür. Bu görüntüde nesneye ait bir nokta "1", arka plana ait bir nokta ise "0" olarak kodlanır. İkincisi, gri seviyeli görüntüdür. Bu görüntüdeki tüm noktalar 0-255 arasında gri tonajlı bir renk değerine sahiptir. Üçüncüsü ise renkli görüntülerdir. Bu görüntüdeki her nokta kırmızı,



Şekil 1.1 Görüntü işleme teknikleri

yeşil ve mavi olmak üzere 3 farklı renge ait 0-255 arasındaki değerlerden oluşan bir diziye sahiptir.

KYA'da iki önemli özelliğin olması gerekmektedir. Birincisi, nesnenin kenar çizgisinin tek nokta kalınlığında elde edilebilmesidir. Böylece nesneyi işlemek için daha az bilgi kullanıldığından, nesne tanımada oldukça önemli olan hız problemine zamandan tasarruf sağlanarak çözüm getirilebilir. Aynı zamanda nesne tanımak için oluşturulacak veri tabanları daha kolay oluşturulabilir. İkincisi, kenar yakalama işleminin sonucu, kenar yönlerinden bağımsız olmalıdır. Yani yakalanan kenar bilgisi, kenarı yakalanan nesne yön değiştirdiğinde de değişime uğramamalıdır [10].

Tezin Amacı:

Bu tezin amacı, iki seviyeli ve gri seviyeli görüntülerde önceden gerçekleştirilmiş geleneksel kenar yakalama yöntemlerini incelemek, uygulamalarını yapmak ve buna bağlı olarak daha hızlı, etkili ve performanslı kenar yakalama yöntemlerini araştırmak ve uygulamalarını gerçekleştirmektir. Sistemlerin zorluk derecelerinin her geçen gün biraz daha artması, daha fazla hacimde bilgiyi daha verimli bir şekilde işleme gerekliliği, daha hızlı ve daha etkili sistemlerin gerçekleştirilmesi zorunluluğunu ortaya çıkarmıştır. Bu nedenle, iki seviyeli görüntülerde, nesneleri tanıma işlemlerinde hayati önem taşıyan hız faktörünün, geliştirilen KYA kullanılarak iyileştirilmesi amaçlanmıştır. Böylece, görüntü analizinin ilk adımı olan kenar yakalama işlemi daha hızlı bir zamanda gerçekleştirilir. Bu sayede nesneleri tanımak için oluşturulan veri tabanları hafızayı daha az işgal eder ve diğer görüntü işleme tekniklerine daha fazla zaman ayrılır.

Gri seviyeli görüntülerde ise, nesnelere ait önemli kenar noktalarının yakalaması işleminde, doğru kenar yeri bulma, köşeleri tam yakalama, bulanık kenar yakalamama ve tek nokta kalınlığında kenar yakalama gibi, uzaysal alanda çalışan geleneksel KYA'nın ulaşamadığı bu hedeflere ulaşabilmek için Yapay Sinir Ağları (YSA) kullanılarak geliştirilen uygun bir yaklaşım sunulur. Lineer olmayan problemlerin çözümünde YSA'nın adaptasyon kabiliyetinin ispatlanmış oluşu ve YSA'nın görüntü iyileştirme, düzenleme, şifreleme ve bölümleme gibi alanlarda yüksek başarı

performansı ile kullanılması, bu hedeflere ulaşabilmek için YSA'yı tercih etmemizin temel nedeni olmuştur [11, 12, 13, 14] .

Yukarıda bahsedilen amaca yönelik olarak, Bölüm 2'de görüntü işleme tekniklerinden, görüntü bölümleme hakkında teorik bilgi verilmiştir. Bunun için, görüntüden nokta yakalama, çizgi yakalama ve kenar yakalama kavramlarından bahsedilmiştir.

Bölüm 3'de, iki seviyeli görüntülerde kenar yakalama problemi ele alınmıştır. Bunun için, geleneksel algoritmalar içerisinde en hızlı algoritmalardan biri olan ve 4-komşuluklu KYA [15] olarak adlandırılan algoritma incelenerek, bu algoritmanın avantajlarından ve dezavantajlarından bahsedilmiştir. Daha sonra, 4-komşuluklu KYA'sının dezavantajlarını ortadan kaldıran veya azaltan yeni bir KYA sunulmuştur. Geliştirilen KYA ile 4-komşuluklu KYA'nın karşılaştırılabilmesi ve sayısal sonuçlarının alınabilmesi amacıyla MATLAB 6.5 program editörü kullanılarak 'Binary_Edge' adlı bir paket program geliştirilmiştir. Bu program sayesinde, bahsedilen iki algoritmanın üstün ve zayıf olan yönleri hakkında gerekli sayısal sonuçlar alınmıştır. Elde edilen sonuçlar bölüm sonunda tablo halinde gösterilmiştir.

Bölüm 4'de, gri seviyeli görüntülerin kenarlarını yakalamak için geliştirilmiş geleneksel algoritmalar hakkında teorik bilgi yer almaktadır. Bu bölümde Prewitt, Sobel, Laplacian ve Canny [16-19] gibi geleneksel KYA'ları içerisinde en sık kullanılan KYA'dan bahsedilmiştir.

Bölüm 5'de, YSA'nın gelişmiş bir modeli olan Model Tabanlı Yapay Sinir Ağı (MYSA, Model-Based Neural Network) [20, 21] metodu kullanılarak yukarıda bahsedilen dezavantajları ortadan kaldıran veya azaltan makul bir KYA teorisi sunulmuştur. Ayrıca bu bölümde, MYSA'nın iyi öğrenme, adaptif filtreleme ve sınıflandırma özellikleri kullanılarak geliştirilen yeni bir kenar yakalama metodu ortaya konmuştur.

Bölüm 6'da, Bölüm 5'de teorisinden bahsedilen MYSA'nın MATLAB 6.5 program editörü kullanılarak yazılmış olan "Edge_Neuro" adlı programdan bahsedilmiştir. Bu program sayesinde yapay ve doğal görüntülerin MYSA algoritması kullanılarak kenar sonuçları elde edilmiştir. Bu kenar sonuçları geleneksel kenar yakalayıcılar içerisinde en iyilerden biri olarak bilinen Canny kenar yakalama

operatöründen elde edilen kenar sonuçları ile karşılaştırılmıştır. Algoritmaların üstün ve zayıf olan yönlerinden bahsedilip sonuçlar üzerinde yorumlar yapılmıştır.

Sonuç bölümünde ise bu tezde yapılan işlemlerin ışığı altında, kenar yakalama probleminin geleceği tartışılmıştır. İleride yapılabilecek benzeri çalışmalar ve uygulama alanları için öneriler sunulmuştur.

2. GÖRÜNTÜ BÖLÜMLEME

Görüntü analizindeki ilk ve en zor adım görüntü bölümlemedir. Analizin başarılı olup olmaması bölümlemenin başarılı olup olmamasına bağlıdır. Görüntünün, bütününü oluşturan parçalara veya nesnelere ayrılması işlemine görüntü bölümleme denir. Başka bir deyişle görüntüdeki aynı karakteristiğe sahip alanların çıkartılması, arka plandan ayrılması, belirgin bir hale getirilmesi işlemine görüntü bölümleme işlemi denir. Görüntü bölümleme işlemine, bir cadde görüntüsünden arabaların çekip çıkartılması, pilotun havadayken yerdeki nesneleri ayırt edebilmesi veya iç içe nesnelerin ayrılması örnek olarak verilebilir. Cadde görüntüsü için düşünülürse, yapılan ilk işlem bu görüntünün bölümlenmesidir. Daha sonra cadde görüntüsünün alt bölmelerinin potansiyel araba boyutlarına karşılık gelip gelmediği kontrol edilir. Belirtilen büyüklük bulunamaz ise cadde üzerinde araba olmadığı anlaşılır. Görüntü bölümleme işlemi görüntüden tüm nesneler ayrıldığı zaman son bulur. Bölmeler herhangi bir nesne olabileceği gibi farklı bileşenlerde olabilir.

Görüntü bölümleme algoritmaları, gri seviyeli görüntüler içerisindeki süreksizlik ve benzerlik özelliklerinin bulunması temeline dayanan iki farklı grupta geliştirilmektedir. Süreksizlik yaklaşımı, gri seviyelerinde ani değişimleri esas alarak görüntünün bölümlenmesini sağlar. Bu yaklaşım sayesinde görüntüdeki kenarlar, çizgiler ve noktalar tespit edilir. Benzerlik yaklaşımı ise, eşik (treshold), alan büyümesi, alanların ayrılması ve bileştirilmesi temeline dayanır. Statik ve dinamik görüntüler için noktalar arasındaki gri seviye değerlerinin süreksizlik ve benzerlik özellikleri beraberce kullanılabilir [22].

Bu tezde, görüntü bölümleme için süreksizlik özelliğini kullanan algoritmalar üzerinde çalışılmıştır.

2.1 Süreksizliğin Tanımlanması

Görüntüyü oluşturan noktaların parlaklık değerlerindeki değişimlerin belirtilen eşik değerinden daha büyük olması durumunda görüntünün o bölgesinde süreksizlik

meydana gelir. Bu bölümde sayısal görüntülerde süreksizliğin üç temel tipinin yakalanması ile ilgili farklı teknik ele alınmıştır. Bu üç temel tip, nokta yakalama, çizgi yakalama ve kenar yakalamadır. Pratikteki uygulamalarda görüntü içerisindeki süreksizliğin yakalanması için kullanılan en genel yöntem maskeleme tekniğidir. Bu tekniğin çalışma prensibi, Şekil 2.1'de gösterilen 3x3 boyutundaki maskenin, görüntüdeki alanlara uygulanarak elde edilen sonuçların eski değerler ile değiştirilmesi olarak söylenebilir [23]. Görüntüdeki her hangi bir noktanın maskelenmesinden elde edilen sonuç o anki değeri ile değiştirilir ve bu işlem görüntünün tüm nokta değerlerine uygulanır. Böylece maskelenmiş görüntü elde edilir. Buna göre her hangi bir noktanın maskelenmiş değeri denklem (2.1) ile hesaplanabilir.

$$R = w_1 z_1 + w_2 z_2 + w_3 z_3 + \ldots\ldots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i \tag{2.1}$$

Denklem (2.1)'deki z değeri, görüntüdeki güncel çerçeve içerisindeki noktaların gri seviye değerlerini gösterir. W değeri ise görüntünün maskeleme katsayıdır. Maskelemenin sonucu, maskelenen alanın orta nokta değerinin değişmesini sağlayacaktır.

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

Şekil 2.1 Genel 3x3 boyutundaki maskeleme

2.2 Nokta Yakalama

Görüntüde içerisinde komşularından farklılık arz eden noktaların tespit edilmesi işlemine nokta yakalama denir. Bu işlem için Şekil 2.2'deki maske kullanılır. Güncel çerçevenin orta noktası için R değeri hesaplanırken 8-komşuluğundaki nokta değerleri kullanılır. Elde edilen değer, denklem (2.2)'de gösterildiği gibi, önceden belirtilen T

eşik değerinden büyükse, o zaman aranan nokta bulunmuş olur. T eşik değeri, negatif olmayan bir değerdir.

-1	-1	-1
-1	8	-1
-1	-1	-1

Şekil 2.2 Noktayı yakalama için kullanılan maske

$$|R| > T \tag{2.2}$$

2.3 Çizgi Yakalama

Çizgi yakalama işlemi, nokta yakalama işleminden daha komplekstir. Görüntü içerisinde tek nokta kalınlığında bir çizgi şeklindeki süreksizliği yakalamak için kullanılır. Bu işlem için Şekil 2.3'deki maskeler kullanılır. Şekil 2.3'teki maskelere bakıldığında, görüntü içerisinde yatay eksen ile arasında sırasıyla $0^0, +45^0, 90^0$ ve -45^0'lik açılar olan çizgiler yakalanır.

-1	-1	-1
2	2	2
-1	-1	-1

0

-1	-1	2
-1	2	-1
2	-1	-1

+45

-1	2	-1
-1	2	-1
-1	2	-1

90

2	-1	-1
-1	2	-1
-1	-1	2

-45

Şekil 2.3 Çizgi yakalamak için kullanılan maskeler

2.4 Kenar Yakalama

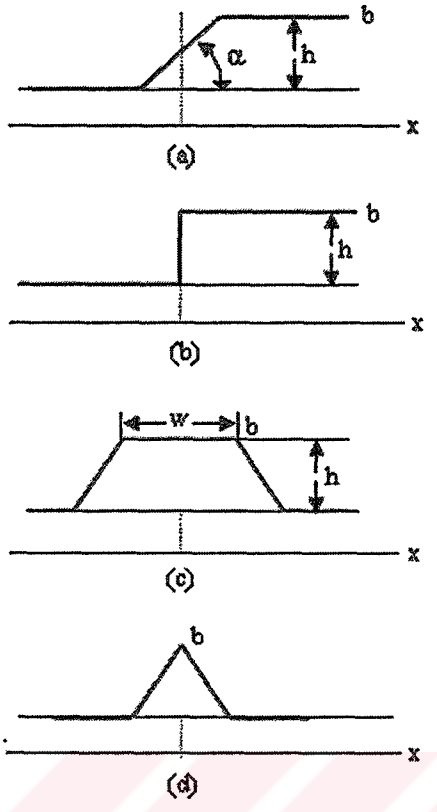
Kenar yakalama işlemi, görüntü içerisinde, komşuları ile farklı yoğunluklara sahip noktaların tespit edilmesidir. Kenar yakalamadaki amaç, nesnenin gereksiz ve tanıma işlemlerinde zaman kaybettiren bilgisini eleyerek yeterli düzeye indirgemektir.

Kenar yakalama işlemi genellikle iki aşamadan oluşur. Birinci aşama, güncel noktanın yoğunluğu komşu noktalar ile karşılaştırılarak yoğunluk değişimleri tespit edilir. Başka bir deyişle, noktaların parlaklık değerleri arasındaki uygun değişim yerleri tespit edilir. İkinci aşama ise, elde edilen değişimlerin önceden belirlenmiş bir eşik değeri ile karşılaştırılarak "önemli" bir değişim olup olmadığı tespit edilir. Önemli değişimin olduğu düşünülen noktalara kenar noktaları denir [24].

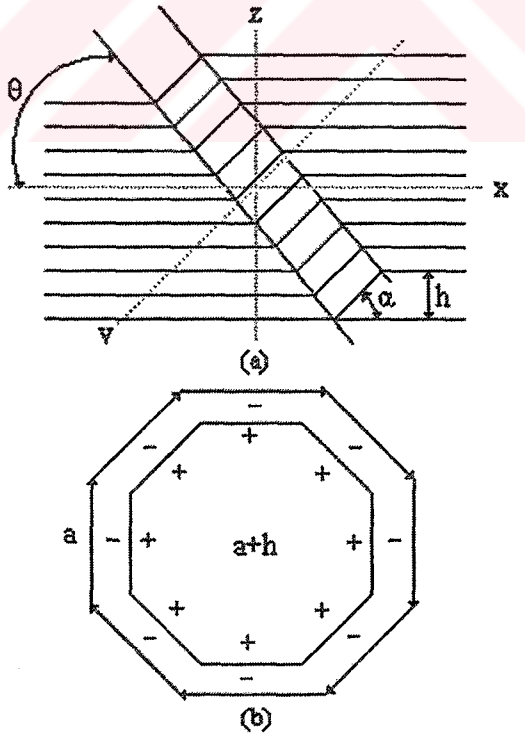
2.5 Kenar, Çizgi ve Nokta Modelleri

Şekil 2.4 (a)'da, sürekli bir alan görünümü gösterilir ve tek boyutlu bir kenarın görüntü genliği seviyesinin, düşük seviyeden yüksek seviyeye veya tersi yönde bir rampa şeklinde değişmesi modellenmiştir. Bu değişimin yüksekliği, eğim açısı ve eğimin yatay koordinatının orta noktası kenarı karakterize etmemizi sağlar. Eğer değişim veya süreksizliğin büyüklüğü, önceden belirlenen değişim sabitinden daha büyükse o zaman bu noktadaki süreksizlik kenar olarak ifade edilebilir. İdeal kenar yakalayıcılardan beklenen sonuç, değişimin orta nokrasını tek bir nokta ile ifade edebilmektir. Şekil 2.4'de, kenar alanlarındaki farklı değişim görüntüleri yer almaktadır. Eğer Şekil 2.4(a)'deki değişim açısı 90^0 ise keskin bir değişimin olduğu anlaşılır. Bu keskin değişeme "*adım kenar*" denir. Şekil 2.4 (b)'de adım kenarın değişimi gösterilir. Sayısal görüntü sistemlerinde genellikle adım kenarlar yalnız test örnekleri ve iki seviyeli grafik verileri gibi yapay olarak üretilen görüntüler için kullanılır. Sayısal görüntülerde, optik görüntülerin sayısallaştırılması işlemleri dışında farklı işlemlerde adım kenarlara pek rastlanılmaz. Çünkü gerçek hayatta kullanılan her hangi bir görüntüyü sayısallaştırmak istersek, öncelikle bir filtreleme fonksiyonundan geçirmeliyiz. Böylece sert geçiş diye adlandırdığımız parlaklık değerindeki ani değişimlere karşılık gelen adım kenarlar ortadan kalkmış olur.

Şekil 2.4(c)'de bir çizginin tek boyutlu görüntüsü gösterilir. Çizgi genişliğinin sınırı olan w (süreksizliğin genişliği) değeri sıfıra yaklaştığı zaman "*çatı kenar*" olarak adlandırılan kenar ortaya çıkar. Sürekli zamanda kenar ve çizginin, iki boyutlu modellerinde, değişim büyüklüğünün, küçük bir değer olduğu varsayılır. Şekil 2.5 (a)'da iki boyutlu kenar modeli verilmiştir. Şekil 2.5 (b)'de ise arka plandan farklı parlaklık değerine sahip olan kenarın yön tanımlanması verilmiştir.



Şekil 2.4 Tek boyutlu, sürekli düzlemde kenar ve çizgi modelleri (a)Rampa kenar (b)Adım kenar (c)Çizgi (d)Çatı kenar



Şekil 2.5 İki boyutlu, sürekli düzlemde kenar modeli

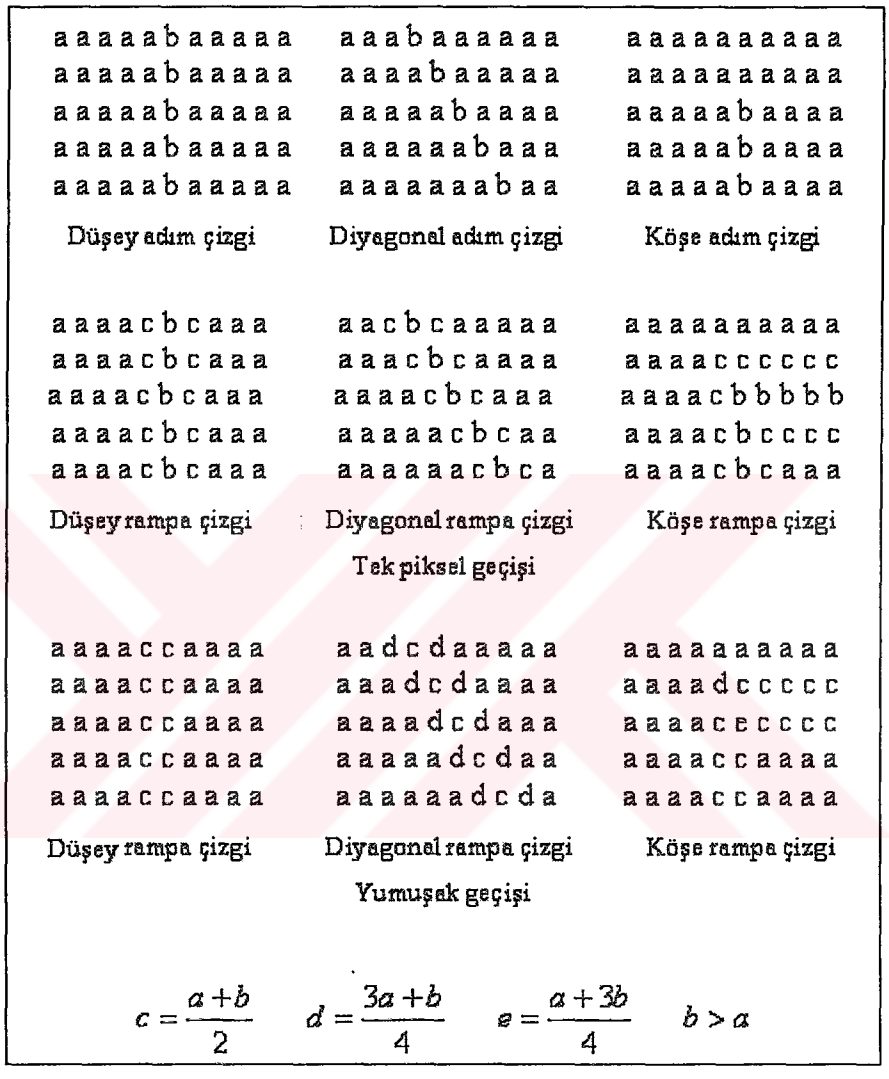
Şekil 2.6, ayrık zamanda adım ve birim rampa kenar modellerini içerir. Şekildeki düşey rampa kenar modeli, tek nokta geçişini içerir. Bu geçiş, çerçevenin orta noktası ile 8-komşuluğundaki noktalar arasında olur. Bu kenar modeli 2x2 boyutundaki bir maske kullanılarak elde edilebilir. Şekil 2.6, bir diyagonal rampa kenarının iki farklı versiyonunu da içerir. Tek noktalık geçiş modeli yüksek ve düşük genliklere sahip alanlar içerisinde sadece bir tane orta-değerli geçiş noktasını içerir. Yumuşak geçiş modellerini yakalayabilmek için, 2x2 boyutuna sahip maskeler kullanılabilir. Bu hareketli maskelerde pencerelerin ortalamaları üretilir. Şekil 2.6, aynı zamanda ayrık adım ve rampa köşe kenarlarının modellerini de gösterir. Ayrık adım kenarlar için kenar yeri genellikle kenar geçişinin daha yüksek genlikli kısmı olarak belirlenir. Tek noktalık geçiş modellerinde ve yumuşak geçişli dikey ve köşe kenar modellerinde uygun kenar yeri geçiş noktasıdır. Yumuşak geçişli diyagonal rampa kenar modellerinin geçiş bölgelerinde komşu noktaları vardır. Kenar yeri genellikle çiftlerden daha yüksek genlikli nokta olarak işaretlenir.

a a a a a b b b b b	a a a b b b b b b b	a a a a a a a a a a
a a a a a b b b b b	a a a a b b b b b b	a a a a a a a a a a
a a a a a b b b b b	a a a a a b b b b b	a a a a a b b b b b
a a a a a b b b b b	a a a a a a b b b b	a a a a a b b b b b
a a a a a b b b b b	a a a a a a a b b b	a a a a a b b b b b
Düşey adım kenar	Diyagonal adım kenar	Köşe adım kenar
a a a a c b b b b b	a a a c b b b b b b	a a a a a a a a a a
a a a a c b b b b b	a a a a c b b b b b	a a a a c c c c c c
a a a a c b b b b b	a a a a a c b b b b	a a a a c b b b b b
a a a a c b b b b b	a a a a a a c b b b	a a a a c b b b b b
a a a a c b b b b b	a a a a a a a c b b	a a a a c b b b b b
Düşey rampa kenar	Diyagonal rampa kenar Tek piksel geçişi	Köşe rampa kenar
a a a a c b b b b b	a a d e b b b b b b	a a a a a a a a a a
a a a a c b b b b b	a a a d e b b b b b	a a a a d c c c c c
a a a a c b b b b b	a a a a d e b b b b	a a a a c b b b b b
a a a a c b b b b b	a a a a a d e b b b	a a a a c b b b b b
a a a a c b b b b b	a a a a a a d e b b	a a a a c b b b b b
Düşey rampa kenar	Diyagonal rampa kenar Yumuşak geçişi	Köşe rampa kenar

$$c = \frac{a+b}{2} \qquad d = \frac{3a+b}{4} \qquad e = \frac{a+3b}{4} \qquad b > a$$

Şekil 2.6 İki boyutlu, ayrık düzlemde kenar modelleri

Şekil 2.7'de ise ayrık iki boyutlu tek nokta kalınlığında çizgi modelleri bulunmaktadır. Tek noktalı geçiş modellerinin düşük değerli arka planları ve çizgilerin yüksek değerleri arasında orta değerli geçiş noktaları vardır. Yumuşak geçiş modelleri adım çizgi modellerindeki 2x2'lik hareketli pencereler kullanılarak elde edilir.

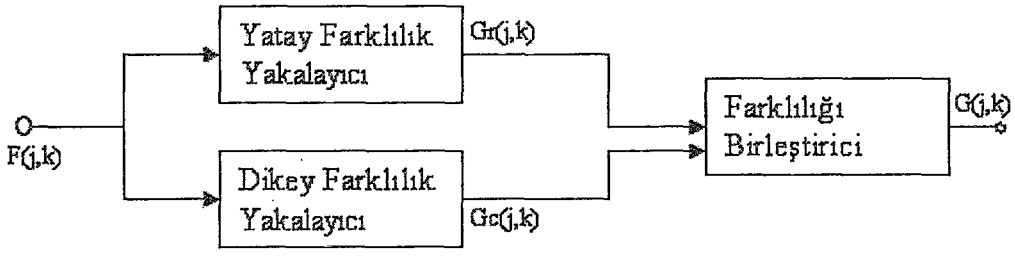


Şekil 2.7 İki boyutlu, ayrık düzlemde çizgi modeli

Bir nokta yalnız iki boyutta tanımlanabilir. Düşük genlikli arka plana karşı yüksek genlikli bir değer içerir. Şekil 2.8 ayrık zamanda tek nokta modelini gösterilmiştir. Buna göre arka planı temsil eden noktaların parlaklık değerlerinden farklı olan tek noktalık değişimin şekilsel gösterimi aşağıdaki gibidir.

Şekil 2.8 İki boyutlu, ayrık düzlemde tek noktalık geçiş modelleri

Görüntüdeki kenar, çizgi ve noktaları yakalamak için farklı yaklaşımlar söz konusudur. Farklılığı yakalama ve uygun modelleme tekniği bu yaklaşımlardan biridir. Şekil 2.9'da gösterilen bu teknik sayesinde, orijinal F(j,k) görüntüsü, uzaysal farklılaştırma teknikleri kullanılarak G(j,k) çıkışını üretir. Böylece görüntüdeki önemli farklılıkların yerleri ve büyüklükleri tespit edilir.



Şekil 2.9 Yatay ve dikey farklılık yakalayıcı

3. İKİ SEVİYELİ GÖRÜNTÜLERDE KENAR YAKALAMA

3.1 Giriş

Görüntü bilgisinin en düşük seviyeli sunumuma iki seviyeli görüntü denir. Nesneye ait noktalar "1", zemine ait noktalar ise "0" olmak üzere iki gri seviye değeri ile ifade edilir. Medikal uygulamalarda, savunma sistemlerinde, gerçek zamanlı sistemlerde ve değişik endüstriyel alanlarda çok yaygın bir şekilde kullanılır. İki seviyeli görüntü işlemenin avantajları ve dezavantajları aşağıdaki gibidir [25].

Avantajları:

- Kolay elde edilme: Basit sayısal kameralar, düşük maliyetli tarayıcılar ile rahatlıkla elde edilebilir. Ayrıca, gri seviyeli görüntüler için bir eşik değeri olarak kullanılabilir.
- Az yer kaplama: Bir noktanın kapladığı alan sadece bir bit'tir. Sıkıştırma algoritmaları kullanılarak kapladığı alan çok düşük bir seviyeye düşürülebilir.
- Basit ve hızlı işlenme: Olası kombinasyon seçenekleri çok az olduğu için gri seviyeli ve renkli görüntülere uygulanan algoritmalardan çok daha basit ve hızlıdır.

Dezavantajları:

- Sınırlı uygulama: Doğal ortamdan elde edilen görüntüleri yeterince ifade edemediği için yapay ortamlarda oluşturulan görüntüler üzerinde kullanılır.
- 3D'ye genişleyememe: Nadirde olsa nesnelerin 3D'li doğal halleri gölgeleme tekniği kullanılarak gösterilmeye çalışılır.
- Kaliteli görüntülerde gölgelemeye ihtiyaç duyma: Çevresel etkiler olmaksızın iki seviyeli kaliteli bir görüntü elde etmek istersek, gölgeleme tekniği kullanılmak zorunda kalınır.

İki seviyeli görüntülerde, nesne kenarlarını yakalama algoritmalarında arzulanan sonuç, hiçbir bilgi kaybına olanak tanımadan, hafızada en az yer tutan, tek nokta genişliğindeki en ince kenar izinin bulunmasıdır. Böylece nesne tanıma işlemi daha

hızlı yapılabilir ve aynı zamanda nesne tanımak için oluşturulacak veri tabanları daha kolay oluşturulabilir [26].

Otomasyonda nesne tanınması amacıyla geliştirilen kenar yakalama algoritmalarının, iki boyutlu görme işlemini hızlı bir şekilde gerçekleştirmeye çalışırken, yüksek bir başarı yüzdesine ulaşmaları beklenir. Nesne tanımada sağlanabilecek olan en üst düzey başarı; boyuttan, parlaklıktan, yer ve duruş açısındaki değişimlerden etkilenmemedir. Bu başarıya ulaşmak için, görüntü işlenirken bilgi kaybı olmamalıdır. Fakat tüm görüntünün işlenmesi çok fazla zaman almaktadır. Bu sebeple, tanınması istenen nesnelerin kenarları yakalanarak, nesne tanıma işleminin hızlı ve yüksek performanslı olması sağlanır [27].

3.2. İki Seviyeli Görüntülerde 4-komşuluklu KYA

İki seviyeli görüntülerde kenar yakalamak için farklı algoritmalar geliştirilmiştir [8,9]. Tez çalışmamızda, bu yöntemler içerisinde en hızlılardan biri olarak bilinen 4-komşuluklu KYA kullanılmıştır. Bu algoritma kullanılarak, nesnenin sınırlarını temsil eden çizgiler tek hat halinde elde edilebilir. Sayısal ve iki renk (zemin beyaz, nesne siyah) halindeki nesne görüntüsüne aşağıda belirtilen KYA uygulanarak nesne sınırları istenen biçimde elde edilir. 4-komşuluklu KYA, nesneye ait olan her noktayı 4 komşuluğundaki nokta değerleri ile birlikte yorumlayarak, bu noktanın nesnenin kenarına ait olup olmadığını tespit eder. Şekil 3.1 (a)'da nesnenin her hangi bir noktası (N5) ve 8 komşuluğundaki noktalar (N1, N2, N3, N4, N6, N7, N8, N9) gösterilmiştir.

Sayısal görüntüde, şayet nokta nesneye ait ise değeri "1", değil ise "0" olarak alınmak üzere Şekil 3.1 (b)'deki şablon, her bir nesne noktası ve komşularına uygulanarak, o noktanın nesnenin kenarına ait olup olmadığı belirlenir. Bu kenar yakalama şablonu klasik var/yok mantığı ile denklem (3.1)'deki gibi ifade edilebilir.

N1	N2	N3
N4	N5	N6
N7	N8	N9

(a)

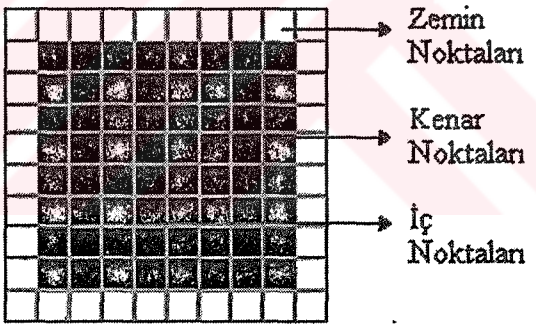
0	1	0
1	N5	1
0	1	0

(b)

Şekil 3.1 (a) N5 noktasının komşuları (b) 4-komşuluklu KYA şablonu

$$Ç = n_2 * n_4 * n_6 * n_8 \tag{3.1}$$

Denklem 3.1'deki ifadenin sonucu "0" çıkarsa, N5 noktasının nesnenin kenarına ait olduğu anlaşılır ve dolayısıyla N5 noktasının değeri "1" olarak kalır. Şayet ifadenin sonucu "1" çıkarsa N5 noktası nesne kenarına ait olmayıp, bu noktanın değeri "0" yapılır. Buna göre, 4-komşuluklu KYA, "Bir kenar noktası, nesneye ait olan ("1" değerini içeren) bir noktadır ve 4-komşuluğundaki noktalardan en az biri zemine aittir ("0" değerini içerir)" yargısına dayanır. 8-komşuluğun benimsenmesi durumunda, kenar ile ilişkilendirilmiş 4-komşuluk elde edilir. Diğer durumda, kenar sonucu 8-komşuluk ile ilişkilidir. Şekil 3.2 de bu kavram örneklenmiştir. İç noktalar nesneye aittir ve nesneye ait olan noktalar tarafından çevrelenir. 4-komşuluklu KYA, Şekil 3.3'te gösterilir. Buna göre, kaynak görüntüdeki her p(x,y) noktasını ve bu noktanın 4-komşuluğundaki üst p(x,y+1), alt p(x,y-1), sağ p(x+1,y) ve sol p(x-1,y) noktalarını giriş alarak iki seviyeli *p* kaynak görüntüsünden, yine iki seviyeli hedeflenen *h* kenar görüntüsü elde edilir.



Şekil 3.2 İki seviyeli kaynak görüntü

```
For her p(x,y) noktası
h(x,y) = 0
    if p(x,y) = 1
        if ( p(x,y+1) = 0) OR ( p(x,y-1 = 0)
            OR ( p(x+1,y) = 0) OR ( p(x,-1,y) = 0)
                h(x,y) = 1
```

Şekil 3.3 İki seviyeli görüntüde klasik kenar yakalama algoritması

3.3. İki Seviyeli Görüntüler İçin Geliştirilen KYA

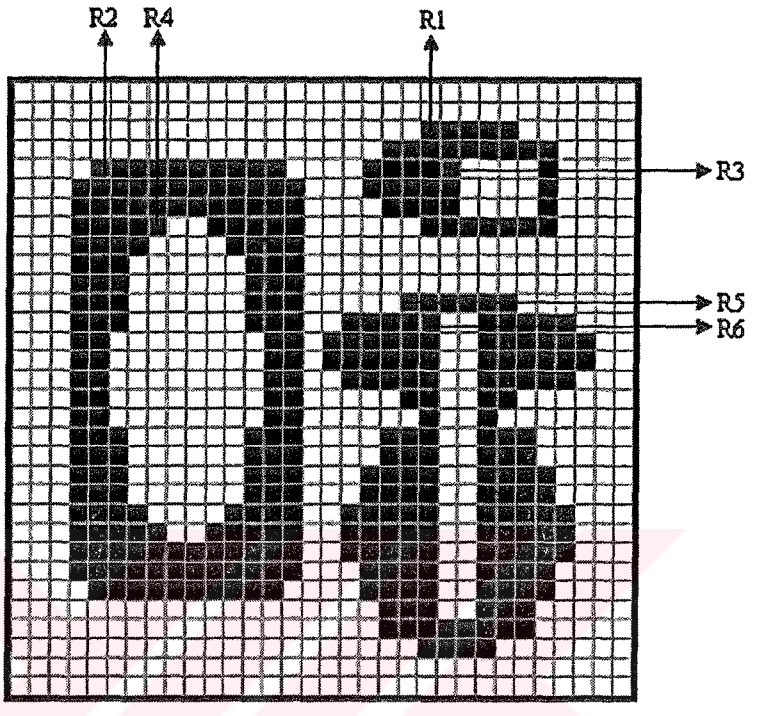
İki seviyeli görüntüler için geliştirilen KYA, görüntüdeki nesnelerin kenarlarını klasik metotlardan daha hızlı bir şekilde yakalamayı amaçlar. Bunun için, klasik 4-komşuluklu KYA'da olduğu gibi, görüntüdeki nesnelere ait tüm noktaların kenar noktası olup olmadığını kontrol etmek yerine, sadece nesnenin zemin ile komşu olan noktalarını kontrol eder. Nesnelerin sadece zemine komşu olan noktaları üzerinde yapılan bu gezinti, onların geometrik biçimi hakkında yeterli bilgiyi elde etmemiz için harcanan zamanın minimize edilmesini sağlar. Bu sayede, iki seviyeli görüntülerde nesne algılama ve nesnelere ait veritabanını oluşturma işlemleri hızlı bir şekilde gerçekleşir [10].

Geliştirilmiş KYA, bir nesnenin kenar noktalarını yakalayabilmek için ardışıl iki işlem gerçekleştirmektedir. Birinci işlem, nesnenin zemine komşu olan ilk noktasını (referans noktası) yakalamaktır. İkinci işlem ise, yakalanan referans noktası kullanılarak nesnenin aynı seviyedeki diğer kenar noktalarının yakalanmasını için kenar izleme algoritmasının çalıştırılmasıdır.

Her hangi bir nesnenin referans nokta sayısı (RNS) $1 \leq RNS \leq +\alpha$ olmak zorundadır. Şekil 3.4'te bir birinden bağımsız birden fazla nesne içeren iki seviyeli bir görüntü, geliştirilen KYA'na giriş olarak verildiğinde yakalanan referans noktaları gösterilir. Bu görüntülerde bulunan nesnelerin maksimum iki referans noktası vardır. Biri nesnenin dış yüzeyinin zemin ile komşu olduğu ilk nokta, diğeri ise nesnenin iç yüzeyinin zemin ile komşu olduğu ilk noktadır. R1, R2, …. , R6 olarak adlandırılan bu referans noktaları, nesnelerin zemin ile komşu olduğu ilk noktalardır. Dikkat edilirse, her bir kenar çizgisine bir referans noktası karşılık gelmektedir.

Nesnelerin sadece zemine komşu olan noktaları üzerinde gezinebilmeyi amaçlayan geliştirilmiş KYA, nesnelere ait referans noktalarını tespit edebilmek için Şekil 3.5'te gösterilen algoritmayı kullanır. Bu algoritmaya göre, ilk olarak beyaz zemin üzerinde siyah nesne aranır. Eğer önceden yakalanan bir kenar noktası değilse o zaman bulunan nokta o nesnenin dış referans noktasıdır ve kenar izleme algoritmasına gönderilir. Kenar izleme işlemi sona erdikten sonra, aranacak nesnenin rengi değiştirilir. Yani siyah zemin üzerinde beyaz nesne aranmaya başlanır. Eğer beyaz nesne bulunursa, bulunan noktaya komşu olan önceki siyah noktanın daha önce yakalanmış bir kenar

noktası olup olmadığı kontrol edilir. Eğer önceden yakalanan bir kenar noktası değilse o zaman bulunan nokta o nesnenin iç referans noktasıdır. Arama işlemi görüntü içerisindeki tüm nesnelerin kenarları yakalanıncaya kadar devam eder.



Şekil 3.4 İki seviyeli bir görüntüdeki referans noktaları

```
Renk = 1 (Siyah)
For her p(x,y) noktası
        if  p(x,y) = Renk
        Begin
                if Renk = 0   Başlangıç_vektörü = 1
                Else          Başlangıç_vektörü = 2
                if  h(x,y) = 0
                        kenar_izle(p,h,x,y,Başlangıç_vektörü)
                        Renk = tersi(Renk)
        End
```

Şekil 3.5 Referans noktalarının tespiti için kullanılan algoritma

Nesneye ait kenar noktalarının yakalanabilmesi için, nesneye ait referans noktalarının tespit edilip, kenar izleme algoritmasına gönderilmesi gerekir. Beyaz zemin üzerinde siyah nesne arama veya siyah zemin üzerinde beyaz nesne arama fikrine dayanan referans noktası yakalama işlemi, yakaladığı her referans noktasını Şekil 3.6'da gösterilen kenar izleme algoritmasına gönderir. Kenar izleme algoritması, Şekil 3.7'de gösterilen yön vektörlerini kullanarak nesnenin kenar noktaları üzerinde gezinebilmeyi sağlar. Kenar izleme algoritması, başlangıç vektörü olarak Şekil 3.7 (a)'da gösterilen ve başlangıç açısı -45^0 olan yön vektörünü kullanır. Bu yön vektörü kullanılarak yeni kenar noktası yakalanır. Yakalanan yeni kenar noktasının yönü ile kullanılan vektörün başlangıç açısı arasındaki fark açısı denklem (3.2)'de gösterildiği gibi hesaplanır. Denklem (3.2)'deki, YYA değişkeni kullanılan vektörde yakalanan yön açısıdır. KBA değişkeni ise kullanılan vektörün başlangıç açısıdır. Bu değişkenlerin farkı ile hesaplanan fark açısı, denklem (3.3)'de belirtilen şarttan geçirilerek kullanılacak yeni yön vektörü bulunur.

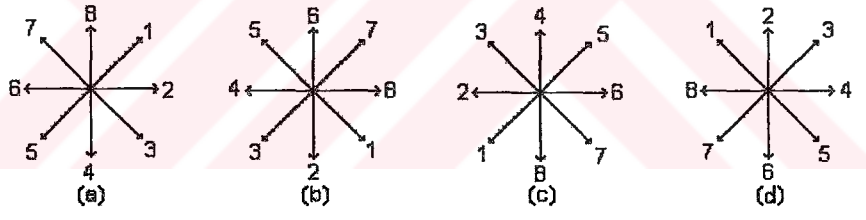
$$\phi = YYA - KBA \tag{3.2}$$

$$Yön\ Vektörü = \begin{cases} \phi = 0^0 & Eski\_Vektör - 1 \\ 0^0 < \phi < 90^0 & Eski\_Vektör \\ \phi > 90^0 & Eski\_Vektör + 1 \end{cases} \tag{3.3}$$

Buna göre, yakalanan kenar noktasının yönü 2 veya 3 yönünde ise bir sonraki kenar noktasını yakalama işlemi için kullanılan eski yön vektörü değiştirilmez. Fakat yakalanan kenar noktasının yönü 1 yönünde ise yeni yön vektörü, eski yön vektöründen bir önce gelen vektör olarak seçilir. Eski yön vektörünün ilk vektör olması durumunda, yeni yön vektörü son vektör olarak seçilir. Yakalanan kenar noktasının yönü 4, 5, 6, 7 veya 8 yönünde ise yeni yön vektörü, eski yön vektöründen bir sonra gelen vektör olarak seçilir. Eski yön vektörünün son vektör olması durumunda, yeni yön vektörü ilk vektör olarak seçilir. Yakalanan kenar noktası, başlangıç noktasına eşitse kenar izleme işlemi son bulur.

1. Yön vektörü = Eski_ vektör = Başlangıç_vektörü
   İlk_vektör = 1, Son_vektör = 4
2. Yön vektörü kullanılarak yeni kenar noktasının yeri ve yönü tespit edilir.
3. **if** Bulunan_Yön = 1 **then**
   Yön vektörü = Eski yön vektörü – 1
   if Yön vektörü < İlk_vektör **then**
   Yön vektörü = Son_vektör
   Else
   Yön vektörü = Eski yön vektörü + int ((Bulunan_Yön – 1) / 3)
   if Yön vektörü > Son_vektör **then**
   Yön vektörü = İlk_vektör
4. **if** Başlangıç noktasına gelindi **then** goto 5. adım.
   Else goto 2. adım.
5. Kenar izleme işlemi sona erdi.

Şekil 3.6 Kenar izleme algoritması



Şekil 3.7 Kenar izleme algoritmasının kullandığı yön vektörleri. Başlangıç açısı = (a) -45^0 (b) $+45^0$ (c) $+135^0$ (d) $+225^0$'lik yön vektörleri

3.4 Binary_Edge Programı ve Uygulama Sonuçlarının Karşılaştırılması

İki seviyeli görüntüler için geliştirilen KYA ile 4-komşuluklu KYA'nın performanslarını daha iyi analiz edilebilmek için MATLAB 6.5 Programlama Dili kullanılarak "Binary_Edge" adlı bir program yazılmıştır. Şekil 3.8'de Binary_Edge programının bir görüntüsü verilmiştir. Binary_Edge programı kullanılarak, farklı referans noktalarına ve farklı boyutlara sahip 12 değişik görüntü üzerinde iki algoritmayı da çalıştırabilir ve elde edilen kenar sonuçlarını görebiliriz. Herhangi bir

görüntü seçildiğinde, görüntünün ebatları ve sahip olduğu siyah nokta sayısı gösterilir. Ayrıca algoritmaların çalışması sonlandığında toplam harcanan zaman ve toplam karşılaştırma sayıları hesaplanarak ekranda gösterilir.

Geliştirilen KYA ile 4-komşuluklu KYA'nın performanslarını görebilmek için Şekil 3.9'daki yapay görüntüler ve Şekil 3.11'deki doğal görüntüler kullanılmıştır. Şekil 3.9'daki yapay görüntülerin her biri "Image Editor" adlı program aracılığıyla hazırlanmıştır. Farklı referans noktalarına sahip olan bu görüntüler farklı kenar özellikleri içerirler. Şekil 3.10'da, yapay nesnelere ait kenar görüntüleri, Şekil 3.12'de ise doğal görüntülerin kenar sonuçları gösterilmiştir. Şekil 3.11'deki ilk doğal görüntü, ulu önder Mustafa Kemal ATATÜRK'e aittir. İkincisi, dağınık bir şekilde duran pirinç tanelerini göstermektedir. Diğer iki görüntü ise gri seviye değerleri ve boyutları bir birinden farklı dairelerden oluşmuş görüntülerdir. Doğal görüntülerin orijinal hali gri seviye değerlerine sahiptir. Gri seviyeli bu görüntüler iki seviyeye dönüştürüldükten sonra algoritmalara giriş olarak verilmiştir.

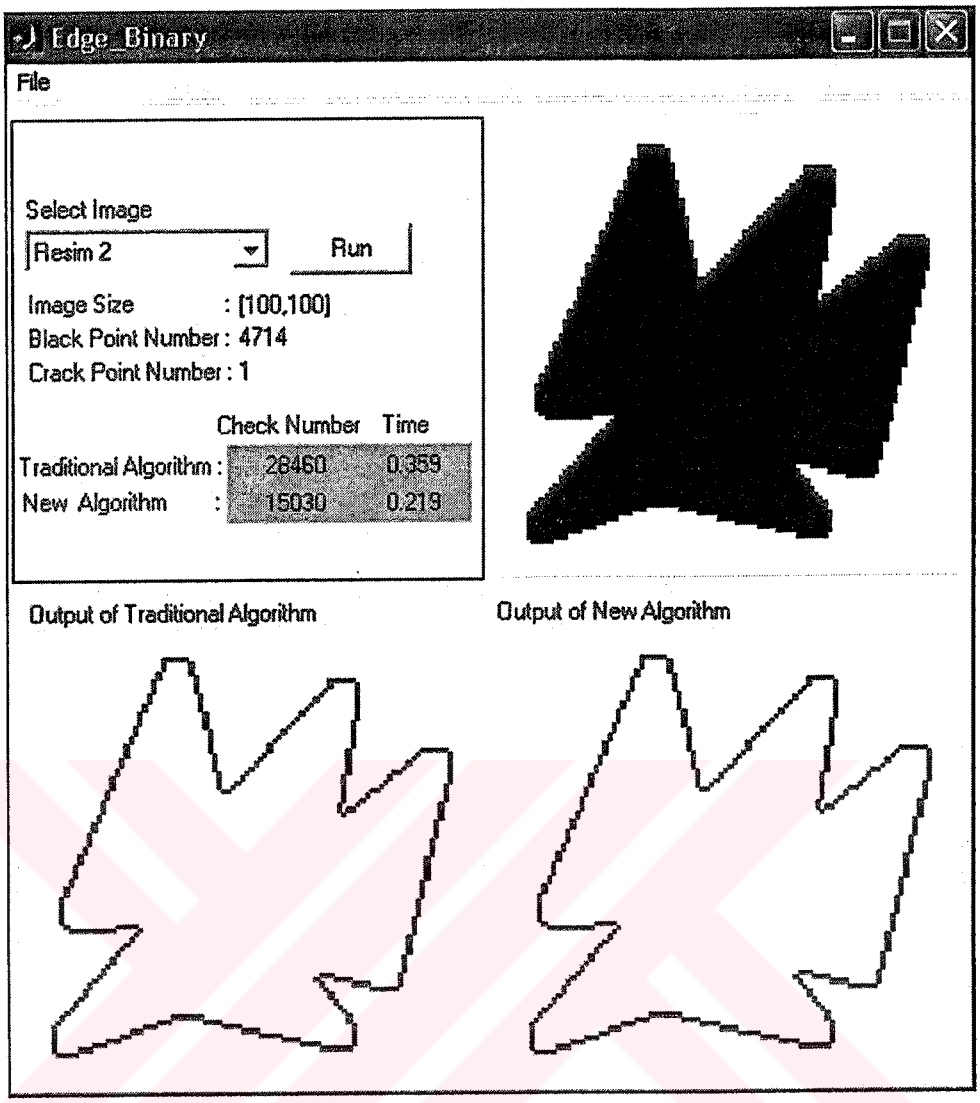
Karşılaştırılan her iki algoritmada tek nokta kalınlığında hatasız kenar sonucu üretmektedir. Ayrıca her iki algoritmada, görüntü içerisindeki nesnelerin yönlerinden etkilenmemektedir. Yani kenarı çıkarılan nesne yön değiştirdiğinde de kenar sonucu değişmemektedir. Bu nedenle, algoritmaların performanslarını görebilmek için toplam harcadıkları zaman ve toplam karşılaştırma sayısı olmak üzere iki kriter belirlenmiştir. Bu kriterlere algoritmaların verdiği sonuçlar Şekil 3.13'de gösterilmiştir.

Elde edilen sonuçlar gösteriyor ki, arzu edilen kenar sonucuna ulaşabilmek için, geliştirilen KYA, 4-komşuluklu KYA göre daha az karşılaştırma yapar ve daha kısa bir zaman kullanır. Görüntü içerisindeki bağımsız nesne sayısının (referans nokta sayısının) az olduğu ve siyah nokta sayısının fazla olduğu görüntülerde, geliştirilen algoritmanın performansı klasik algoritmanın performansından yaklaşık iki kat daha iyidir. Fakat görüntüdeki bağımsız nesne sayısının fazla olduğu ve bu nesnelerin boyutlarının küçük olduğu görüntülerde geliştirilen algoritmanın toplam karşılaştırma sayısı klasik algoritmanın toplam karşılaştırma sayısından daha iyi sonuç vermesine rağmen toplam harcanan zaman değeri daha kötü sonuç vermektedir. Bunun nedeni ise, görüntüdeki referans nokta sayısı arttıkça kenar izleme algoritmasının çağrılma sayısı da artar. Buda geliştirilen KYA'nın diğer matematiksel işlemlere daha fazla zaman harcaması demektir.

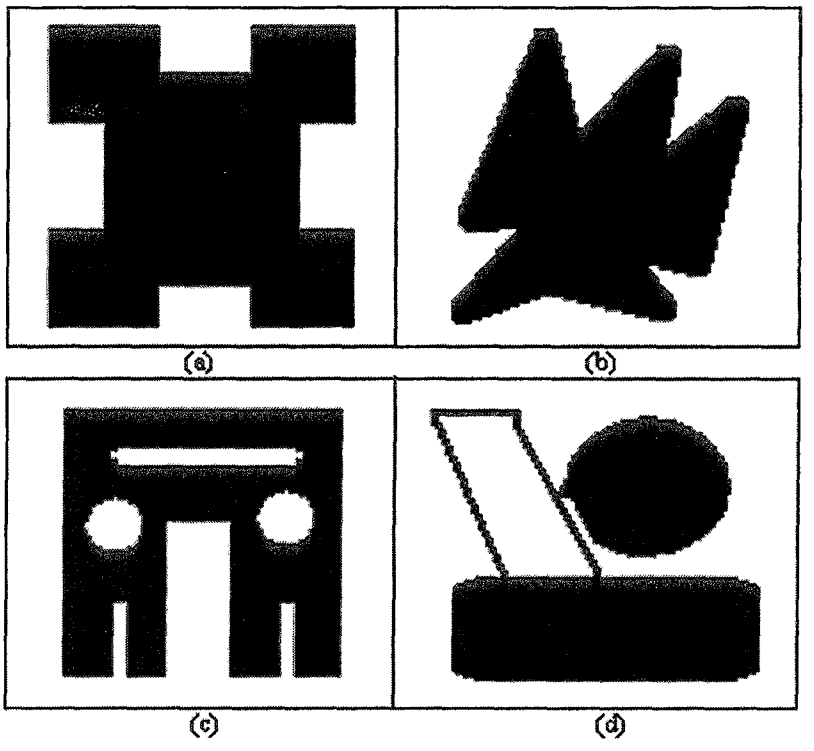
Geliştirilen KYA'nın karşılaştırma sayısı her durumda 4-komşuluklu KYA'dan daha azdır. Bunun nedeni geliştirilen KYA, görüntü içerisindeki nesnelerin dış hatlarına tutunarak geçinebilmeyi başarmasıdır. Klasik 4-komşuluklu KYA, nesne ait her hangi bir noktanın kenar olup olmadığını yorumlayabilmek için o noktanın kendisi, altı, üstü, sağı ve solundaki olmak üzere toplam 5 karşılaştırma yapar. Oysa geliştirilmiş KYA, nesnenin iç noktaları için sadece bir karşılaştırma yapar. Bu karşılaştırmada o noktanın nesneye ait olup olmadığını belirlemek içindir.

Geliştirilmiş KYA, bağımsız her nesne için maksimum iki referans noktası tespit ettiği düşünülürse, kenar izleme fonksiyonu maksimum görüntü içerisindeki nesne sayısının iki katı kadar çağrılır. Kenar izleme fonksiyonu ne kadar çok çağrılırsa geliştirilen KYA'nın zaman kriteri açısından performansı o kadar düşer.

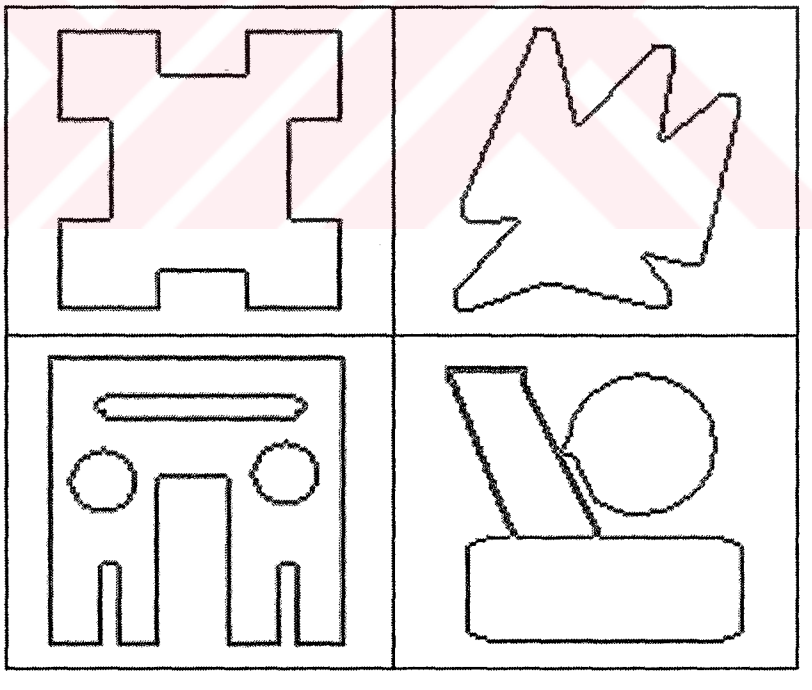
Geliştirilen KYA, yapay ve doğal görüntülerde hız kriterine yaptığı bu iyileştirme nesne tanıma işlemlerinde hayati bir önem taşımaktadır.



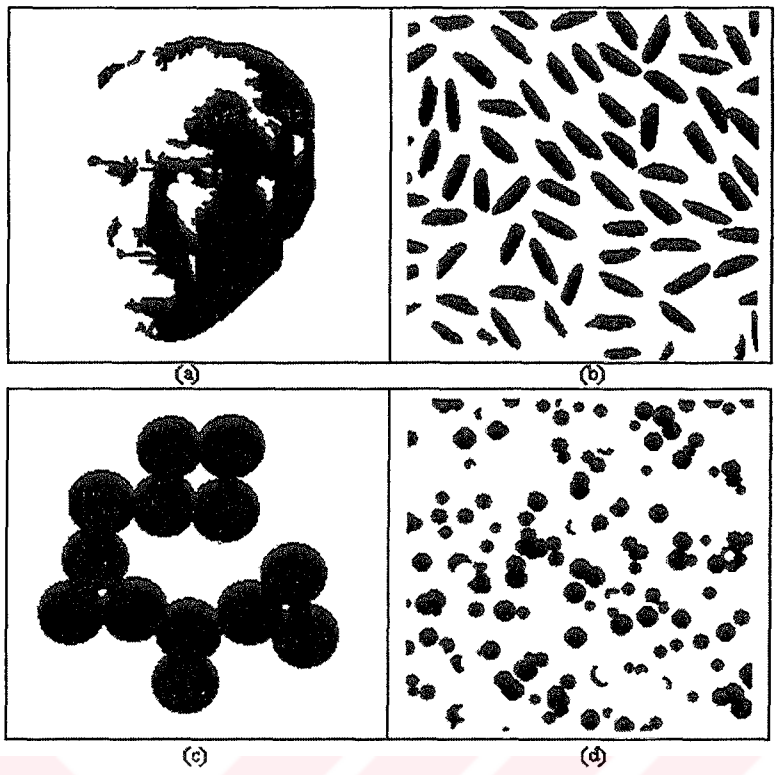
Şekil 3.8 Binary_Edge programının görüntüsü



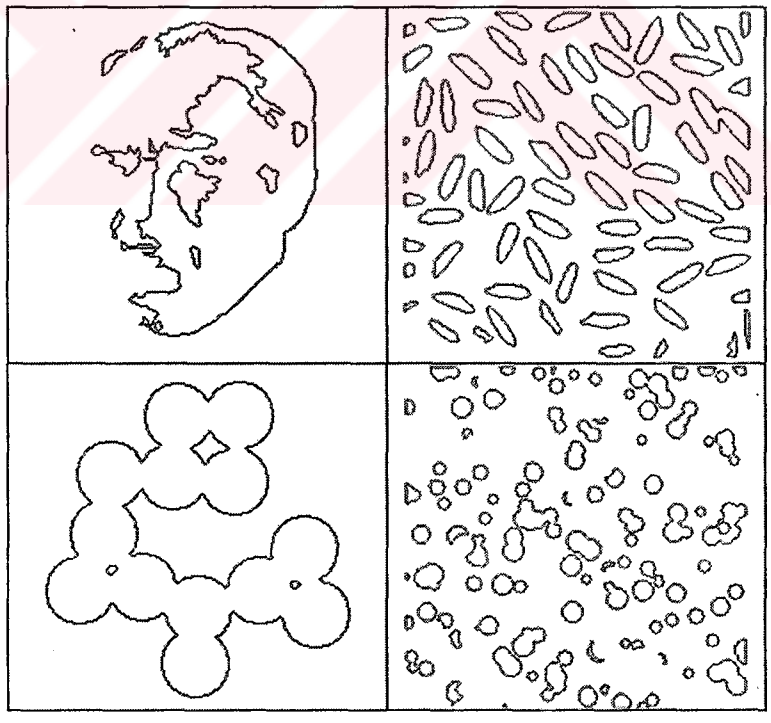
Şekil 3.9 Kullanılan yapay nesne görüntüleri (a) Nesne 1 (b) Nesne 2 (c) Nesne 3 (d) Nesne 4



Şekil 3.10 Yapay nesnelerin kenar görüntüleri



Şekil 3.11 Doğal nesne görüntüleri (a) Nesne 5 (b) Nesne 6 (c) Nesne 7 (d) Nesne 8



Şekil 3.12 Doğal nesnelerin kenar görüntüleri

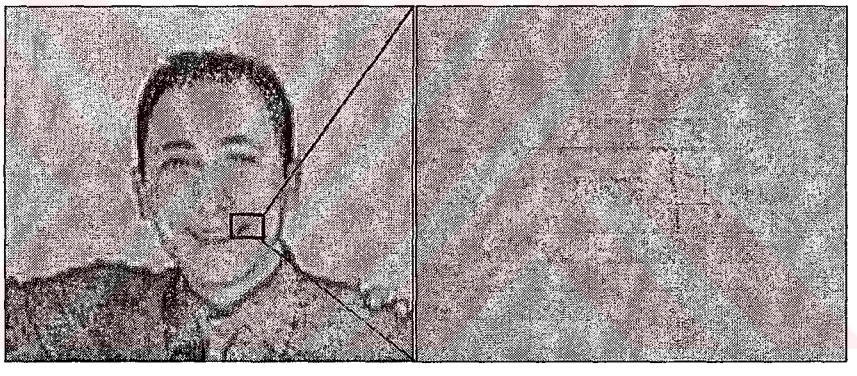
		Karşılaştırma Sayısı		Geçen Zaman (sn)		Boyut	Siyah Piksel Sayısı	Referans Nokta Sayısı
		Klasik Algoritma	Geliştirilen Algoritma	Klasik Algoritma	Geliştirilen Algoritma			
Yapay	Nesne1	34512	15433	0.421	0.203	100x100	15433	1
	Nesne2	28460	15033	0.343	0.205	100x100	4714	1
	Nesne3	26996	18328	0.327	0.285	100x100	4348	4
	Nesne4	26932	17640	0.328	0.255	100x100	4332	2
Doğal	Nesne5	84535	56869	1.024	0.797	209x183	11767	12
	Nesne6	140868	133189	1.385	2.777	256x256	19088	73
	Nesne7	127440	79302	1.641	1.172	256x256	15731	4
	Nesne8	119416	122894	1.562	3.031	256x256	13725	101

Şekil 3.13 4-komşuluklu KYA ile geliştirilen KYA'nın yapay ve doğal görüntülerdeki sonuçları

4. GRİ SEVİYELİ GÖRÜNTÜLERDE KENAR YAKALAMA

4.1 Giriş

Gri seviyeli görüntü tipi, görüntü işlemede en fazla kullanılan görüntü tiplerinden biridir. Görüntüdeki her nokta 0 ile 255 arasında bir gri seviye değerine sahiptir. Sadece iki seviyeli ve yapay görüntülerde karşılaştığımız ve Bölüm 2'de bahsedilen adım kenar tipinin aksine rampa kenar tipinin sıklıkla görüldüğü görüntü tipidir. Şekil 4.1'de gri seviyeli görüntüye bir örnek verilmiştir. Aynı zamanda, gerçek hayatta kullanılan gri seviye değerlerine sahip bu görüntü, bizlere noktalar arasındaki ilişkilerin ve geçişlerin nasıl olduğunu da gösterir.



Şekil 4.1 Gri seviyeli bir görüntü ve belli bir alanının büyütülmüş hali

Gerçek ortamdan elde edilen gri seviyeli görüntüler genellikle yüksek miktarda gürültü içerir. Görüntü üzerinde bu şekilde çalışmak oldukça zordur. Bu nedenle, bu tür görüntüler üzerinde görüntü işleme tekniklerini çalıştırmadan önce, gürültüyü ortadan kaldırmak veya azaltmak için makul bir filtre fonksiyonu kullanılır. Filtre fonksiyonu sayesinde, görüntüdeki istenmeyen gürültü işaretleri elenmiş olur. Elde edilen filtrelenmiş görüntü, bundan sonra anlatılacak klasik ve güncel kenar yakalama algoritmalarına giriş olarak verilir.

Klasik ve güncel algoritmalara giriş olarak Şekil 4.2'de gösterilen 512x512 boyutlarına sahip "Lena" görüntüsü verilir. Görüntü işleme alanında en fazla kullanılan

görüntülerden biri olan Lena görüntüsü, çok farklı özellikler ve yapılar içerdiğinden dolayı görüntü işleme uygulamalarında sıklıkla tercih edilir. Örneğin, bu görüntü, Lena'nın omzundaki pürüzsüzlük ile şapkası üzerindeki tüylerin dağınıklılığı bir arada içerir.

Şekil 4.2 Lena görüntüsü

4.2. Gri Seviyeli Görüntülerde Klasik Kenar Yakalama Operatörleri

Bu bölümde anlatılacak olan tüm klasik kenar yakalama operatörlerinde, kaynak görüntüdeki her $a \in R^x$ olmak üzere a_0, a_1, ..., a_7 noktaları, Şekil 4.3'de görüldüğü gibi güncel (i,j) noktasının saatin tersi yönünde numaralandırılmış 8-komşuluğundaki nokta değerlerini gösterir.

a_3	a_2	a_1
a_4	(i,j)	a_0
a_5	a_6	a_7

Şekil 4.3 Güncel (i,j) noktasının 8 komşuluğu

4.2.1 Prewitt ve Sobel Kenar Yakalama Operatörleri

Prewitt Kenar Yakalama Operatörü: Prewitt kenar yakalama operatörü [16] en eski ve en anlaşılır kenar yakalama metotlarından biridir. Bu operatör, kaynak görüntüdeki

her nokta için bir kenar vektörü hesaplar. Bu vektör kenar büyüklüğünü ve kenar yönünü gösteren değerler içerir. Kenar vektörünü hesaplamak için kullanılan en genel yaklaşım, Şekil 4.4'te gösterildiği gibi güncel noktanın 3x3 komşuluğundaki noktaları kapsayan bir maske kullanmaktır. Matematiksel formu denklem (4.1)'de gösterilen *u* maskesi, *x* eksenine karşılık gelen kısmi türevleri gösterir. Denklem (4.2)'de gösterilen *v* maskesi ise, *y* eksenine karşılık gelen kısmi türevleri gösterir. Denklem (4.3)'te, yatay ve düşey eksenden elde edilen kısmi türevler kullanılarak farklılığın büyüklüğü hesaplanır. Denklem (4.4)'te ise farklılığın yönü hesaplanır. Hesaplanan farklılık değeri, daha önceden belirlenmiş eşik değerinden büyükse arzulanan kenar elde edilmiştir. Eğer hesaplanan farklılık değeri eşik değerinden küçükse kenar noktası bulunamamıştır.

u=

-1	0	+1
-1	0	+1
-1	0	+1

v=

-1	-1	-1
0	0	0
+1	+1	+1

Şekil 4.4 Prewitt operatörünün kullandığı yatay ve düşey kenar yakalama maskeleri

$$u = (a_5 + a_6 + a_7) - (a_1 + a_2 + a_3) \tag{4.1}$$

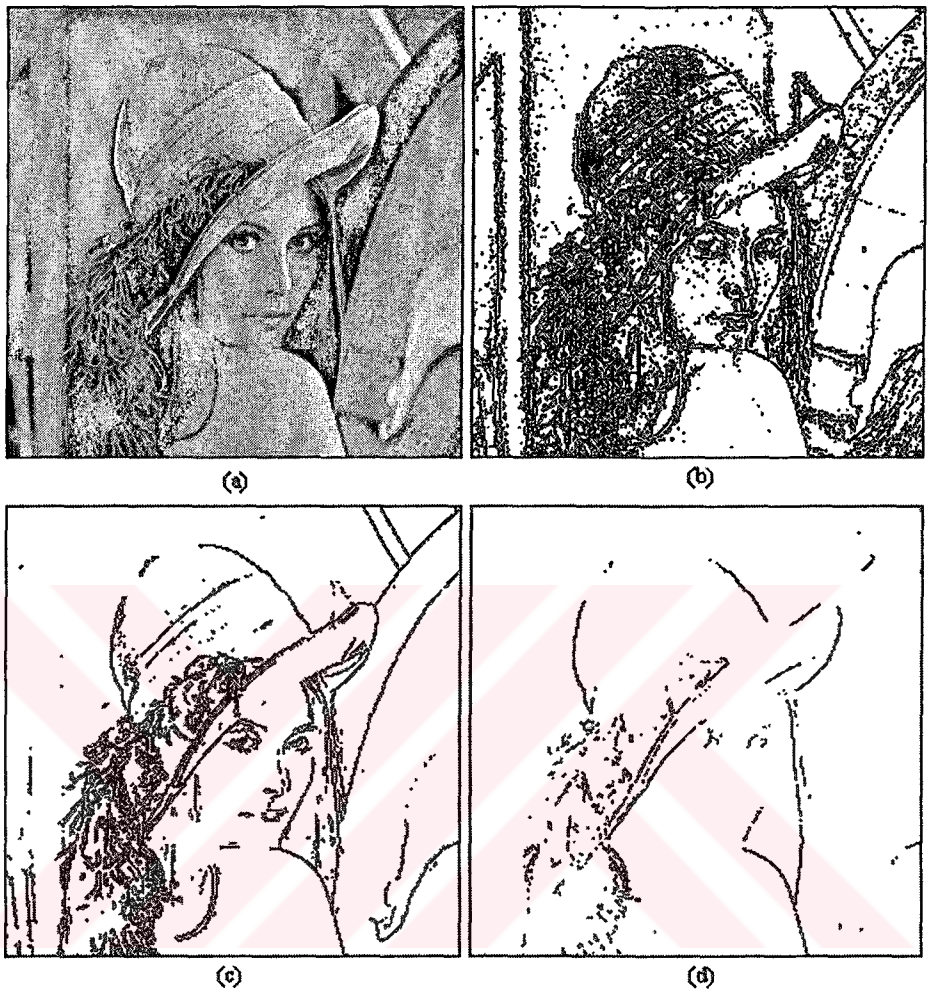
$$v = (a_0 + a_1 + a_7) - (a_3 + a_4 + a_5) \tag{4.2}$$

$$b(i,j) = (u^2 + v^2)^{1/2} \tag{4.3}$$

$$d(i,j) = \arctan\left(\frac{u}{v}\right) \tag{4.4}$$

Şekil 4.5'te, Prewitt algoritmasının Lena görüntüsü üzerinde, farklı eşik değerleri kullanılarak elde edilen sonuçları gösterilir. Buna göre, eşik değerinin çok düşük seçilmesi durumunda istenmeyen kenarların ortaya çıktığı görülür. Tersini

düşünürsek, eşik değerinin çok büyük seçilmesi durumun da arzu edilen kenarların yakalanmadığı görülür.



Şekil 4.5 (a) Orijinal Lena görüntüsü, Prewitt kenar yakalama algoritmasının sonuçları (b) *T=0.02* (c) *T=0.06* (d) *T=0.15*

Sobel Kenar Yakalama Operatörü: Optimal kenar yakalama operatörlerinden biride Sobel kenar yakalama operatörüdür [17]. Sobel operatörü ile Prewitt operatörü arasındaki fark, kenar yönlerinin çıkarılması hususundadır. Sobel operatörü görüntüdeki nesnelerin kenar yönlerinden bağımsızdır. Oysa Prewitt operatörü kenar yönlerinin çıkarılmasında Sobel operatörlerinden daha hassas davranır.

Sobel operatörü aslında uygun bir donanım kullanılarak, bir diziyi kaydırma yaklaşımı ile, gerçek zamanlı işlemlerde kullanılabilir. Her görüntü karesini saniyenin 1/30'da hesaplayabilir ve klasik video görüntülerini işleyebilir. Denklem (4.5) ve

denklem (4.6)'de gösterilen u ve v maskeleri, Sobel kenar yakalayıcının yatay ve düşey kenarları yakalamak için kullandığı operatörlerdir. Kenar yerinin belirlenmesinde etkin rol oynayan d eğim yönünün hesaplanması, denklem (4.8)'te gösterilir. Şekil 4.6'da gösterilen sobel kenar yakalama operatörleri gri seviyeli görüntüye uygulandığı zaman elde edilen kenar sonucu Şekil 4.7'de gösterildiği gibi olur.

$$u = (a_5 + 2a_6 + a_7) - (a_3 + 2a_2 + a_1) \tag{4.5}$$

$$v = (a_1 + 2a_0 + a_7) - (a_3 + 2a_4 + a_5) \tag{4.6}$$

$$b(i,j) = (u^2 + v^2)^{1/2} \tag{4.7}$$

$$d(i,j) = \arctan\left(\frac{u}{v}\right) \tag{4.8}$$

u=

-1	0	+1
-2	0	+2
-1	0	+1

v=

-1	-2	-1
0	0	0
+1	+2	+1

Şekil 4.6 Sobel operatörünün kullandığı yatay ve düşey kenar yakalama maskeleri

Şekil 4.7 (a) Orijinal Lena görüntüsü, Sobel kenar yakalama algoritmasının sonuçları (b) *T=0.02* (c) *T=0.06* (d) *T=0.15*

Maske Seçimi: Kenar noktalarını yakalamak için kullandığımız maskeler ile ilgili şu soruları yanıtlamalıyız. Kenar noktasını yakalayabilmek için nasıl bir maske seçmeliyiz? Sobel ve Prewitt operatörlerinin kullandığı maskeler acaba nasıl hesaplanır? Bu maskelerden hangisini tercih etmeliyiz?

Kenar noktasını yakalaya bilmek için kullanılan maskeler şu şekilde hesaplanır: Görüntü içerisindeki *(x,y)* noktasındaki yoğunluk değerinin $M_{x,y}$ olduğunu, bu noktanın denklem (4.9)'deki hesaplanan değere sahip olduğunu ve istenilen maskenin de denklem (4.10)'daki gibi olduğunu kabul edelim. O zaman hesaplanacak farklılığın büyüklüğü, $[\beta,\alpha]$ arasındadır. Görüntüdeki (x,y) noktasının 8 komşuluğundaki nokta değerleri de denklem (4.11)'deki gibi olur.

$$M_{y,x} = \alpha y + \beta x + \gamma \tag{4.9}$$

$$Gx = \begin{bmatrix} -a & -b & -a \\ . & . & . \\ a & b & a \end{bmatrix} \quad Gy = \begin{bmatrix} -a & . & a \\ -b & . & b \\ -a & . & a \end{bmatrix} \tag{4.10}$$

$$I(x,y) = \begin{bmatrix} -\alpha-\beta+\gamma & -\alpha+\gamma & -\alpha+\beta+\gamma \\ -\beta+\gamma & \gamma & \beta+\gamma \\ \alpha-\beta+\gamma & +\alpha+\gamma & \alpha+\beta+\gamma \end{bmatrix} \tag{4.11}$$

Maskeleme işlemini uyguladığımız zaman, denklem (4.12)'de gösterilen formül ile x yönündeki süreksizliğin büyüklüğü Gx ve y yönündeki süreksizliğin büyüklüğü Gy değerleri hesaplanır. Buna göre farklılık büyüklüğü denklem (4.13)'da hesaplanır.

$$Gx = 2\beta(2a+b)\ ,\ \ Gy = 2\alpha(2a+b) \tag{4.12}$$

$$G = 2(2a+b)\sqrt{\alpha^2+\beta^2} \tag{4.13}$$

Bu nedenle giriş filtre seçimindeki a ve b katsayıları belirlenirken $2(2a+b)=1$ eşitliğinin sağlanması beklenir. $a=b=1/6$ seçildiği zaman Prewitt operatörü, $a=1/8$ ve $b=1/4$ seçildiği zaman ise Sobel operatörü elde edilir.

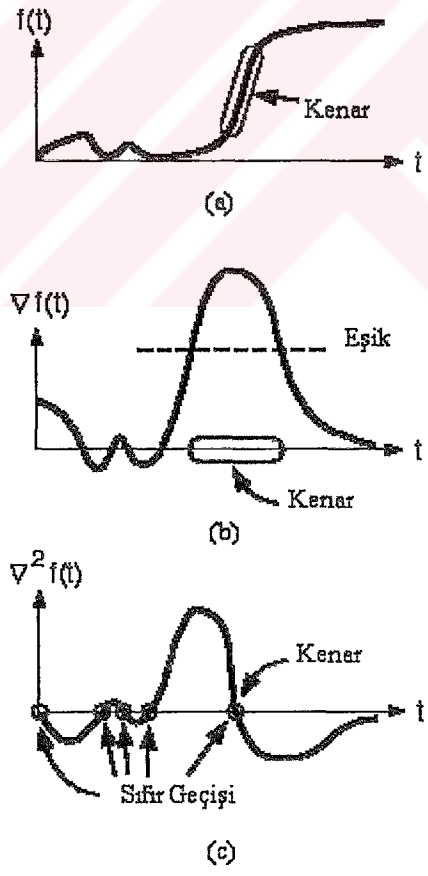
Yukarıda anlatılanların ışığı altında Prewitt ve Sobel kenar yakalama operatörleri için belirlenen olumsuzluklar aşağıdaki gibi listelenebilir.

1. İlk olarak, bu operatörler gürültüye eğilimlidir. Rasgele dalgalanmalara, orijinal değerlerden çok daha fazla duyarlıdırlar.
2. İlk dezavantajı ortadan kaldırmak için yumuşatma tekniğinin kullanılması önerilebilir. Fakat yumuşatma tekniği, var olan kenarların ve kenar yerlerinin kaybolma riskini beraberinde getirir.
3. Bu operatörler yoğunluk değişimlerinin yüksek olduğu alanlarda "ince kenar" üretebilirler.
4. İzotropik değillerdir. Yani görüntüdeki nesnelerin yönlerinin değişimlerinden etkilenirler.

5. İki boyutlu bir fonksiyondaki farklılığı hesaplamak için farklı iki maske kullanır.

4.2.2 Gaussian Filtreli Laplacian Kenar Yakalama Operatörü (LOG)

Etkili kenar yakalama operatörlerinden biride Laplacian kenar yakalama [18] operatörüdür. Şekil 4.8 (a)'daki gibi tek boyutlu bir f(t) işareti olduğunu düşünelim. Laplacian operatörüne göre, bu fonksiyonunun Şekil 4.8 (c)'de gösterildiği gibi ikinci türevindeki sıfır geçiş noktaları arzulanan kenar noktalarının yerlerini verir. Gürültülü görüntülerde çok tutarsızdır. Bu nedenle, Laplacian operatörünü yüksek frekanslı gürültülere sahip bir görüntü için kullanmak istersek, Gaussian filtresi ile birlikte çalıştırmalıyız. Laplacian operatörü ve Gausian filtresinin genellikle birlikte kullanıldığı için, LOG operatörü olarak adlandırılır. LOG operatörü şeklinden dolayı Meksika şapkasına olarak bilinir.



Şekil 4.8 Laplacian kenar yakalama operatörünün sıfır geçiş noktalarını yakalayışı (a) f(t) fonksiyonu (b) f(t) fonksiyonunun birinci türevi (c) f(t) fonksiyonunun ikinci türevi

Bir görüntüdeki en büyük farklılık noktası bükülmenin olduğu noktalardır. Bir bükülme noktası, Şekil 4.8'de gösterildiği gibi ikinci türevin sıfıra eşit olduğu yerlerde mevcuttur. İki boyutlu bir f(x,y) fonksiyonunun ikinci türevi denklem (4.14)'deki gibi tanımlanır. Denklem (4.14)'un hesap maliyetinin yüksek oluşundan dolayı, hemen hemen aynı sonucu üreten denklem (4.15) kullanılır. Popüler Laplacian maskelerinden birkaç tanesi Şekil 4.9'da gösterilir. Bu maskelerin görüntü üzerinde hızlı çalışması pratik hayatta tercih edilmesini sağlar.

$$\nabla^2 f = \left(\frac{\partial^2}{\partial^2 x^2} + \frac{\partial^2}{\partial^2 y^2}\right) f(x,y) = \frac{\partial^2 f(x,y)}{\partial^2 x^2} + \frac{\partial^2 f(x,y)}{\partial^2 y^2} \tag{4.14}$$

$$\nabla^2 f(m,n) \approx 4f(m,n) - [f(m-1,n) + f(m+1,n) + f(m,n+1) + f(m,n-1)] \tag{4.15}$$

0	-1	0	-1	-1	-1	1	-2	1
-1	4	-1	-1	8	-1	-2	4	-2
0	-1	0	-1	-1	-1	1	-2	1

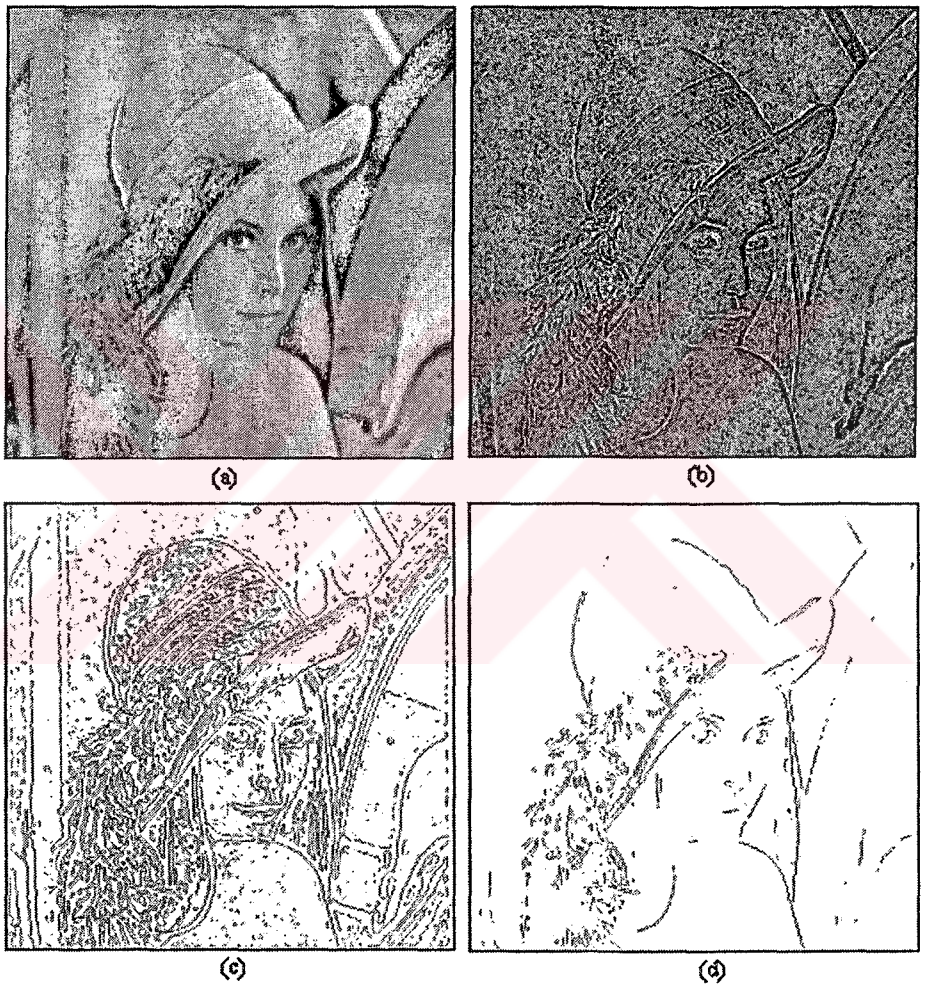
Şekil 4.9 En çok kullanılan LOG maskeleri

Fonksiyonun ilk türevinin büyüklüğü bizlere o noktada olabilecek bir kenarın varlığı hakkında bilgi verir. İkinci türevindeki sıfır geçiş noktaları ve ikinci türevin işareti kenar yerinin belirlenmesinde kullanılır. Ayrıca, Laplacian operatörü isotropictir. Bunun manası giriş görüntüdeki nesnelerin konumları ve yönlerinin değişmesinden etkilenmez.

Şekil 4.10'da Laplacian operatörünün farklı eşik değerlerine verdiği sonuçlar gösterilir. Buna göre, gürültüyü azaltmak için kullanılan Gaussian fonksiyonunun verimliliği ne kadar fazla ise, Laplacian operatörünün sonucu da o kadar etkili ve hatasız olur.

Yukarıda anlatılanların ışığı altında Laplacian kenar yakalama operatörü için belirlenen olumsuzluklar aşağıdaki gibi listelenebilir.

1. Laplacian operatörü fonksiyonun ikinci türevini hesapladığı için gürültüye karşı çok duyarlıdır.
2. Laplacian operatörü çift kenar yakalayabilir. Ayrıca kenar yönünü tespit edemez.
3. Kenar yakalama işleminde ikinci derecede bir rol oynar.
4. Kenar noktasındaki gri seviye geçişlerinin sert oluşu, Laplacian operatörünün verimliliğini azaltır.



Şekil 4.10 (a) Orijinal Lena görüntüsü (b) Laplacian Operatörünün Sonucu (c) Eşik fonksiyonunun sonucu (*T=0.001*) (d) *T=0.01*

4.2.3 Canny Kenar Yakalama Operatörü

Canny kenar yakalama operatörü [19], en karmaşık ve en başarılı kenar yakalama operatörlerinden biridir. Bu operatör, maskeleme teknikleri ile ifade edilemez. Canny, kenar yakalama problemini aşağıdaki üç kriter ışığında ele alır.

i. İyi kenar yakalama (detection): Yüksek kenar olma olasılığına sahip noktaları "kenar", düşük kenar olma olasılığına sahip noktaları ise "kenar değil" olarak sınıflandırmak.
ii. İyi kenar yeri belirleme (localization): Geçerli kenar yerine yakın olan noktaları kenar noktası olarak işaretlemek.
iii. Tek nokta kalınlığında kenar yakalama: Tek boyutlu bir işarette kenar çizgisini tek nokta kalınlığında göstermek.

i. ve ii. kritere bakıldığı zaman, Canny kenar yakalama operatörü, belirsiz olan bu ilkelere dikkatle yaklaşır. Eğer bir filtrenin kenar yakalama özelliği arttırılırsa, yer belirleme özelliği azalır. Tersi düşünülürse, bir filtrenin yer belirleme özelliği arttırılırsa kenar yakalama özelliği azalır. Canny, denklem (4.16)'de gösterilen işaret-gürültü (signal to noise-SNR) oranını kullanarak kenar yakalama özelliğini arttırır. Canny, gausian gürültüsü eklenmiş bir görüntüde kenar yerinin belirlenmesi ve SNR'nin sonucunun optimize eden bir filtrenin nasıl geliştirileceğini gösterir. Canny, iii. kriteri altında, i. ve ii. kriterlerden tatmin edici kenar noktalarını elde edebilmek için zoraki optimizasyon kullanır. Zoraki optimizasyon problemini çözdükten sonra, Canny optimal filtreye Gausianın ilk türevini kullanarak yaklaşabilir. Bu işlem, i. ve ii. kriterin performansının %20, iii. kriterin performansının ise %10 düşmesine neden olur.

$$SRN = \frac{\left| \int_{-w}^{w} G(-x) f(x) dx \right|}{n_0 \sqrt{\int_{-w}^{w} f^2(x) dx}} \qquad (4.16)$$

İki veya daha fazla boyutlu görüntülerde kenarlar yakalandığı zaman, yönsel kenarları bulmak için bazı mekanizmalar kullanılmalıdır. Kenar filtrelemeden sonra, Canny ardışıl eşik işlemini uygular. Bu işlem, son olarak elde edilen kenar görüntüsündeki gürültü etkisini azaltmayı sağlar.

Canny kenar yakalama algoritması sırası ile aşağıdaki adımları izler.

1. Orijinal görüntü Gaussian filtresinden geçirilerek yumuşatılır.
2. Filtrelenmiş görüntünün kısmi türevi için sonlu-fark yaklaşımı kullanılarak eğim büyüklüğü ve yönü hesaplanır.
3. Elde edilen eğim büyüklüğüne nonmaxima [28] baskısı uygulanır.
4. Kenarları yakalamak için çift eşik algoritması kullanılır.

1. **Yumuşatma:** Orijinal görüntüyü gösteren ifade eden I matrisini G(σ) gaussian yumuşatma filtresinden geçirme işlemidir. σ, gaussian fonksiyonun yayılım değeridir. Yumuşaklık derecesini kontrol eder. G(σ) ile I matrislerinin katlama (konvolisyon) sonucu ise yumuşatılmış görüntüyü bizlere verir.

$$S[i,j] = G[i,j,\sigma] * I[i,j] \tag{4.17}$$

2. **Eğimi Hesaplama:** İlk olarak, yumuşatılmış S[i,j] dizisinin eğimi x ve y kısmi türevleri sırasıyla P[i,j] ve Q[i,j] olmak üzere iki şekilde hesaplanır.

$$\begin{aligned} P[i,j] &\approx (S[i,J+1] - S[i,j] + S[i+1,j+1] - S[i+1,j])/2 \\ Q[i,j] &\approx (S[i,j] - S[i+1,j] + S[i,j+1] - S[i+1,j+1])/2 \end{aligned} \tag{4.18}$$

x ve y kısmi türevleri 2x2 kare üzerinde sonlu farkların ortalaması ile hesaplanır. Elde edilen eğilimin büyüklüğü ve yönü ise M[i,j] ve θ[i,j] olarak gösterilir.

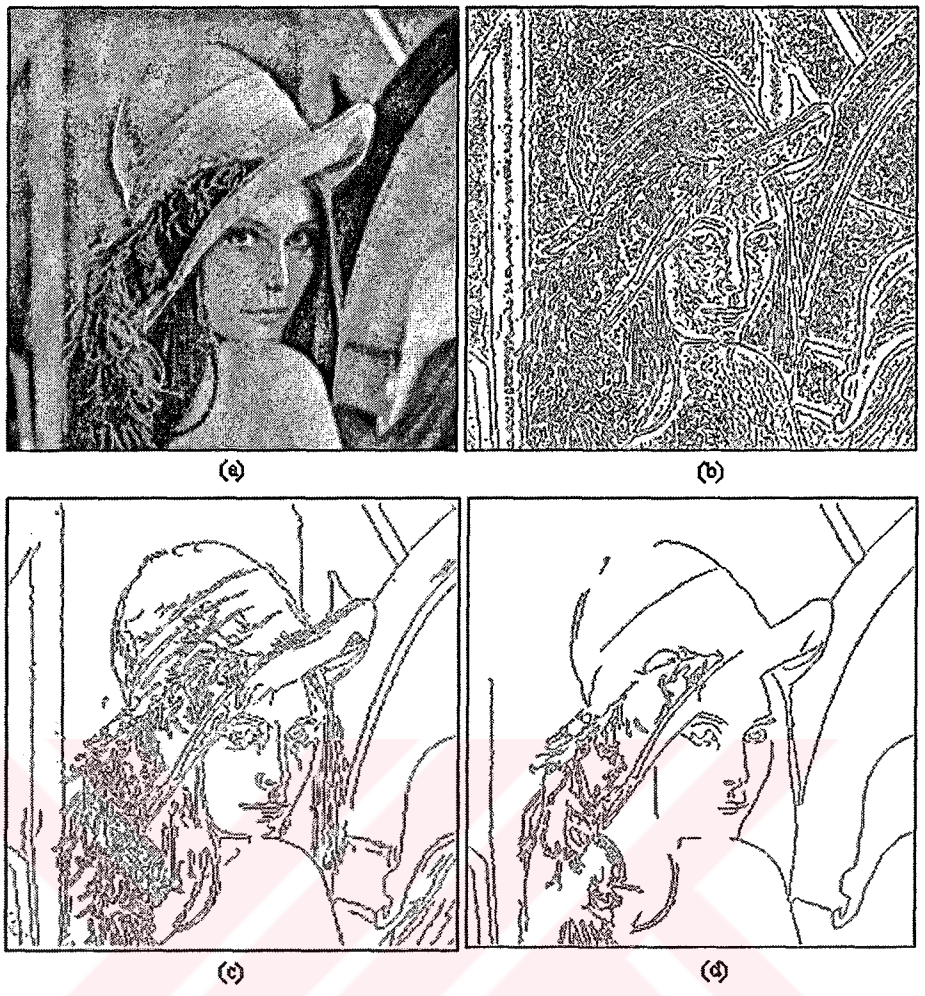
$$\begin{aligned} \mathrm{M}[i,j] &= \sqrt{\mathrm{P}[i,j]^2 + \mathrm{M}[i,j]^2} \\ \theta[i,j] &= \arctan(Q[i,j], P[i,j]) \end{aligned} \tag{4.19}$$

3. **Non-maksimum Baskısı:** Eğim (gradient) tabanlı metotlarda görüntü dizisine eşik işlemini uygulayabiliriz. Bu işlemde, noktalar arasındaki farklılığın büyüklüğü ile belirlenen eşik değeri karşılaştırılır. Farklılık eşik değerinden büyük ise "kenar" olarak

algılanırdı. Oysa Canny'nin bu probleme daha karmaşık bir çözüm önerisi vardır. Bu yaklaşıma göre kenar noktası tek nokta olarak tanımlanır. Ve bu noktanın kuvveti gradient yönündeki lokal maksimum değere sahiptir. Tek nokta kalınlığında olması istenen kenarlar için bu yöntem daha güçlüdür. Eşik yöntemi ile tespit edilen kenarları inceltmek içinde kullanılır. Bu işlem *non-maksimum baskısı* olarak adlandırılır. $N[i,j]=nms(M[i,j],\zeta[i,j])$ olarak gösterilir. Lokal maksimum nokta değerleri korunur.

4. **Eşik:** Kenar yakalamanın ilk adımında gerçekleşen yumuşatma ve üçüncü adımda gerçekleşen non-maksimum baskısına rağmen elde edilen $N[i,j]$ görüntüsü gürültülü ve ince arka plan dokusundan dolayı birçok yanlış kenar parçaları içermektedir. Bu yanlış kenar parçalarının kontrast değeri küçüktür. Non-maksimum baskısından dolayı elde edilen görüntüdeki küçük değerli yanlış kenar parçacıklarını eleyebilmek için küçük bir eşik değeri sunulmaktadır. Önerilen bu eşit yaklaşımı zordur ve hata içerir. Bu zorluğundan dolayı daha etkili sonuç veren iki eşik değeri kullanılır.

Bu problemin üstesinden gelmek için elde edilen $N[i,j]$ matrisine σ_1 ve σ_2 iki eşik değeri uygulanır ($\sigma_2=2*\sigma_1$). Belirlenen eşik değerleri sonucunda, $T_1[i,j]$ ve $T_2[i,j]$ adlı iki tane kenar görüntüsü elde edilir. $T_2[i,j]$ daha az hatalı kenar içerir. Fakat kenar çizgilerinin son noktasında bazı hatalı kenarlar içerir. Bu noktalarda $T_1[i,j]$ görüntüsü kullanılır. Böylece bir eşik değeri seçme problemi çözüme kavuşturulur.



Şekil 4.11 (a) Orijinal Lena görüntüsü (b) Canny kenar yakalayıcının sonucu t_1=0.001, t_2=0.002 (c) t_1=0.01, t_2=0.02 (c) t_1=0.3, t_2=0.6

5. MODEL TABANLI YAPAY SİNİR AĞI İLE KENAR YAKALAMA

5.1 Giriş

Bu bölümde, geleneksel yapay sinir ağlarından farklı bir mimariye sahip olan özel tasarlanmış MYSA kullanılarak, gri seviyeli bir görüntü içerisinde göze çarpan renk değişimlerini tespit edebilen bir sistemden bahsedilir. MYSA, etkileşimli çalışması sayesinde belirli özellikleri tespit edebilme yeteneğine sahiptir. Ayrıca, eğitme işlemi sonucunda, öğrenme datasının içermediği değişik görüntüler ile karşılaşıldığı zaman benzer özellikleri belli bir küme halinde genelleyebilme özelliğine de sahiptir [28]. Bu çalışmada, kenar yakalama problemi için özel tasarlanmış MYSA hakkında gerekli bilgiler verilmiştir.

Kenar yakalama, özellik çıkarma işleminde önemli bir alt problemdir. Bu işlem genellikle iki aşamadan oluşur. Birinci aşama, güncel noktanın yoğunluğu komşu noktalar ile karşılaştırılarak yoğunluk değişimleri tespit edilir. İkinci aşamada ise, elde edilen değişimlerin önceden belirlenmiş bir eşik değeri ile karşılaştırılarak "önemli" bir değişim olup olmadığı tespit edilir. Başka bir deyişle, ilk aşamada uygun değişim yerleri tespit edilir. Örneğin Prewitt ve Sobel kenar yakalama operatörleri bu aşamaya ihtiyaç duyarlar. Diğer taraftan, ikinci aşamada ise kenar büyüklüğünün öneminin tespit edilmesi için bir eşik değerine ihtiyaç duyulur [29].

Basit kenar yakalayıcılar, iki değerli (0 ve 1) bir kenar üretmek için global bir eşik değeri tanımlarlar. Bu yöntem gürültülü görüntülerde tatmin edici bir sonuç vermez. Böylece daha karmaşık bir yaklaşım olan ve sözde ardışıl bir eşik değeri kullanan Canny kenar yakalama operatörlerine ihtiyaç duyulur. Bu yöntemde, "önemli" olarak adlandırılan kenarlar, küçük bir eşik değerini geçen noktaların büyüklükleri ile yüksek bir eşik değerini geçen noktaların büyüklükleri diye sıralanırlar. Kenar çizgisinin tek nokta kalınlığında olması için bu yöntemlere ilaveten, daha fazla ortak parametre tanımlayarak sıfır geçişini yakalayan Laplacian (LOG) filtrelerine ihtiyaç duyulur [30, 31].

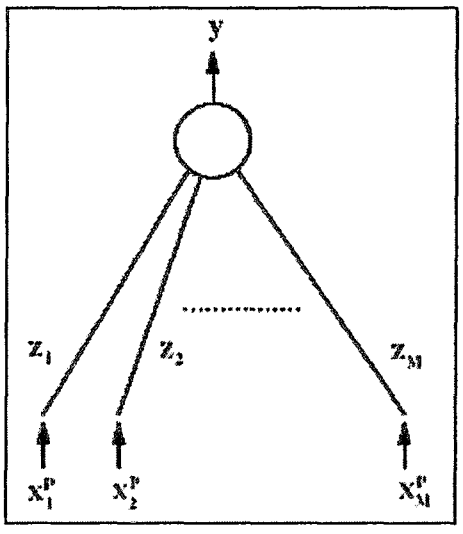
Bu bölümde, kenar yakalama için gerekli olan parametre kümesini hesaplaya bilen yeni bir MYSA mimarisinden bahsedilir. Bu yaklaşımın yararlarını, parametre alanında önemli bir azalma, gelişmiş genelleme, ve bir öğrenme prosedürü (önceden belirlenen bir değişken ile sınırları garanti eden) olarak ifade edebiliriz. Daha karmaşık kenar yakalama işlemleri için daha büyük parametrelerin olması beklenir. Onun sunumu için, ilişkili ağırlıklar kümesine sahip bir yapay sinir ağının tercih edilmesi mantıklı bir seçimdir [32, 33].

5.2 Kenar Yakalama için önerilen MYSA Modeli

Geleneksel MYSA mimarileri hesaplama birimi olarak ağırlık parametreli model tabanlı hücrelerin bir kümesini içerir. Her bir alt ağın çıkışı, lokal hücre çıkışlarının lineer bir birleşimi olarak tanımlanır. Güncel uygulamalar için bu mimarinin benimsenmesi makul bir yaklaşım olabilir. Fakat, kenar yakalama probleminin farklı ihtiyaçlarından dolayı, alternatif bir hücre modeli olan, giriş parametreli model tabanlı hücreyi kullanmak daha uygundur. Ayrıca geleneksel yapay sinir ağı modellerinde alt ağ çıkışlarını belirlemek için kullanılan lineer birleştirme işlemi yerine, tek bir alt ağ içerisindeki tüm model tabanlı hücrelere "kazanan yarışmacı" işlemi uygulanır [34].

5.2.1 Giriş Parametreli Model Tabanlı Hücre

Ağırlık parametreli model tabanlı hücrede olduğu gibi, yüksek boyutlu ağırlık vektörü olan $p \in R^N$'i düşük boyutlu ağırlık vektörü olan $z \in R^M$'e gömerek haritalamak yerine, giriş parametreli model tabanlı hücre tasarlanmıştır. Öyle ki, yüksek boyutlu giriş vektörü olan $x \in R^N$, düşük boyutlu $x^p \in R^M$ vektörü ile haritalanır. Bu varsayım altında, problemi x^p vektörüne indirgemiş oluruz ve yüksek boyutlu x vektörünün tüm özelliklerini sunabiliriz. Eğer x giriş vektör kümesi için böyle bir haritalama yapılırsa, ağırlık vektörü, büyük boyutlu p vektörü yerine küçük boyutlu z vektörü olarak sunulabilir.



Şekil 5.1 Giriş parametreli model tabanlı hücre

Giriş parametreli model tabanlı hücre modeli Şekil 5.1'de gösterilmiştir. Biçimsel olarak, $P = R^N \longrightarrow R^M$ haritasının varlığını kabul ederiz. Öyle ki, $x^P = P(x) \in R^M$ M<N olmak üzere model tabanlı hücre işlemi denklem (5.1)'deki gibi tanımlanır.

$$\begin{aligned} y_s &= fs(x, p) \\ &= fs(x^P, z) \end{aligned} \tag{5.1}$$

Farklı yapıya sahip her iki model tabanlı hücrede, küçük boyutlu bir ağırlık uzayında daha etkili ilerleyebilmek için optimal ağırlıkları aramaya izin verir. Yukarıda bahsedilen hücre yapılarından hangisini kullanacağımıza problemin yapısını inceledikten sonra karar verdik. Sırasıyla, ağırlıklar için hazırlanan *M* vektörünün veya girişler için oluşturulan *P* vektörünün uygunlukları incelenir. Eğer problemdeki parametre alanı, yüksek boyutlu bir alanda düşük boyutlu bir yüzey ile sınırlandırılırsa, o zaman ağırlık parametreli model tabanlı yapay hücreleri kullanmak uygun bir yaklaşımdır. Diğer taraftan, ağırlık vektörünün bu şekilde sunumu uygun ve kolay değildir. Bu durumda giriş parametreli model tabanlı hücreyi kullanmak daha uygundur. Güncel kenarları belirleyen MYSA, orijinal giriş vektörünü (3x3'lük kenar nokta değerlerine sahip pencere) ağırlıklı iki gri seviyede (0 ve 1) sunulan iki boyutlu bir vektörde haritalar.

5.2.2 Alt ağın çıkışını belirlemek

Güncel uygulamalar (örneğin etkileşimli düzenleme problemleri), genellikle alt ağın çıkışını belirtmek için lineer birleşimli bir yaklaşım kullanır. Oysa bu çalışmada klasik birleştirme yaklaşımı yerine denklem (5.2)'de gösterilen "kazanan yarışmacı" işlemi lokal kazananı belirlemek için bir alt ağ içerisindeki tüm hücrelere uygulanır.

$$p_{s^*_r} = \underset{s_r}{\operatorname{argmin}}\, {}_r(x, p_{s_r}) \tag{5.2}$$

$p_{s^*_r}$ kazanan hücrenin indeksidir. $\phi(x, p_r)$ alt ağın çıkışı olarak tanımlanır ve alt ağ içerisindeki kazanan hücre çıkışına eşittir.

$$\phi(x, p_r) = {}_r(x, p_{s^*_r}) \tag{5.3}$$

Bu yarışma işlemine uygun olarak lokal hücre çıkışını hesaplamak için Euclidean uzaklığı benimsenir.

$${}_r(x, z_{s_r}) = \left\| x - p_{s_r} \right\| \tag{5.4}$$

Denetlemesiz örnekleri sınıflandırmak için özellikle bu sınıf ağlar tercih edilir. Her bir örnek sınıf, farklı karakteristiklere sahip birkaç alt kümeden oluşur. Her bir birinci örnek sınıfını bir alt ağ'a işaretleyebiliriz. Birinci örnek sınıf altında, her bir ikinci sınıfı ise seçilen alt ağ içindeki bir hücreye işaretleyebiliriz.

Giriş parametreli model tabanlı hücreye, yüksek boyutlu bir uzayda model tabanlı z_{s_r} vektörünü gömmek yerine, denklem (5.5)'te gösterildiği gibi, P operatörü içerisinden düşük boyutlu R^M için $x \in R^N$ giriş vektörü haritalanır.

$${}_r(x, p_{s_e}) \equiv {}_r(x^P, z_{s_e}) \tag{5.5}$$

$$= {}_r(P(x), z_{s_e}) \tag{5.6}$$

$x^P = P(x)$, yüksek boyutlu R^N'deki x'e karşılık gelen düşük boyutlu giriş vektörüdür.

5.2.3 Kenar belirleme ve yakalama

Farklı parlaklık değerleri altında önemli bir kenar özelliği oluşturmak için, gri seviye değerlerinde belirgin bir değişime ilişkin, hedeflenen model olan insanın bir çok üstünlüğünü gözlemlememiz, bizleri güncel ağ mimarisini benimsemeye sevk etti. Kenar yakalama problemine, bu kriterleri uygulayabilmek için, MYSA'nın ağ mimarisini benimsemek, makul bir yaklaşımdır. Geliştirilen MYSA'da, her bir alt ağ farklı bir parlaklık seviyesine karşılık gelecek şekilde dizayn edilir. Bu çalışmada iki boyutlu bir kenar vektörü $w \in R^2$ "kenar örneği" tanımlanır. Kenarın her iki tarafındaki gri seviye değerleri, ağırlıklı iki değer kullanılarak ifade edilir.

İnsan kaynaklı öğrenme örneklerinden çıkartılan prototiplerin ihtiyaçlarından dolayı, "denetimsiz rekabetçi yarışma" modelinin benimsenmesi gayet normaldir. Bu modele göre, her bir prototip, bir hücrenin ağırlık vektörü olarak sunulur. Yarışma sonucunun doğal galibi, MYSA mimarisindeki alt kümelerin hangisinin kullanılması gerektiğini bizlere gösterir. Bu işlem bizlere, her bir alt ağ içerisinde sadece lokal kazanan hücrenin ağırlık vektörünü değiştirme kolaylığını sağlar. Ayrıca, *r.* alt ağ için yerel hücre çıkışı, doğru kenar örneği ve doğru kenar prototipi arasındaki denklem 5.7'de gösterildiği gibi Euclide farkı kullanılarak hesaplanır.

$$_r(x^P, z_{s_r}) = \left\| x^P - z_{s_r} \right\| \tag{5.7}$$

$x^P \in R^2$ doğru kenar örneğidir. Ve z_{s_r}, *r.* alt ağın *s.* kenar prototipidir. Doğru kenar örneği genellikle $x \in R^N$ olmak üzere gri seviye değerleri içeren bir pencerede belirtilir. Diğer bir deyişle $P = R^N \longrightarrow R^2$ olan bir harita türetilmelidir. Denklem (5.8), giriş parametreli model tabanlı bir hücreye karşılık gelir.

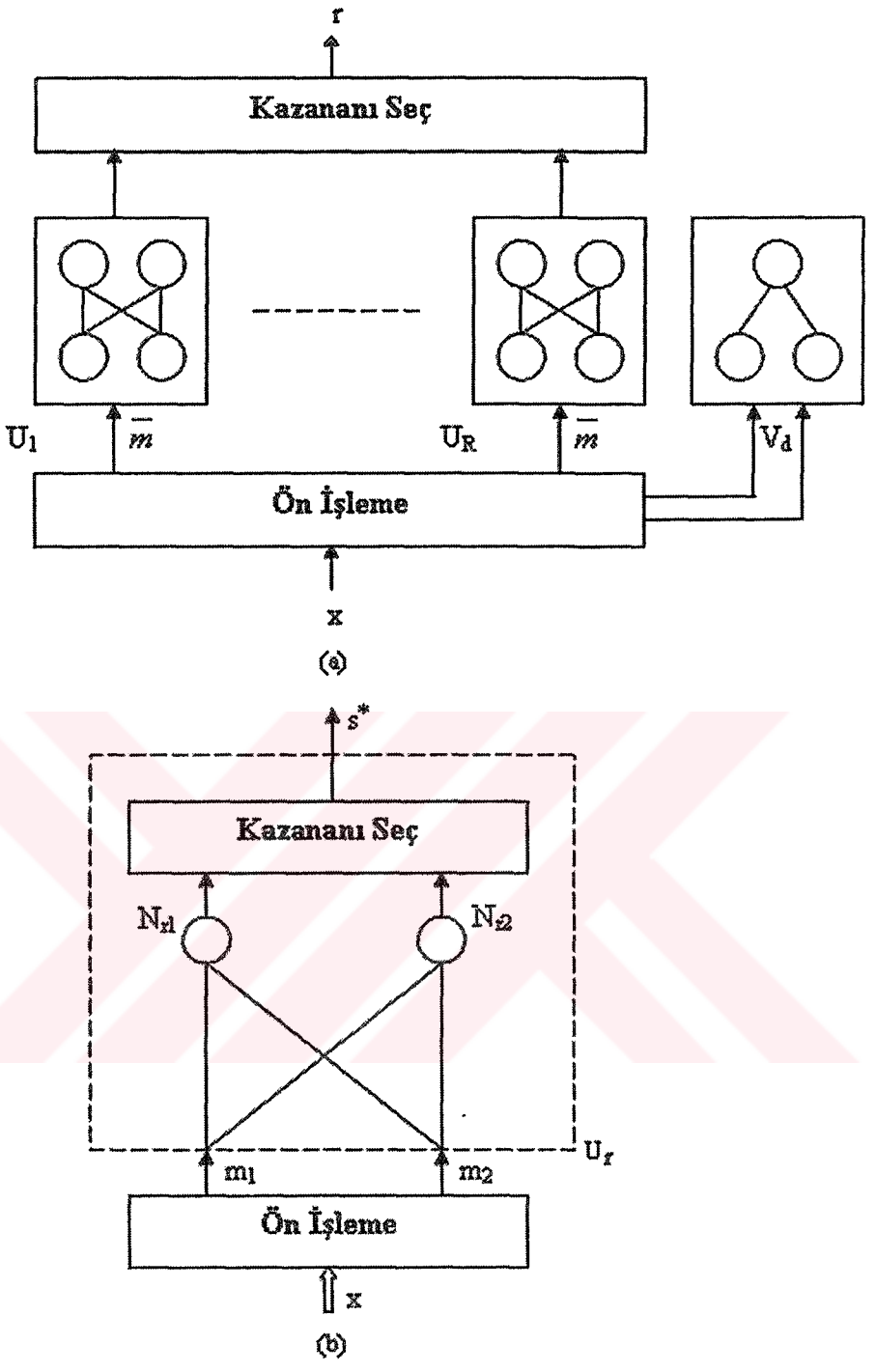
$$x^P = P(x) \tag{5.8}$$

5.3 Kenar yakalama için geliştirilen ağ mimarisi

Geliştirilen MYSA mimarisi numaralanmış alt ağlardan oluşur. Alt ağlar içerisindeki her bir hücre, öğrenme örneklerinin belirlenmiş bir alt kümesini kodlar. Başlangıç öğrenme aşamasında, öğrenme kümesindeki kenar örnekleri, alt ağlar arasında denetimsiz yarışma işlemi sayesinde, farklı alt kümelere bölünür. Sonraki tanıma aşamasında ise, kenar noktaları, değişik alt ağlar ile ilgili kodlanmış şablonlar ile güncel kenar noktaları karşılaştırılarak "kenar" veya "kenar değil" sonucu üretilir. .

MYSA mimarisini benimsememizdeki maksat kenar yakalama işlemindeki gözlemlerimizdir. Önceki yaklaşımlarda benimsendiği gibi tek parametreli bir küme kullanmak yerine farklı lokal bilgilere karşılık gelen çoklu eşik değeri kümesini benimsemek daha etkilidir.

Her bir alt ağ, arka planı farklı parlaklık değerine sahip bir kenar şablonunu kodlar. Alt ağdaki her bir hücre, parlaklık seviyelerine karşılık gelen kenar örneğinin mümkün değişik durumlarını kodlar. Kenar yakalamak için geliştirilen MYSA'nın global ağ mimarisi ve alt ağ mimarisi Şekil 5.2'de gösterilir. Kenar örneği için MYSA'nın kodlama şeması aşağıdaki bölümde tanımlanır.



Şekil 5.2 Kenar yakalamak için geliştirilen MYSA mimarisi (a) Global ağ mimarisi (b) Alt ağ mimarisi

5.3.1 Kenar Bilgisini Belirleme

Kenar yakalama işlemi güncel noktanın NxN komşuluğunda gerçekleşir. N, güncel pencere boyutudur ve genellikle 3 (3x3) veya 5 (5x5) değerini alır. N=3 olduğu zaman her pencere içerisinde toplam 9, N=5 olduğu zaman her pencere içerisinde toplam 25 gri seviye değeri bulunur. Güncel pencerenin gri seviye değerlerine karşılık gelen $x=[x_1,x_2,...,x_{N^2}]^T \in R^N$ vektörünün ortalama değeri denklem (5.9)'da gösterildiği gibi hesaplanır.

$$\bar{x}=\frac{1}{N^2}\sum_{n=1}^{N^2}x_n \tag{5.9}$$

Hesaplanan bu ortalama değeri ile güncel penceredeki gri seviye değerleri kullanılarak pencere içerisindeki iki temel gri seviye değerine karşılık gelen $m=[m_1,m_2]^T \in R^2$ vektörü elde edilir. Buna göre, güncel pencere içerisinde $\bar{x}$ ortalama değerinden küçük olan noktaların ortalaması m_1, büyük olan noktaların ortalama değeri ise m_2 olarak hesaplanır. m_1 ve m_2 değerleri kullanılarak genel ortalama değeri $\bar{m}$ hesaplanır. m_1 ve m_2 değerlerini hesaplarken kullanılan I(.) fonksiyonu, kendisine gönderilen şart eğer doğru ise 1 değilse 0 değerini üretir.

$$m_1=\frac{\sum_{i=1}^{N^2}I(x_i<\bar{x})x_i}{\sum_{i=1}^{N^2}I(x_i<\bar{x})} \tag{5.10}$$

$$m_2=\frac{\sum_{i=1}^{N^2}I(x_i\geq\bar{x})x_i}{\sum_{i=1}^{N^2}I(x_i\geq\bar{x})} \tag{5.11}$$

$$\bar{m}=\frac{m_1+m_2}{2} \tag{5.12}$$

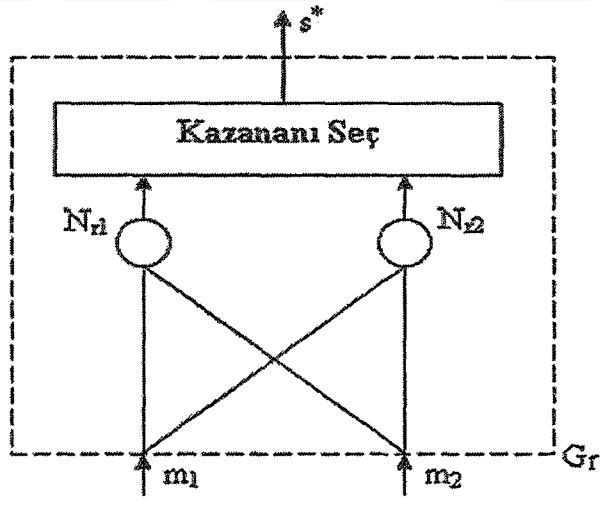
5.3.2 U_r Alt Ağı

Daha öncede tanımlandığı gibi, $r = 1,2,...,R$ olmak üzere, U_r her bir alt ağını gösterir. Bu alt ağ, bir prototipin arka plan parlaklık değerini ifade eden p_r değeri ile ilişkilendirilir. p_r değerine en yakın ortalamaya sahip *NxN* boyutundaki güncel pencere diğer kodlamalar için U_r alt ağına gönderilir. Görüntüdeki W penceresi, denklem (5.10), (5.11) ve (5.12)'deki formüllerde gösterildiği gibi m_1, m_2 *ve* $\overline{m}$ değerleri ile ilişkilendirilir. Bu değerler göz önüne alınarak aşağıdaki şartlar gerçekleşirse, W penceresi U_{r*} alt ağına bağlanır.

$$p_{r*} \in [m_1, m_2] \tag{5.13}$$

$$\left|\overline{m} - p_{r*}\right| < \left|\overline{m} - p_r\right|, \quad r = 1,...,R, \quad r \neq r* \tag{5.14}$$

NxN boyutlu tüm W pencereleri, arka plan parlaklığı ile aynı seviyede olan her bir alt kümenin üyeleri olacak şekilde bölümlenir. Denklem (5.13) ve (5.14)'deki şartları uygunluk için tercih ederiz. Şekil 5.3'de tek bir alt ağ mimarisi gösterilir.



Şekil 5.3 Tek bir G_r alt ağın mimarisi.

5.3.3 U_r Alt ağının $N_{r,s}$ hücresi

Her bir U_r alt ağı $s = 1,2,...,S$ olmak üzere S tane $N_{r,s}$ hücresi içerir. Bu hücre p_r genel parlaklık değeri altında değişik durumları gösteren yerel kenar şablonlarını kodlar. Her bir hücre $w_{r,s} = [w_{rs,1}, w_{rs,2}]^T \in R^2$ bir ağırlık vektörü ile ilgilidir. $W_{r,s}$ ağırlık vektörü NxN boyutundaki pencerelerin iki temel gri seviye değerini içerir. Her bir W penceresi denklem (5.15)'deki şart doğrultusunda $N_{r,s}$ hücresine işaretlenir.

$$\|m - w_{r^*s^*}\| < \|m - w_{rs}\| \qquad s = 1,...,S \ , \qquad s \neq s^* \tag{5.15}$$

S=2 olduğunu kabul edelim. Böylece her alt ağda iki tane $V_{r,s}$ hücresi olacaktır. Hücrelerden biri giriş örneğinin zayıf kenar olma olasılığını gösterir, diğeri ise güçlü kenar olma olasılığını gösterir. Buna karşılık s=1,2 olmak üzere w_{r^*s} ağırlık vektörlerinden bir tanesi zayıf kenar örneğinin $W^l{}_{r^*}$ ağırlığını, diğeri ise kuvvetli kenar örneğinin $W^s{}_{r^*}$ ağırlığını gösterir. Güçlü ve zayıf kenar örneklere karşılık gelen hücrelerin belirlenmesi aşağıdaki şartlara bağlıdır.Buna göre;

- $w^l{}_{r^*} = w_{rs'}$, $s' = \arg\min_s (w_{r^*s,2} - w_{r^*s,1})$
- $w^u{}_{r^*} = w_{rs''}$, $s'' = \arg\max_s (w_{r^*s,2} - w_{r^*s,1})$

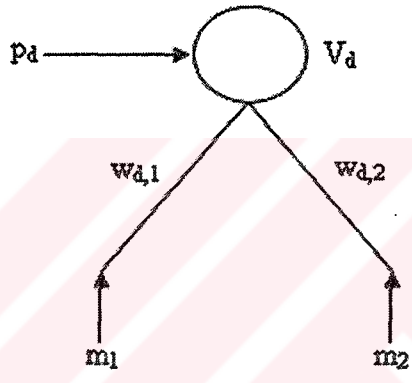
Zayıf kenar örneği $W^l{}_{r^*} = \left[W^l{}_{r^*,1}, W^l{}_{r^*,2}\right]$ geleneksel kenar yakalama algoritmalarında kullanılan eşik değeri ile benzer rol oynar. Görülebilir kenarların en düşük limitini belirler. Kenar izleme için görüntüdeki muhtemel başlangıç noktalarını belirlemede yararlıdır.

5.3.4 Dinamik kenar izleme hücresi

Geliştirilen MYSA mimarisinde, U_r alt ağı ve N_{rs} hücresinin yanında, birde bağımsız bir hücre olan V_d dinamik kenar izleme hücresi vardır. Başka bir deyişle; bu

hücre kapsam olarak geneldir ve herhangi bir U_r alt ağına bağlı değildir. Dinamik V_d hücresi aslında hücre ile alt ağ arasında melez bir yapıya sahiptir. Dinamik ağırlık vektörü olan $w_d = \lfloor w_{d,1}, w_{d,2} \rfloor$ hücresini içerir. Ayrıca p_d girişine de sahiptir ki bu giriş her bir alt ağın parlaklık seviyesini gösteren p_r değerine benzer.

Bu hücrenin maksadı, kenar izleme işlemi esnasında arka planın gri seviyeli parlaklık değerlerinin değişimini izlemektir. Bu hücre öğrenme aşamasında pasiftir. Tanıma aşamasında ve birincil kenarı yakalama aşamasında dinamik ağırlık vektörü w_d ile parlaklık seviyesi göstergesi p_d sürekli değişir. Böylece başlangıç kenar noktasından daha az önemli olan diğer kenar noktalarını tespit eder. Şekil 4'de dinamik kenar izleme hücresinin yapısı gösterilir.



Şekil 5.4 Dinamik kenar izleme hücresinin yapısı

5.3.5 İkili seviyeli kenar belirleme

m vektörüne sahip güncel W pencereli bir giriş örneğinin, w_{r^*,s^*} ağırlık vektörüne sahip N_{r^*,s^*} hücresi ile ilişkilendirildiğini düşünelim. $x = [0,1,2,...,255]$ arasındaki gri seviye değerlerine sahip $x \in R^{N^2}$ giriş örneklerini, b=[0, 1] değerlerine sahip $b \in B^{N^2}$ çıkış değeri ile haritalamak gerekir. Bu amaca ulaşabilmek için, denklem (5.16)'daki gibi $Q : R^{N^2} x R^2 \longrightarrow B^{N^2}$ haritasını tanımlarız.

$$b = Q(x, w_{r^*s^*}) = \left[q(x_1, w_{r^*s^*}),..., q(x_{N^2}, w_{r^*s^*})\right]^T \in B^{N^2} \qquad (5.16)$$

$$q(x_n, w_{r^*s^*}) = \begin{cases} 0 & if \left|x_n - w_{r^*s^*,1}\right| < \left|x_n - w_{r^*s^*,2}\right| \\ 1 & if \left|x_n - w_{r^*s^*,1}\right| \geq \left|x_n - w_{r^*s^*,2}\right| \end{cases} \quad (5.17)$$

Şekil 5.5'de geçerli kenar bilgileri için ikili kodlanmış örnekler yer almaktadır (*N=3* için). Başlangıç öğrenme aşaması esnasında, her bir öğrenme sonucu elde edilen iki seviyeli kenar örnekleri, *C* adı altında bir kenar konfigürasyon kümesine yüklenir. Bu kümede, tüm kenar örnekleri bulur. Şekil 5.6'da ise bir mevcut kenar örneklerinden her hangi birini kullanarak yeni bir kenar örneği üretme işlemi gösterilmektedir. Buna göre bir kenar örneğinin 45^0'lik döndürülmesi sonucu farklı bir kenar örneği elde edilir.

0	0	0
0	0	0
1	1	1

0	0	1
0	0	1
0	0	1

0	1	1
0	0	1
0	0	0

Şekil 5.5 Geçerli kenar bilgisi örnekleri

0	0	0
0	0	0
1	1	1

$R_{\pi/4}$

0	0	0
0	0	1
0	1	1

Şekil 5.6 $R\pi_{/4}$ işlemi örneği

5.3.6 Genel MYSA mimarisinin uygunluğu

Güncel kenar yakalama ağları genel olarak MYSA mimarisini kullanır. Bu mimarinin özelliklerini şu şekilde belirtebiliriz: U_r alt ağları, $r = 1,2,...R$ değerine sahiptir ve her bir alt ağdaki N_{rs_r} hücresi, $s_r = 1,...,S_r$ kadar hücre içerir. Güncel uygulamalarda genellikle $S_r = 2$ değerini alır.

Ağırlık parametreli model tabanlı yapay sinir ağlarında bilindiği gibi yüksek boyutlu bir uzayda yer alan ağırlıklar düşük boyutlu bir uzayda haritalanırdı. Oysa biz, yüksek boyutlu giriş değerlerine sahip x vektörünü, iki elemandan oluşan ve düşük boyutlu değerlere sahip olan m vektörü ile ilişkilendirdik. Bu işleme *"giriş parametreli model tabanlı ağlar"* denir ve bu işlem $P = R^{N^2} \longrightarrow R^2$ olarak gösterilir.

$$m = P(x) \tag{5.18}$$

Güncel kenar belirleme ağlarının yapısı ile alt kümeli hiyerarşik yapılı ağların matematiksel formülü arasında az bir fark vardır. Orijinal ağlarda; alt ağ çıkışı $\phi(x, w_r)$ denklem (5.19)'da gösterildiği gibi doğrudan doğruya kazanan hücrenin çıkışına eşittir.

$$\phi(x, w_r) = {}_r(x, w_{s^*}) \tag{5.19}$$

s^* kazanan hücrenin indeksidir. ${}_r(x, w_{s^*})$ ise kazanan hücrenin çıkışıdır.

Güncel ağ uygulamalarında, diğer taraftan, alt ağ seviyesindeki yarışma işlemi, hücresel seviyedeki işlemlerden bağımsızdır ve çok farklı bir doğaya sahiptir: Alt ağlar arasındaki yarışma işlemi, yerel parlaklık değeri ile giriş örneğinin arka plan parlaklık değeri arasındaki uygunluğu belirler. Sayısal değerler arasında bir karşılaştırma söz konusudur. Hücre seviyesindeki yarışma ise, ancak alt ağın genel parlaklık değeri olan p_r altındaki iki boyutlu vektörlerin karşılaşmalarını içerir. Sonuç olarak alt ağ çıkışı ve hücre çıkışı, iki hiyerarşik seviyede bir birinden bağımsız olarak belirlenir. Alt ağın çıkışı denklem (5.20)'de gösterildiği gibi hesaplanır.

$$\phi(\overline{m}, p_r) = \left|\overline{m} - p_r\right| \tag{5.20}$$

Alt ağ içerisindeki her bir hücrenin çıkışı denklem (5.21)'de gösterildiği gibi hesaplanır.

$${}_r(m, w_{rs}) = \left\|m - w_{rs}\right\| \tag{5.21}$$

5.4 Eğitme Aşaması

Eğitme aşaması üç alt aşamadan oluşur. İlk alt aşama; her bir U_r alt ağın çıkışı olan p_r değerinin yarışmacı öğrenme tekniği ile belirlenmesidir. İkinci aşamada; her bir giriş penceresi olan W'nun p_r ve m değerlerini kullanarak bir alt ağa ve bu alt ağ içerisinde bir hücreye bağlanılır. Kazanan lokal hücrenin w_{rs} hücresi, yarışmacı öğrenme yöntemi ile güncellenir. Üçüncü aşamada ise; elde edilen w_{rs} hücreleri içerisinden b ikili kenar örnekleri çıkartılır. Eğer pencere boyutu $N = 3$ seçilirse, elde edilen b kenar bilgisinin değişik versiyonlarını bulmak için sekiz defa $R_{\pi/4}$ işlemi uygulanır. Ve bu örnekler hafızaya kaydedilir.

5.4.1 U_{r*} alt ağındaki p_r değerinin belirlenmesi

NxN boyutundaki güncel W giriş penceresinin, U_r alt ağındaki p_r değeriyle denklem (5.13) ve (5.14)'deki formüller kullanılarak ilişkilendirildiğini farz edelim. Buna göre, parlaklık değeri denklem (5.22) kullanılarak, öğrenme oranı ise denklem (5.23) kullanılarak güncellenir. t_f toplam iterasyon sayısıdır.

$$p_{r*}(t+1) = p_{r*}(t) + \eta(t)(\overline{m} - p_{r*}(t)) \tag{5.22}$$

$$\eta(t+1) = \eta(0)\left(1 - \frac{t}{t_f}\right) \tag{5.23}$$

5.4.2 N_{r*s*} hücresindeki W_{r*s*} değerinin belirlenmesi

m vektörüne sahip W giriş penceresinin U_r alt ağı içerisindeki N_{r*s*} hücresi ile ilişkilendirildiğini farz edelim. Buna göre, W_{r*s*} ağırlık vektörü denklem (5.24)'de gösterildiği gibi güncellenir.

$$w_{r^*s^*}(t+1) = w_{r^*s^*}(t) + \eta(t)(m - w_{r^*s^*}(t)) \tag{5.24}$$

5.4.3 Geçerli kenar bilgilerinin yakalanması

Önceki aşamaların hepsi tamamlandıktan sonra, alt ağlara ait arka plan parlaklık değerini ifade eden p_r değeri ile ağırlık vektörü w_{rs} değeri elde edilir. Sonuç olarak, *NxN* boyutlu tüm *W* giriş pencereleri, *m* vektörü ve $\overline{m}$ değeri kullanılarak bir alt ağa ve bu alt ağ içerisinde bir hücreye bağlanmıştır. Daha sonra gri seviye değerine sahip *x* giriş vektörü, $w_{r^*s^*}$ ağırlık vektörü üzerinden geçirerek, denklem (25)'de gösterilen ikili seviyeli *b* vektörü elde edilmiştir.

$$b = Q(x, w_{r^*s^*}) \tag{5.25}$$

Elde edilen iki seviyeli *b* vektörü, geçerli kenar bilgisini içeren *C* kümesine kaydedilir. Geçerli kenar vektörü olan *b* çerçevesine, *N=3* olduğu kabul edilirse, denklem (5.26) ve (5.27)'de gösterildiği gibi b_j=*0,1,...,7* için $R_{\pi/4}$ işlemi uygulanır. Elde edilen tüm değişik kenar tipleri *C* kümesine kaydedilir.

$$b_0 = b \tag{5.26}$$

$$b_{j+1} = R_{\frac{\pi}{4}}(b_j) \qquad j = 0,...,6 \tag{5.27}$$

5.5 Tanıma Aşaması

Bu aşamada, test görüntüsündeki tüm noktaların kenar özelliği taşıyıp taşımadığı belirlenir. Tanımlama aşaması iki alt aşamadan oluşur. İlk aşamada, test görüntüsündeki noktalar içerisinden, bilinen kenar örneklerine eşit olanlar seçilir ve *"birinci kenar noktası"* olarak tanımlanır. İkinci aşamada, bir önceki aşamada tespit edilen birincil kenar örneklerini başlangıç noktası olarak alıp kenar izleme işlemi gerçekleştirilir.

Birincil kenar örneği kadar önemli olmayan bu kenar noktalarına ise *"ikincil kenar noktası"* denir. Kenar izleme işlemi tekrarlamalı (recursive) olarak tanımlanır.

5.5.1 Birincil kenar noktalarının belirlenmesi

Bu alt aşamada, test görüntüsündeki tüm *NxN* pencereler incelenir. Her bir giriş örneği $\overline{m}, m_1, m_2$ parametreleri ile ilişkilendirilmiştir. Buna göre, aşağıdaki şartları sağlayan tüm giriş örnekleri *"birinci kenar noktası"* olarak işaretlenir.

(A1). Denklem (13) ve (14) ifadelerini sağlayan tüm x giriş vektörleri bir alt ağa yönlendirilir.

(A2). $w^l{}_r$, U_r alt ağının zayıf ağırlık vektörü olmak üzere, $m_2 - m_1 \geq w^l{}_{r^*,2} - w^l{}_{r^*,1}$ olmak zorundadır.

(A3). $w_{r^*s^*}$, seçilen $N_{r^*s^*}$ hücresinin ağırlık vektörü olmak üzere, $b = Q(x, w_{r^*s^*}) \in C$ olmak zorundadır.

(A1) şartı, güncel pencerenin gri seviye değerlerinin ağın hangi seviyesine daha yakın olduğunu gösterir. (A2) şartı, $m_2 - m_1$ farklı ile kenar büyüklüğünü garanti eder. Bu fark zayıf kenara karşılık gelen bileşenler arasındaki farktan daha büyüktür. (A3) şart ise, güncel pencerenin ikili kenar bilgisinin, C kümesindeki geçerli kenar şablonlarından birine eşit olduğunu garanti eder.

5.5.2 İkincil kenar noktalarının belirlenmesi

İkinci aşamada, dinamik izleme hücresi V_d, birincil kenar noktası ile ilişkili olan ikincil kenar noktalarını izlemek için aktif hale getirilir. Ağın gri seviye göstergesi p_d ve hücrenin ağırlık vektörü w_d, daha önce tespit edilen birincil kenar noktasının $\overline{m_p}$ ve m_p değerlerini denklem (5.28) ve (5.29)'da gösterildiği gibi başlangıç değeri olarak kabul edip kenar izleme işlemine başlar.

$$p_d(0) = \overline{m}^p \tag{5.28}$$

$$w_d(0) = m^p \tag{5.29}$$

Dinamik hücrenin başlangıç parametrelerini verdikten sonra, rekürsif bir kenar izleme algoritmasını ile, güncel birincil kenar noktasının her bir 8 komşusuna aşağıdaki şartlar kümesi uygulanarak ikincil kenar noktalarını belirleriz.

(B1) $b = Q(x, w_d) \in C$

(B2) $p_d \in [m_1, m_2]$

(B1) şartı, birincil kenar noktalarını yakalamak için belirtilen (A3) şartına benzer. (B2) şartı ise denklem (5.13)'deki ifadenin değişik bir halidir. Önceki kenar noktalarına benzeyen p_d, ikincil kenar noktalarının gri seviye değerlerini garanti etmek için denklem (5.13)'deki ifadenin değişik bir versiyonunu kullanır.

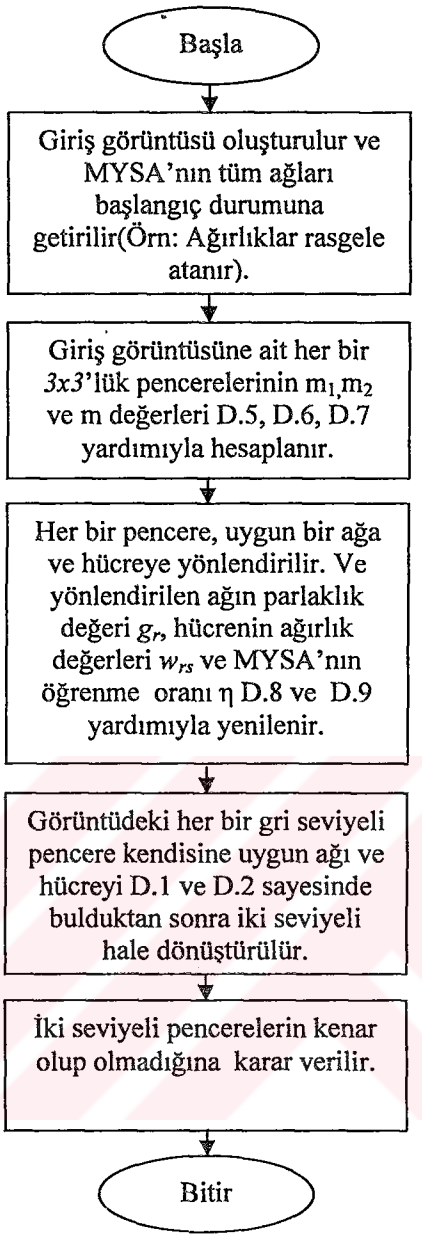
Yukarıdaki şartları sağlayan her bir ikincil kenar noktası için, lokal parlaklık seviyesi olan p_d aşağıdaki gibi güncellenir.

$$p_d(t+1) = p_d(t) + \eta(t)(\overline{m}^s - p_d(t)) \tag{5.30}$$

Ayrıca, bu kenar noktası, birincil kenar noktasını yakalamak için önerilen (A2) şartını sağlarsa, onun kenar büyüklüğü, birincil kenar noktası ile benzerdir. Dinamik hücrenin ağırlık vektörü w_d'de güncel noktanın özellikleri ile birleştirilerek güncellenir.

$$w_d(t+1) = w_d(t) + \eta(t)(\overline{m}^s - w_d(t)) \tag{5.31}$$

Yukarıda teorisinden detaylı bir şekilde bahsedilen MYSA kullanılarak geliştirilmiş kenar yakalama algoritmasının çalışma prensibi Şekil 5.7'de akış şeması biçiminde gösterilebilir.



Şekil 5.7 MYSA kullanarak kenar yakalama algoritmasının akış şeması

6. EDGE_NEURO ve UYGULAMA SONUÇLARININ KARŞILAŞTIRILMASI

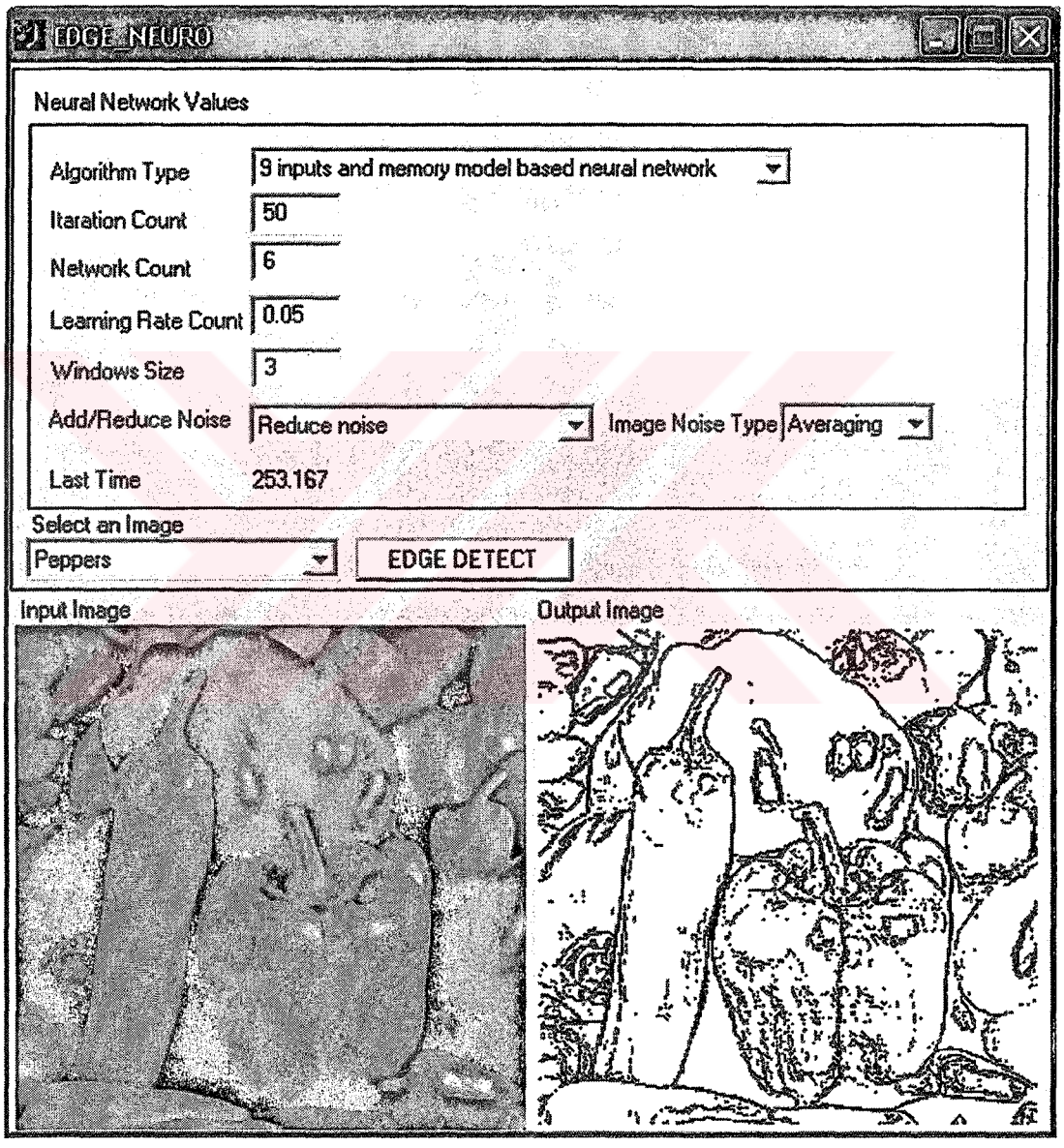
Bu çalışmada, iki seviyeli ve gri seviyeli görüntülerin kenarlarını yakalayabilmek için MATLAB 6.5 Programlama Dili kullanılarak "Binary_Edge" ve "Edge_Neuro" adlı iki tane program yazılmıştır. İki seviyeli görüntüler için geliştirilen Binary_Edge adlı program hakkında Bölüm 3'de gerekli bilgiler verilmiştir. Bu bölümde gri seviyeli görüntüler için hazırlanmış Edge_Neuro programından bahsedilir ve bu program kullanılarak elde edilen kenar sonuçları karşılaştırılır.

Edge_Neuro programı, gri seviyeli görüntüler için MYSA mimarisini kullanarak kenar yakalama problemine makul bir çözüm getirmek amacı ile yazılmıştır. Edge_Neuro programının görüntüsü Şekil 6.1'de gösterilmiştir. Edge_Neuro programı, giriş olarak beş farklı parametre alır. Bunlar sırasıyla giriş görüntüsü, ağ sayısı, mimari tipi, iterasyon sayısı ve güncel pencere boyutudur. Ayrıca Edge_Neuro programı kullanılarak, giriş görüntünün sahip olduğu gürültü miktarı farklı filtreler yardımıyla azaltılabilir veya arttırılabilir.

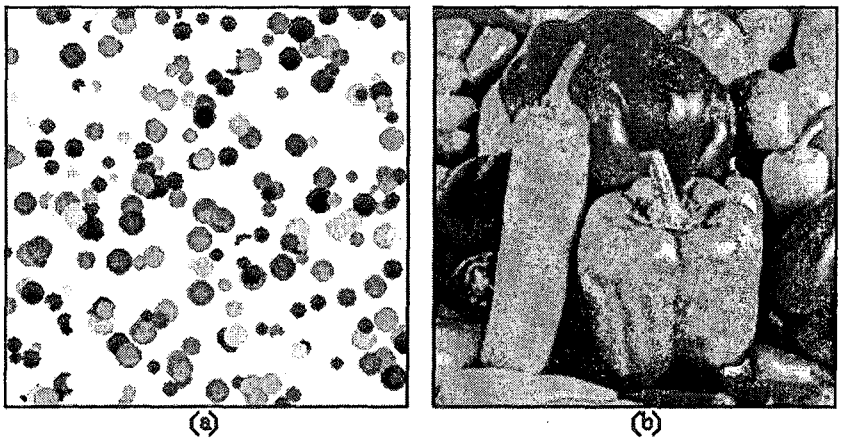
MYSA'nın sonuçları ile geleneksel KYA'ları içerisinde en ideallerden biri olarak bilinen Canny KYA'sının sonuçlarını karşılaştırmak için üç tane gri seviyeli giriş görüntüsü seçilmiştir. Birinci görüntü, Şekil 6.2 (a)'da gösterilmiştir. Bu görüntü farklı yoğunluk değerlerine sahip dairelerden oluşan, 256x256 nokta boyutunda, her bir noktası 8 bit ile ifade edilen ve herhangi bir gürültü içermeyen yapay bir görüntüdür. İkinci görüntü, Şekil 6.2 (b)'de gösterilen biber görüntüsüdür. Üçüncü görüntü ise Şekil 4.2'de gösterilen "Lena" görüntüsüdür.

Edge_Neuro kullanılarak arzu edilen kenar sonucuna ulaşabilmek için R (ağ sayısı), itn (iterasyon sayısı), lr (öğrenme oranı) ve W (pencere boyutu) parametrelerinin iyi seçilmesi gerekir. Bu parametreler arasındaki olası kombinasyonların sayısı oldukça büyüktür. MYSA'nın küçük parlaklık değişimlerine duyarlı olması isteniyorsa, R parametresinin değeri yüksek tutulmalıdır. Bu sayede, görüntüdeki güncel pencereler, ortalama parlaklık değerlerine uygun bir ağ bularak eğitilebilir. Eğer ağ sayısı küçük seçilirse, güncel pencerelerin bir çoğu kendine uygun bir ağ bulamaz. Böylece arzu edilen bir çok kenar noktası yakalanamaz. İtn ve lr parametreleri klasik YSA'lardan hatırlanacağı gibi ağın eğitilmesi için gerekli olan parametrelerdir. Bu parametreler

doğru kenar yeri ve kenar noktasını tespit etmede büyük rol oynar. İtn, hedefe yaklaşma sayısı, lr ise her bir yaklaşma büyüklüğü olarak adlandırılabilir. W parametresi ise MYSA'ya giriş olarak verilen pencere büyüklüğüdür. Giriş penceresinin büyük olması o noktanın kenar olup olmadığı hakkındaki tahminin doğruluğunu arttırır. Fakat MYSA'nın eğitim süresinin artmasına da neden olur. Bu nedenle, W parametresi genellikle 3 (3x3) seçilir.



Şekil 6.1 Edge_Neuro programının görünüşü



Şekil 6.2 Test için hazırlanan (a) Yapay görüntü (b) Biber görüntüsü

Canny kenar yakalama algoritması α filtre katsayısı, t_1 ve t_2 eşik parametreleri olmak üzere üç tane parametreye ihtiyaç duyar. Bu parametreler arasındaki olası kombinasyonların sayısı oldukça büyüktür. Bu kombinasyonlar arasından orijinal görüntünün kenar sonucuna en yakın olan değerleri alınmıştır. Canny algoritmasında genellikle, t_1 ve t_2 değerleri düşük seçildiğinde elde edilen kenar sonucu daha az detay içerir. Aynı zamanda arzu edilen kenarlarda büyük bir azalma olur. Diğer durumda, eşik değerleri yüksek tutulduğunda istenmeyen kenarlar elde edilir.

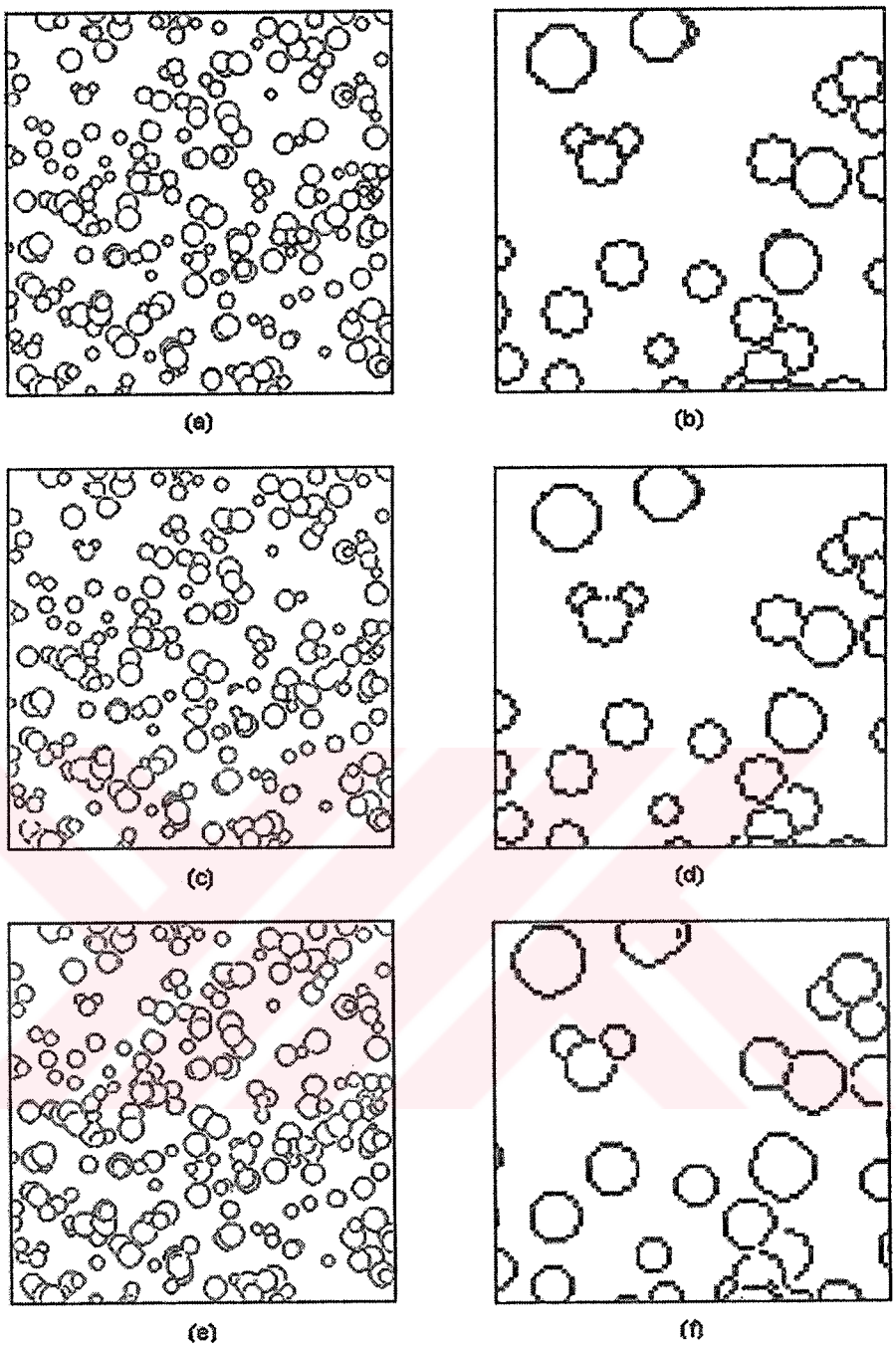
Şekil 6.3 (a), Şekil 6.2 (a)'daki yapay görüntünün arzu edilen ideal kenar görüntüsünü gösterir. Kullanılan MYSA algoritması için $R = 2$, $lr = 0.2$ ve itn = 25 değerleri seçilmesi durumunda elde edilen kenar sonucu Şekil 6.3 (c)'de gösterilmiştir. Canny algoritması için $t_1 = 20$, $t_2 = 25$ ve $\alpha = 0.3$ seçildiğinde elde edilen kenar görüntüsü, arzu edilen kenar görüntüsüne en yakın kenar sonucunu üretmiştir. Bu değerler kullanılarak Canny algoritmasından elde edilen kenar sonucu Şekil 6.3 (e)'de gösterilmiştir.

Elde edilen sonuçların daha iyi analiz edilebilmesi için Şekil 6.3 (b)'de arzu edilen kenarın, Şekil 6.3 (d)'de MYSA'nın ve Şekil 6.3 (f)'de ise Canny kenar yakalama algoritmasının ($\alpha = 0.3$, $t_1 = 20$, $t_2 = 25$) sonuçlarının sadece belli bir parçasının büyütülmüş şekli gösterilir. Buna göre, Şekil 6.3 (d)'deki kenar sonucu gösterir ki; MYSA, zemin üzerindeki her hangi bir dairenin kenar noktalarını hatasız bir şekilde tespit edebilirken, birden fazla dairenin üst üste gelmesiyle oluşan kesişme noktalarındaki kenar noktalarını tam olarak tespit edemez. Bu noktaların tespit edilmesi

MYSA'nın eğitim için kullandığı iterasyon sayısı ile doğrudan alakalıdır. İterasyon sayısı arttıkça kesişme noktalarındaki çıkarılamayan nokta sayısı azalacaktır.

Şekil 6.3 (f)'deki kenar sonucu, bizlere Canny algoritmasının sonucunun MYSA'nın sonucu kadar hassas olmadığını gösterir. Hazırlanan yapay görüntüde, Canny'nin kenar noktalarını tespit edebildiği, fakat kenar noktasının yeri hakkındaki kararı doğru veremediği görülür. Daha öncede bahsedildiği gibi, Canny algoritmasının kenar yerine karar verme ve kenarı tespit edebilme özellikleri arasında bir ters orantı vardır. Bu özelliklerden herhangi birisinin hassasiyeti arttırılınca diğerinin hassasiyeti azalır. Böylece, MYSA'nın kenar sonucunun arzu edilen kenar görüntüsüne daha yakın olduğu görülür.

MYSA'nın, doğal bir görüntü olan, bir çok detay bilgisi ve gürültü içeren, gri seviyeli ve 256x256 nokta boyutundaki biber görüntüsüne verdiği kenar yanıtı Şekil 6.4 (a)'da, Lena görüntüsüne verdiği yanıt 6.5 (a)'da gösterilmiştir. Canny algoritmasının, biber görüntüsüne verdiği kenar yanıtı Şekil 6.4 (b)'de, Lena görüntüsüne verdiği yanıt 6.5 (b)'de gösterilmiştir. Buna göre, doğal görüntülere MYSA'nın verdiği kenar sonucu, Canny'nin kenar sonucuna göre daha güçlü olmasına rağmen daha fazla istenmeyen kenar içermektedir. MYSA'nın verdiği kenar sonucu, kenar yakalama algoritmalarından beklenen tek nokta kalınlığındaki ideal kenar bilgisi değildir. Oysa Canny algoritması tek nokta kalınlığında bir kenar sonucu üretmiştir. Bunun nedeni, Canny algoritması kenar çıkarma işlemine başlamadan önce görüntüdeki gürültüyü azaltmak için α katsayılı bir filtreden geçirmesi ve iki eşik değeri kullanmasıdır.



Şekil 6.3 (a) Arzu edilen kenar görüntüsü (b) Arzu edilen kenar görüntüsünün belli bir kısmının büyütmüş hali (c) Orijinal görüntü için kullanılan MYSA'nın kenar sonucu (d) MYSA sonucunun görüntüsünün belli bir kısmının büyütmüş hali (e) Farklı t_1=20 ve t_2=25 eşik değerleri için Canny kenar çıkarma algoritması kullanılarak elde edilen kenar sonucu (filtre parametresi α=1) (f) Canny kenar çıkarma algoritmasının sonucunun belli bir kısmının büyütmüş hali (α=1, t_1=20, t_2=25).

Şekil 6.4 Biber görüntüsü için (a) MYSA'nın kenar sonucu (its=100, R=6, lr=0.01) (b) Canny algoritmasının kenar sonucu (α=0.5, t_1=5, t_2=20).

Şekil 6.5 Lena görüntüsü için (a) MYSA'nın kenar sonucu (its=100, R=6, lr=0.01) (b) Canny algoritmasının kenar sonucu (α=0.5, t_1=5, t_2=20).

7. SONUÇLAR

Bu çalışmada, iki seviyeli ve gri seviyeli görüntülerin kenarlarını yakalayabilmek için MATLAB 6.5 Programlama Dili kullanılarak "Binary_Edge" ve "Edge_Neuro" adlı iki tane program yazılmıştır.

Binary_Edge programı kullanılarak, iki seviyeli yapay ve doğal görüntülerde arzu edilen kenar görüntüsü daha kısa zamanda ve daha az karşılaştırma yapılarak elde edilmiştir. Binary_Edge programının, iki seviyeli görüntülerde hız kriterine yaptığı bu iyileştirme, nesne tanıma işlemlerinde hayati bir önem taşımaktadır.

Edge_Neuro programı kullanılarak, elde edilen yapay kenar görüntüleri incelendikten sonra, MYSA mimarisi kullanılarak hazırlanan kenar yakalayıcıların geleneksel metotlardan çok daha doyurucu ve ümit verici yanıtlar verdiği sonucuna varılmıştır. Ayrıca benimsenen mimari sayesinde kenar elementleri arasında kabul edilebilir ilişkilerde ve önemli kenar yapılarının olduğu noktalarda daha güçlü sonuçlar elde edilmiştir. Fakat yüksek miktarda gürültü içeren doğal görüntülerde aynı başarı yüzdesine ulaşılamamıştır. Bunun sebebi, MYSA'na giriş olarak verilen görüntülerin gürültü miktarını azaltacak veya ortadan kaldıracak bir filtrenin kullanılmamasıdır. Bu noktada filtre seçimi büyük önem arz etmektedir. Bu nedenle, yüksek gürültü miktarına sahip bu tür görüntülerde, MYSA kullanılarak elde edilen kenar sonucu tek nokta kalınlığına sahip değildir. Buda kenar yakalama algoritmalarında büyük bir dezavantajdır.

Bundan sonra yapılacak çalışmalar için, kenar yakalama özelliğinin yanı sıra filtreleme işlemini de gerçekleştirebilen, MYSA tabanlı bir algoritma tasarlanması ve uygulamasının yazılması düşünülebilir. Bu sayede, sadece yapay görüntüler için değil gerçek hayatta kullanılan doğal görüntüler için de arzu edilen kenar sonuçlarına ulaşılabilir.

MYSA'nın, geleneksel yapay sinir ağı metotlarından çok daha yüksek sınıflandırma yeteneğine sahip olmasından dolayı, bu ve benzer alanlarda yapılacak olan çalışmaların daha fazla ilgi göreceğine inanıyoruz.

KAYNAKLAR

[1] Mayo Clinic Ansiklopedisi, Cilt 2, s. 436.

[2] Arthur C. Guyton, 1986. Tıbbi Fizyoloji, 7.b., Merk Publishing, s. 1012.

[3] Russ, J.C., 2002. The Image Processing Handbook CRC Press LLC North Carolina State University.

[4] Gonzalez, R., and Wintz, P., 1987. Digital Image Processing. Reading, MA: Addison Wesley, second ed.

[5] Jahne, B., 1997. Digital Image Processıng Concept,Algorithms, and Scientific Applications.

[6] Ioannis, P., 1993. Digital Image Processing Algorithms, Prentice hall international series in signal processing.

[7] Ritter, G. X., Wilson, J. N., 2001. "Handbook of Computer Vision Algorithms in Image Algebra", Boca Raton London New York Washington, D.C.

[8] Türkoğlu, İ. and Arslan, 1999. An Edge Dection Method Developed For Object Recognition, Eleco'99 International Conference On Electrical And Electronics Engineering, 428-430, Bursa.

[9] Bağcıoğulları, F., ve C.H., 1994. Nesne tanıma sistemi için geliştirilmiş özel bir kenar çıkarma algoritması, 2. Signal İşleme ve Uygulamaları Sempozyumu Bildiriler Kitabı, s.47-52.

[10] Türkoğlu, İ. and Arslan, 1999. An Edge Tracing Method Developed For Object Recognition, The 7-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media'99, 9-12, Plzen, Slovakya.

[11] Pinho, J., and Almeida, L.M., 1994. Some Results On Edge Enhancement Wıth Neural Networks. Proc. of the 1st IEEE Int. Conf. on Image Processing, ICIP'94, Austin, TX, vol. III, pp. 893-897.

[12] Perry, S., and Guan, L., 2000. Weight assignment for adaptive image restoration by neural Networks. IEEE Transactions on Neural Networks, vol. 11, pp. 156–170.

[13] Shen, L., and Rangayyan, M. M., 1997. A segmentation-based lossless image coding method for high-resolution medical image compression. IEEE Trans. Medical Imaging, vol. 16, no. 3, pp. 301–307.

[14] A. Fitzgibbon, the Department of Artificial Intelligence, University of Edinburgh.

[15] Costa, L. F., and Cesar, R. M., 2001. Shape Analysis and Classification. pp. 240-244.

[16] Prewitt, J.M.S., 1970. Object Enhancement and Extraction", in Picture Processing and Psychopictorics, Lipkin, B.S. and Rosenfeld, A., eds., Academic Pres, New York.

[17] Sobel, 1970. Camera models and machine perception AIM-21. Stanford Artificial Intelligence Lab, Palo Alto, CA.

[18] Berzins, V., 1984, Accuracy of Laplacian edge detector. Computer Vision, Graphics and Image Processing, pp, 27:195-210.

[19] Canny, J.F., 1986. A Computational Approach to Edge Detection, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol.PAMI-8, no.6, pp. 679-698.

[20] Caelli, T., Squire, D., Wild, T., 1992. Model-Based Neural Network, vol. 6, pp. 613-625.

[21] Wong, H., Caelli, T., and Guan, L., 2000. A Model-Based Neural Network for Edge Characterization. Pattern Recognition, 33, 3, pp. 427-444.

[22] Pratt, W. K., 1991. Digital Image Processing, ISBN 0-471-85766-1, pp. 490-497

[23] Pitas, A., 2000. Digital Image Processing Algorithms and Applications, John Wiley&Sons, Inc., New York, Aristotale University of Thessalariki.

[24] Türkoğlu, İ. and Arslan, A, 1996. Yapay Sinir Ağları İle Nesne Tanıma, Yüksek Lisans Tezi, Fırat Üniversitesi Fen Bilimleri Enstitüsü, Elazığ.

[25] Duda, R.O., and Hart, P.E., 1989. Pattern Classification and Scene Analysis. Stanford Research Institute.

[26] Uzunalioğlu, H., 1992. Model-Based Recognition of Polyhedral Objects, Yüksek Lisans Tezi, ODTÜ Fen Bilimleri Enstitüsü, Ankara.

[27] Banks, S. 1990. Signal Processing, Image Processing and Pattern Recognition. Sheffield University.

[28] Caelli, T., Squire, D., Wild, T., 1992. Model-Based Neural Network, vol. 6, pp. 613-625.

[29] Kang, C. W., 1991. Extraction of Straight Line Segments Using Rotation Transform, Generalized Hough Transform, Pattern Recognition, vol. 24 (7), pp. 633.

[30] Perry, S. W., Wong, H., Guan, L., 2002. Adaptive Image Processing, A Computational Intelligence Perspective, Boca Raton London New York Washington, D.C.

[31] Shi, Y. Q., Sun, H., 2000, Image And Video Compression For Multimedia Engineering, Boca Raton London New York Washington, D.C.

[32] M-T. E., 2000. Supervised And Unsupervised Pattern Recognition, Rutgers University, N.W., Boca Raton, Florida 33431.

[33] Irwin, J. D., 2000. Supervised And Unsupervised Pattern Recognition. CRC Press LLC, Auburn University.

[34] Wong, H., Caelli, T., and Guan, L., 2000. A Model-Based Neural Network for Edge Characterization. Pattern Recognition, 33, 3, pp. 427-444.

ÖZ GEÇMİŞ

Fatih TALU

Fırat Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği
23119, Elazığ

Tel: 424-2370000-5015
E.posta: fatihtalu@firat.edu.tr

1979	Elazığ'da doğdu.
1997-2001	Fırat Üniversitesi M.F. Bilgisayar Mühendisliği Bölümünden mezun oldu.
2001-	Fırat Üniversitesi Rektörlüğü Enformatik Bölümünde Öğretim Görevlisi olarak çalışmaktadır.