İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRONİK BİR HÜCRESEL YAPAY SİNİR AĞI GERÇEKLEMESİ OLAN ACE16K ÜZERİNDE GÖRÜNTÜ BÖLÜTLEME

YÜKSEK LİSANS TEZİ

Müh. Recep USLU

Anabilim Dalı: ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ

Programı: ELEKTRONİK MÜHENDİSLİĞİ

HAZİRAN 2007

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRONİK BİR HÜCRESEL YAPAY SİNİR AĞI GERÇEKLEMESİ OLAN ACE16K ÜZERİNDE GÖRÜNTÜ BÖLÜTLEME

YÜKSEK LİSANS TEZİ

Müh. Recep USLU

Tezin Enstitüye Verildiği Tarih: 7 Mayıs 2007

Tezin Savunulduğu Tarih: 6 Haziran 2007

Tez Danışmanı: **Yrd. Doç. Dr. Müştak Erhan YALÇIN**

Diğer Jüri Üyeleri **Yrd. Doç. Dr. Neslihan Serap ŞENGÖR (İ.T.Ü.)**

Prof. Dr. Tülay YILDIRIM (Y.T.Ü.)

HAZİRAN 2007

ÖNSÖZ

Bu tez çalışması sırasında başta yardımlarını esirgemeyen tez danışmanım Yrd. Doç. Dr. M. Erhan YALÇIN olmak üzere, laboratuar imkanlarını kullanmama izin veren İstanbul Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölüm Başkanı Doç. Dr. Sabri ARIK' a ve yine aynı üniversiteden Araştırma Görevlisi Fethullah KARABİBER' e, maddi desteğinden dolayı TÜBİTAK Bilim İnsanı Destekleme Başkanlığı'na ve her zaman yanımda olan aileme teşekkürlerimi sunarım.

Haziran 2007 **Recep USLU**

İÇİNDEKİLER

KISALTMALAR

OCR	: Optical Character Recognation
RAG	: Region Adjacency Graph
HYSA	: Hücresel Yapay Sinir Ağları
CNN	: Cellular Nonlinear Networks
CNN-UM	: CNN Universal Machine
LAM	: Local Analog Memory
LLM	: Local Logical Memory
LAOU	: Local Analog Output Unit
LLU	: Local Logic Unit
LCCU	: Local Communication and Control Unit
GAPU	: Global Analogic Programming Unit
APR	: Analog Program Register
DSP	:Digital Signal Processor
AMC	:Analogic Macro Code
MRF	:Markov Random Field
DMF	:Dense Motion Field
ES	:Early Segmentation

TABLO LİSTESİ

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

ELEKTRONİK BİR HÜCRESEL YAPAY SİNİR AĞI GERÇEKLEMESİ OLAN ACE16K ÜZERİNDE GÖRÜNTÜ BÖLÜTLEME

ÖZET

Bir görüntüyü anlamlı ve türdeş bölümlerden oluşan kümelere parçalamayı içeren görüntü bölütleme sayısal görüntü işlemenin en önemli konularından biridir. Bu durumda parçalanmış her bölümdeki pikseller; piksel grilik seviyesi, piksel RGB rengi, pikselin kameradaki sırası, pikselin yeri, bölgesel eş-değişirlik matrisi, gri seviyeler, kontrast, spektral değerler gibi nitelikleri veya özellikleri aynı olan türdeş bir kümeye sahiptirler. Literatürde birçok görüntü bölütleme metodu mevcuttur. Ancak bunların hiçbiri tüm görüntülere uygulanabilir nitelikte değildir. Ayrıca görüntü bölütleme işlemi en hızlı bilgisayarlar yardımıyla dahi oldukça zaman alan bir işlemdir. Bu yüzden analog ve paralel yapılar bu işlem için oldukça etkili olacaktır. Hücresel sinir ağlarının elektronik bir gerçeklemesi olan ACE16K kırmığı görüntü bölütleme gibi birçok sayısal görüntü işleme işlemini çok kısa sürede gerçekleştirmektedir.

Tezin ilk bölümünde mevcut olan görüntü bölütleme metotları hakkında detaylı bir bilgi verilmiş ve bu metotların sınırlamaları belirtilmiştir. İkinci bölümde hücresel yapay sinir ağları hakkında detaylı bilgi verilmiş ve elektronik bir gerçeklemesi olan ACE16K dan bahsedilmiştir. Üçüncü bölümde bir görüntü bölütleme algoritması olan Early bölütleme algoritmasından detaylı olarak bahsedilmiş ve adımları sırasındaki giriş ve çıkış görüntüleri verilmiştir. Ayrıca yeni bir algoritma ACE16K üzerinde gerçeklenmiş ve sonuçları gösterilmiştir. Gerçeklenen algoritmaların sistem üzerinde işlem süreleri hesaplanmış ve algoritmaların gerçek zamanlı bölütlemeye izin verdiği gösterilmiştir. Son bölümde ACE16K üzerinde elde edilen sonuçlar yorumlanmış ve mevcut algoritmalar karşılaştırılarak daha iyi sonuçlar elde edebilmek için bazı önerilerde bulunulmuştur.

IMAGE SEGMENTATION ON ACE16K, AN ELECTRONIC IMPLEMENTATION OF CELLULAR NEURAL NETWORKS

SUMMARY

Image segmentation which involves partitioning an image into a set of homogeneous and meaningful regions is one of the most important task of digital image processing. The pixels in each partitioned region posses an identical set of properties like pixel gray level, pixel RGB color, range of the pixel from the camera, position of the pixel, local covariance matrix, gray levels, contrast, spectral values. There are several image segmentation methodologies in literature. But none of these methodologies can be applied all kinds of images. Moreover, image segmentation is a so much time remaining application even with the help of powerful computers. Therefore, analog and parallel architectures will be helpful for image segmentation applications. ACE16K which is an electronic implementation of cellular neural networks can perform the most of the image processing tasks.

In chapter one, the overview of image segmentation methods in the literature is presented and the restrictions of these methods are represented. In chapter two, detailed information about cellular nonlinear networks is given and an electronic implementation of CNN, ACE16K is mentioned. In chapter three, an image segmentation algorithm, which is early algorithm, is explained and the result of each step are given. Moreover a new segmentation algorithm is realized on ACE16K and the results of this algorithm are given. The process times of these algorithms on ACE16K are computed and it is shown that these algorithms allow to real time segmentation. In chapter four, some concluding remarks are stated on the performance of image segmentation algorithms and some suggestions are given for better performance.

1. GİRİŞ

Sayısal görüntü işlemenin önemli bir konusu, bir görüntünün türdeş bölümlere bölütlenmesidir [1]. Görüntü bölütleme, sayısal görüntü işlemenin en zor problemlerinden biridir ve literatürde birçok farklı yaklaşım ve metotlar önerilmiştir. Ancak halen bütün görüntü tiplerine uygulanabilen ve mükemmel sonuçlar sağlayan kesin bir çözüm mevcut değildir.

Görüntü bölütleme, sayısal görüntü işlemenin görsel-yönlendirimli otonom robotik, ürün kalite denetimi, tıbbi teşhis, uzaktan algılanan resimler gibi alanlarda uygulamaları olan önemli bir araştırma konusudur. Görüntü bölütlemenin amacı, bir görüntüyü, görüntüden çıkarılan belirleyici niteliklere göre türdeş bölgelerine ayırmak olarak tanımlanabilir. Görüntü bölütleme metotları dört kategori içinde sınıflandırılabilir:

1) Kümeleme metotları

2) Bölge-tabanlı metotlar

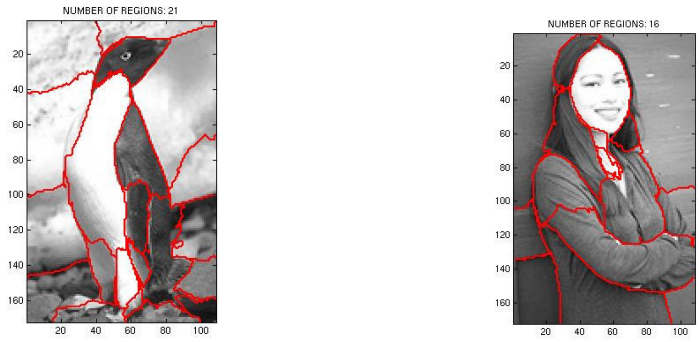
3) Melez metotlar

4) Bayesgil metotlar.

1.1. GÖRÜNTÜ BÖLÜTLEME PROBLEMİ:

Bölütleme, bir görüntüyü anlamlı ve türdeş bölümlerden oluşan kümelere parçalamayı içerir, bu durumda parçalanmış her bölümdeki pikseller, nitelikleri veya özellikleri aynı olan türdeş bir kümeye sahiptirler. Görüntü özelliklerinin bu kümeleri; piksel grilik seviyesi, piksel RGB rengi, pikselin kameradaki sırası, pikselin yeri, bölgesel eş-değişirlik matrisi, gri seviyeler, zıtlık, spektral değerler veya niteliksel özellikleri içerebilir. Bölütlemenin sonucu, her biri tek etikete sahip olan türdeş bölümlerin bir sayısıdır. Böylece bir görüntü, birbirine bağlı ve üst üste binmeyen bölümlerin bir kümesi olarak tanımlanır, bu durumda görüntüdeki her

piksel ait olduğu bölümü gösteren tek bir bölüm etiketine sahip olur. Görüntü bölütlemenin sonuçları genellikle daha yüksek seviyeli sayısal görüntü işleme işlemlerinin ilk parametreleri olarak kullanılırlar.



Şekil 1.1: Bölütlenmiş görüntü örnekleri

Bir görüntünün tam bir bölütlenmesi R , bir sonlu bölümler kümesinin (R_1, R_2, R_3,……., R_N) tanımlanmasını içerir, öyle ki;

1) $R = R_1 \cup R_2 \cup ..... \cup R_N$

2) $R_i \cap R_j = \phi$, $\forall i \neq j$

3) $P(R_i) = Dogru$, $\forall i$

4) $P(R_i \cup R_j) = Yanlis$, $\forall i \neq j$.

Burada bir miktar bölüm mevcut olabilir. Bu bölümlerin uygun bir kümesinin seçimi, bölümdeki P ilişkisinin özelliğinin seçimine bağlıdır.

İlave olarak, bir bölümdeki pikseller arasındaki bağlanabilirliğin dikkate alınması gerekmektedir. Bölütleme algoritmaları pikseller arasındaki gri seviye değerlerinin iki temel özelliği olan süreksizlik ve benzerliğe dayanır.

Birinci kategorideki algoritmalarda, gri seviyelerdeki ani değişmelere dayanan bir görüntü parçalanır. Bu kategorideki ilişkinin başlıca alanları bir görüntüdeki kenarların ve çizgilerin tespit edilmesidir. Bu durumda, eğer bir görüntüdeki kenarları çıkarabilir ve bunları bağlayabilirsek, bölüm içerdiği kenar hatları tarafından tanımlanır [1].

Bölütleme işlemine başka bir perspektiften de bakabiliriz. Bu bakış noktasında, daha fazla ya da daha az aynı türdeş yoğunluğa sahip piksellerin birleşik kümesi bölümleri biçimlendirir. Bu durumda, bölümlerin içindeki pikseller bölümü tanımlar ve bölütleme işlemi tüm görüntüyü sonlu bir sayıda bölüme parçalamayı içerir.

İkinci kategorideki başlıca yaklaşımlar, bir bölümdeki pikseller arasındaki benzerliğe dayanır. Bir görüntüyü bölütlerken, piksellerin çeşitli yerel özellikleri kullanılır. Saptanan iyi bölütleme teknikleri:

- Histogram tabanlı eşikleme
- Bölüm büyütme
- Bölüm ayırma ve birleştirme
- Kümeleme/Sınıflandırma
- Graf teorik yaklaşımı
- Kural tabanlı veya bilgi güdümlü yaklaşım

1.2. GÖRÜNTÜ BÖLÜTLEME METOTLARI:

1.2.1. KENAR TABANLI BÖLÜTLEME:

Kenar tabanlı bölütlemeye geçmeden önce kenar, çizgi, nokta gibi kavramlardan ve nasıl bulunduklarından bahsedelim.

1.2.1.1. Kenar, Çizgi ve Nokta Tespiti:

Kenarlar, çizgiler ve noktalar görüntüdeki çeşitli bölgeler hakkında birçok bilgi taşırlar. Bu özellikler, yerel özelliklerden tek başına elde edildikleri için genellikle yerel özellikler olarak adlandırılır. Kenar ve çizgilerin her ikisi birden gri seviyedeki ani değişimden elde edilmesine rağmen halen bu ikisi arasında önemli bir fark vardır. Bir kenar aslında birbirinden açıkça farklı iki bölüm arasında sınır çizer, bu bir kenarın iki farklı bölüm arasında sınır olduğu anlamına gelir. Diğer taraftan, bir çizgi, düzenli türdeş bir bölümün içine yerleştirilebilir. Örneğin, ince bir çizgi, aynı bitkileri bulunduran iki tarım arazisi arasında kullanılabilir. Bir nokta düzenli türdeş

bir bölümün içine yerleştirilir ve gri değeri yerleştirildiği bölümdeki ortalama gri değerinden farklıdır. Bu bir sivri uca benzer.

Kenar tespit işlemi temelde bir görüntüdeki yoğunluk seviyelerindeki önemli yerel değişimlerin tespit edilmesi işlemidir. Yoğunluk seviyesindeki değişim görüntünün gradyanı ile ölçülür. İki boyutlu bir görüntü *f(x,y)* ise bu görüntünün gradyanı aşağıdaki gibi bir vektör olacaktır;

$$\begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{df}{dx} \\ \frac{df}{dy} \end{bmatrix}$$

Gradyanın büyüklüğü değişik birkaç yolla hesaplanabilir.

$$G|f(x,y)| = \sqrt{G_x^2 + G_y^2}$$

$$G|f(x,y)| = |G_x| + |G_y| \qquad \textbf{(1.1)}$$

$$G|f(x,y)| = \max\{|G_x|, |G_y|\}$$

Gradyanın doğrultusu aşağıda görüldüğü gibidir;

$$\theta(x,y) = \tan^{-1}(G_y / G_x) \qquad \textbf{(1.2)}$$

Burada θ açısı *x* eksenine göre ölçülür.

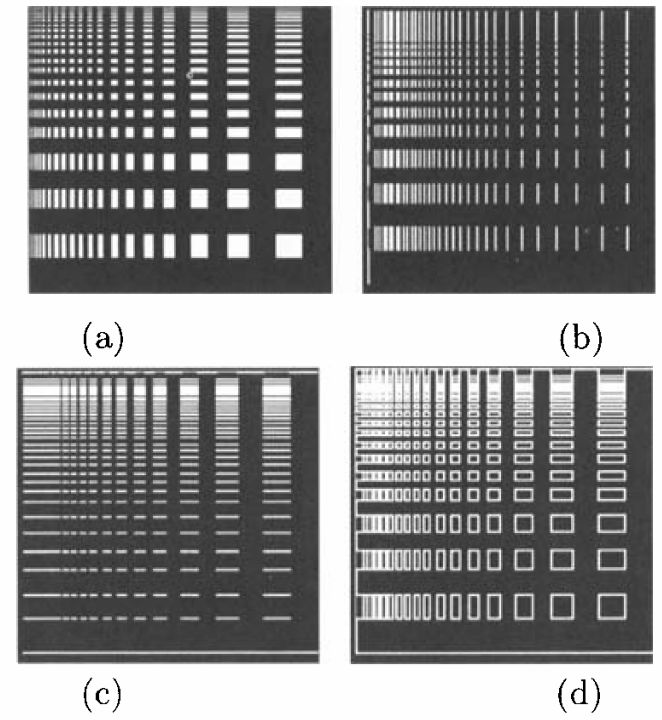
Gradyan operatörü gri seviye yoğunluklarındaki değişimleri ve ayrıca değişimlerle birlikte meydana gelen doğrultuyu hesaplar. Bu, komşu piksellerin değerlerinin farkıyla hesaplanır. Örneğin *x* ve *y* ekseni boyunca olan değişimler. İki boyutlu bir görüntüde gradyanlar yaklaşık olarak aşağıdaki gibidir;

$$G_x = f(i+1, j) - f(i, j)$$

$$G_y = f(i, j+1) - f(i, j)$$

Gradyan operatörleri, biri *x* doğrultusundaki diğeri *y* doğrultusundaki gradyanı elde etmek için iki maskeye gereksinim duyarlar. Bu iki gradyan, büyüklüğü görüntüdeki

bir noktadaki kenar gradyanının kuvvetini gösteren ve açısı gradyan açısını gösteren bir vektör büyüklüğü elde etmek için birleştirilir. Şekil 1.2 de verilen bir giriş görüntüsünün yatay, dikey ve her iki doğrultudaki gradyan görüntüleri görülmektedir.



Şekil 1.2: (a) Giriş Görüntüsü, (b) Dikey Kenarlar, (c) Yatay Kenarlar, (d) Her iki yöndeki kenar görüntüsü

Kenar gradyanlarının hesaplanmasına alternatif bir yaklaşım, görüntünün sekiz eş-bölümlenmiş doğrultuda (doğu, kuzeydoğu, kuzey, kuzeybatı, batı, güneybatı, güney, güneydoğu) seçilen bir şablon kümesi ile konvolüsyona sokulmasını içerir. Her şablon belirli bir yöndeki kenarların bulunmasını sağlar [1].

İdeal bir kenar bulucunun bir kenar noktasını, görüntüdeki yanlış, var olmayan bir kenar noktasını yanlışlıkla tespit etmeyecek, doğru bir kenar noktası kaçırmayacak hassasiyette kesin olarak tespit etmesi gerekmektedir. Bu iki gereklilik sık sık birbiriyle çelişir. Bir kenar noktasının varlığı hakkındaki karar bir eşik değerine bağlıdır. Bu durumda, eğer gradyanın büyüklüğü bir eşik değerinden büyükse bu noktada bir kenar noktasının var olduğu sonucuna varırız, aksi halde kenar noktası yoktur. Eğer seçilen eşik değeri büyükse, doğru kenar noktalarının tespit edilememe

olasılığı vardır. Benzer şekilde seçilen eşik değeri küçükse, pek çok gürültülü noktanın yanlışlıkla kenar noktası olarak tespit edilmesi mümkündür. İdeal bir kenar bulucunun amacı eşik değerini uygun bir şekilde seçmektir.

Gradyan işlemi görüntüdeki gürültüyü arttırdığından dolayı bu sorunu azaltmanın en iyi yolu gradyan operatörlerinin uygulanmasından önce gürültü içeren yüksek uzaysal frekansları filtrelemektir.

1.2.1.2. Kenar Tabanlı Bölütleme:

Kenar bulucuların bir kısmı araştırmacılar tarafından geliştirilen basit bir değişime dayanmaktadır. Kenar tabanlı metotlar değişik parlaklık veya renk değerlerinin bir bölgeden diğer bölgeye hızlı geçiş yerlerini belirlemeye çalışır. Temel prensip bazı gradyan operatörlerinin görüntüyle konvolüsyona sokularak uygulanmasıdır. Bunların içinden en önemlileri; Robert operatörü, Sobel operatörü, Prewitt operatörü, Canny operatörü, Krisch operatörüdür [1]. Bu operatörler içinden en iyi kenar bulucu Canny operatörüdür.

Bu operatör tabanlı kenar belirleme tekniklerinin her birinde, gradyan büyüklüğünün değerini hesaplarız. Gradyan büyüklüğü belirli bir eşik değerinden daha büyükse, bir kenarın varlığını tespit ederiz.

Kenar tabanlı bölütlemede, bir doğrultudaki aydınlıktan karanlığa geçiş hızındaki mümkün olan en büyük artışın doğrultusunu veren her noktadaki görüntü yoğunluğunun gradyanı hesaplanır. Bu durumda sonuç o noktadaki görüntünün ne kadar hızlı ya da yavaş değiştiğini gösterir [2].

Ne var ki, görüntüde kırık sınırlar mevcut olduğunda, kenar tabanlı yaklaşımlar yanlış sonuçlara neden olabilmektedir. Bu sınırların güvenilir sonuçlar elde edilebilmesi için birleştirilmesi gerekmektedir [3].

1.2.1.3. Kenar Tabanlı Bölütlemenin Sınırlamaları:

Kenar belirleme metotlarının başlıca sınırlamaları:

1-) Düşük kaliteli görüntüleme araçları ile elde edilen birçok görüntüde, bazı sıradan metotlar sahte kenarlar ve boşluklar oluşturur ve bunların uygulanabilirliği bu durumda sınırlıdır.

2-) Kenar görüntüleme teknikleri görüntünün yerel komşuluğunda yer alan bilgiye dayanmaktadır. Kenar görüntüleme tekniklerinin birçoğu, bir görüntüde gömülü olarak bulunan model tabanlı bilgiyi ele almaz.

3-) Birçok durumda kenar bulma stratejileri, görüntüde anlamlı bir şekilde var olabilen daha yüksek dereceden düzenlemeleri göz ardı eder.

4-) Kenar noktaları görüntüden çıkarıldıktan sonra, bu noktalar sınırları belirlemek için birbirine bağlanır. Bu işlem genellikle ilk olarak kenar elemanlarını kenar bölümlerine benzeterek ve ardından bölümleri sınırlara benzeterek yapılır. Kenarları birleştirme işlemi bazen görüntüde süreksizliklerin ve boşlukların oluşmasına yol açar.

5-) Kenar birleştirme metotları, sınırdaki boşlukları kapatmak için sıklıkla keyfi eklentilere başvurur.

6-) Sahte kenarları teşhis etmek ve sınıflandırmak genellikle güç bir işlemdir.

1.2.2. GÖRÜNTÜ EŞİKLEME TEKNİKLERİ:

Eşikleme bölütlemenin çok basit bir uygulama biçimidir. Gri seviyeli eşikleme teknikleri, sayısal bir görüntüyü ortak özel ve geniş bölümlere ayırmak için kolay hesaplanabilir metotlardır. Eşikleme işlemi, görüntünün birkaç anlamlı parçaya ayrılmasına dayanan en iyi eşik değerlerinin bir kümesinin belirlenmesi işlemini içermektedir.

Öncelikle bir eşik değeri tanımlanır, daha sonra bir görüntüdeki her piksel bu eşik değeri ile karşılaştırılır. Eğer piksel eşik değerinden büyükse nesne, aksi takdirde arka plan olarak belirlenir. Eşik değeri sıklıkla yoğunluk veya renk değeri olacaktır. Eşik değerinin görüntü üzerinde değişebildiği başka biçimleri de vardır fakat eşikleme ilkel bir teknik olduğundan sadece çok basit bölütleme uygulamalarında çalışacaktır. Eşikleme ile bölütlemenin bazı biçimleri, "Adobe Photoshop" gibi birçok grafik paketinde kullanılmaktadır [4-5].

1.2.2.1. İki Seviyeli Eşikleme:

İki seviyeli eşikleme çift tepeli histograma sahip görüntüler üzerinde kullanılır. İki seviyeli eşiklemede, nesne ve arka plan farklı gri seviyeli iki değişik gruba ayrılır.

Örnek verecek olursak:

1-) Bir kitaptaki alfa nümerik karakterler genellikle arka plandaki beyaz kağıttan daha karanlıktır.

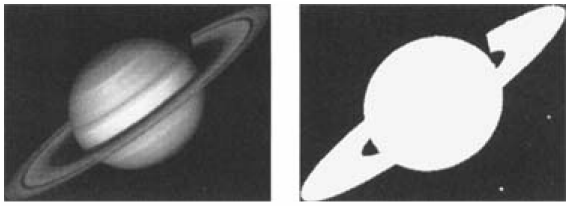
2-) Mitoz hücrelerin bir görüntüsündeki kromozomlar arka plandan daha karanlıktır.

Bu durumların her birinde, histogramların şekilleri aralarında bir oluk olan nesne ve arka plan bölümlerine karşı gelen çift tepeli bir yapıdadır. Oluk noktası genellikle eşik değeri olarak seçilir. Çift seviyeli eşiklemede, *T* eşik değerinden büyük tüm gri değerler nesne olarak belirlenirken, diğer tüm gri değerler arka plan olarak belirlenir, böylece nesne pikselleri arka plan piksellerinden ayrılır. Eşikleme böylece aşağıda görüldüğü gibi bir giriş görüntüsü *A* nın bölütlenmiş çıkış görüntüsü *B* ye dönüşümü işlemidir.

(a) $b_{ij} = 1,\ a_{ij} \geq T$ ise

(b) $b_{ij} = 0,\ a_{ij} \leq T$ ise (Burada *T* eşik değeridir)

Burada nesne pikselleri için $b_{ij} = 1$, arka plan pikselleri için $b_{ij} = 0$ dır. Şekil 1.3'de Satürn görüntüsünün çift seviyeli görüntü eşiklemesi görülmektedir.



Şekil 1.3: Çift tepeli görüntü eşiklemesi

Çift tepeli bir görüntüde eşik değeri seçimi için basit bir yinelemeli algoritma aşağıda gösterilmiştir [1].

1. Adım: Başlangıç için bir eşik değeri seçilir $T \leftarrow T_0$.

2. Adım: Görüntü *T* eşik değeri kullanılarak arka plan ve nesne olmak üzere iki bölüme ayrılır.

3. Adım: Sırasıyla arka plan ve nesnenin ortalama gri seviye değerleri μ_1 ve μ_2 hesaplanır.

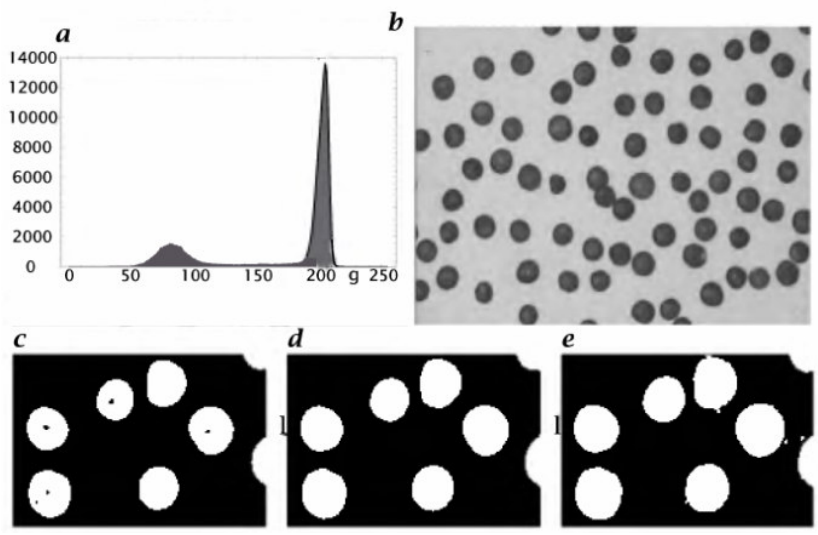
4. Adım: Yeni eşik değeri hesaplanır $T = \frac{\mu_1 + \mu_2}{2}$.

5. Adım: 2 den 4 e kadar olan adımlar *T* eşik değerinde bir değişim olmayana dek devam tekrarlanır.

1.2.2.2. Çok Seviyeli Eşikleme:

Çok seviyeli eşiklemede, görüntü birçok eşik değeri kullanılarak değişik bölümlere ayrılır. Bu durumda oluşan histogramlar, aralarında oluk olan çok tepeli yapılardır.

Eğer nesneler ayrıksa ve gri seviyeleri belirgin bir şekilde arka plandan farklı ise, bu durumda histogram her tepesi diğerinden açık bir şekilde ayrılan çok tepeli bir yapıdadır. Böyle bir görüntüyü bölütlerken, tepelerin arasındaki oluklar eşik değerleri olarak seçilirler. Böyle bir çok nesneli bölütleme örneği Şekil 1.4'de görülmektedir.



Şekil 1.4: Çok seviyeli görüntü eşiklemesi: (a) Çok tepeli histogram, (b) Giriş görüntüsü, (c)-(e) Giriş görüntüsünün sağ üst köşesinin sırasıyla 110, 147, 185 eşik değerleri için bölütlenmesi [5]

T eşik değeri aşağıdaki durumlara göre sınırlı ya da geniş çaplı olabilir.

1-) *T* eşik değeri sadece *g(x,y)* gri seviyesine dayanıyorsa eşik değeri geniş çaplıdır.

2-) Eşikleme, *T* eşik değeri, *N(x,y)* nin *(x,y)* noktasındaki sınırlı bir görüntünün özelliğini gösterdiği, hem *g(x,y)* hem de *N(x,y)* ye dayandığı zaman sınırlıdır. Sınırlı

görüntü özelliği, *(x,y)* noktası etrafındaki bir komşuluğun ortalama gri seviyeleri gibi sınırlı görüntü istatistikleriyle hesaplanabilir.

İlave olarak, eğer *T* eşik değeri *g(x,y)* ve *N(x,y)* üzerindeki gibi bir zamana dayanıyorsa bu durumda eşikleme dinamik olarak isimlendirilir.

1.2.2.3. Entropi Tabanlı Eşikleme:

Entropi tabanlı eşikleme genellikle çift seviyeli eşiklemede kullanılır. Entropi Shannon tarafından tanımlanan bir görüntüdeki bilginin bir ölçüsüdür [6]. Shannon entropisinin değişkenleri, görüntü bölütlemede eşik değerlerinin belirlenmesi için etkin bir biçimde kullanılır. Entropi tabanlı eşiklemede, nesne ve arka plan bölümlerinin entropisi eşik değerlerinin en uygun değer olarak seçilmesi için kullanılır.

Kapur eşikleme tekniğinde nesne ve arka plan bölümlerinin entropileri aşağıdaki gibidir:

$$H_f(T) = -\sum_{g=0}^{T} \frac{p(g)}{P_f(T)} \ln \frac{p(g)}{P_f(T)} \qquad \textbf{(1.3)}$$

$$H_b(T) = -\sum_{g=T+1}^{L-1} \frac{p(g)}{P_f(T)} \ln \frac{p(g)}{P_f(T)} \qquad \textbf{(1.4)}$$

L seviyeli gri bir görüntüde, nesne gri seviye dağılımı 0-*T* aralığında ve arka plan pikselleri $[T+1, L-1]$ aralığında yer alır. (1.3) ve (1.4) eşitliklerinde *p(g)* olasılık küme fonksiyonu, *h(g)* nin *g* gri değerinin histogramı ve *N* nin piksellerin toplam sayısı olduğu aşağıdaki ifadeyle belirlenir:

$$p(g) = \frac{h(g)}{N}, \; g = 0,1,\ldots\ldots\ldots,L-1$$

Nesne ve arka plan alan olasılık değerleri aşağıdaki biçimdedir:

$$P_f(T) = \sum_{g=0}^{T} p(g) \text{ ve } P_b(T) = \sum_{g=T+1}^{L-1} p(g)$$

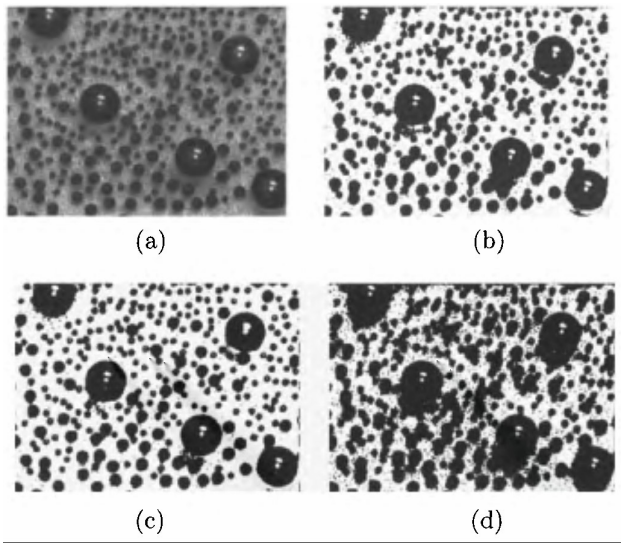
Eşikleme stratejisi hem nesne hem de arka planın toplam entropisini maksimize eder. Toplam entropinin maksimum değeri için arzu edilen eşik değeri 1 dir.

Renyi entropisi de görüntü eşikleme için kullanılabilir [7]. Renyi entropisi için nesne ve arka plan bölümleri aşağıdaki gibidir:

$$H_f(T) = \frac{1}{1-\rho}\ln\left[\sum_{g=0}^{T}\left(\frac{p(g)}{P(T)}\right)^{\rho}\right],\ H_b(T) = \frac{1}{1-\rho}\ln\left[\sum_{g=T+1}^{L-1}\left(\frac{p(g)}{1-P(T)}\right)^{\rho}\right]$$

Bu eşikleme tekniğinde, nesne ve arka plan bölümlerinin toplam entropisi, çeşitli ρ için hesaplanır ve en iyi eşikleme sonuçlarını sağlayan uygun ρ değeri seçilir.

Şekil 1.5'de Otsu, Kapur, Renyi entropi tabanlı eşikleme algoritmaları kullanılarak elde edilen eşiklenmiş görüntüler görülmektedir.



Şekil 1.5: Eşikleme tekniklerinin karşılaştırılması: (a) Orijinal görüntü, (b) Otsu, (c) Kapur, (d) Renyi entropileri [1]

1.2.2.4. Karşılaşılan Problemler ve Mümkün Çözümler:

Bir görüntüde iki sınırlı maksimumun aynı geniş çaptaki maksimuma ait olduğu durumlarla oldukça sık karşılaşmaktayız. Bu sınırlı maksimumun algılanmasını engellemek için bazı teknikler kullanılabilir:

- Bu maksimumlar arasındaki gri seviyelerin minimum bir mesafesi, genellikle iyi bir eşikleme için gereklidir. Histogram çift tepeli olsa bile, doğru bölütleme farklı gri seviyeler içeren bir arka plandaki nesnelerle meydana gelmeyebilir. Histogram iyi bir eşikleme elde etmek için düzlenebilir.

- Yüksek bir görüntü gradyanı ile piksellerin etkisi, azaltılabilir. Bu durum, histogramın orta grilikteki değerlere sahip kenar piksellerini silerken, büyük çoğunlukla ya nesneye ya da arka plana ait gri seviyeleri içereceğini göstermektedir. Bu durum daha derin bir oluk oluşturacaktır ve böylece eşik değerinin daha kolay belirlenmesine olanak sağlayacaktır.

- Başka bir teknik, tepesi nesne ve arka plan arasındaki gri seviye sınırlarına karşı gelen gri seviye histogramını biçimlendirmek için sadece gri seviyeli gradyan piksellerini kullanır. Bu histogram tek tepeli olmalı ve eşik değeri tepenin gri seviyesine göre seçilmiş olmalıdır.

1.2.3. BÖLGE TABANLI BÖLÜTLEME:

Bölge tabanlı metotlar kenar tabanlı metotları tamamlayıcı niteliktedir. Bölge tabanlı metotlardaki temel nokta, aynı ya da benzer parlaklıktaki pikselleri verilen homojenlik kıstasına göre bölgelere ayırmaktır. Bu metotlar verilen pikselin komşu piksellerine bakar ve eğer homojenlik kıstası sağlanırsa bir bölgede birleştirir. Homojenlik kıstası, seçimi büyük bir problem olan bazı eşik değerlerine dayanmaktadır. Çünkü doğru eşik değerini belirleyebilmek için sürekli deneme yapılmaktadır ve bu eşik değerleri her zaman görüntü bilgisine dayanmaktadır [8].

1.2.3.1. Bölge Büyütme Metodu:

Bölge büyütme, pikselleri veya alt bölgeleri daha büyük bölgeler oluşturacak şekilde gruplayan prosedür ile ilgilidir. Kaynak noktalarının bir kümesiyle başlayarak, bölgeler, her bir kaynak noktasına yoğunluk, gri seviye derecesi, renk gibi aynı niteliklere sahip olan komşu pikselleri ilave ederek bu kaynak noktalarından büyütülür. Bu, her piksel işleme sokulana kadar her kaynak pikselin yinelemeli olarak büyüdüğü ve öylelikle sınırları kapalı çokgenlerle belirlenen değişik bölgelerin şekillendiği yinelemeli bir işlemdir.

Bölge büyütmede önemli konular:

- Başlangıç kaynaklarının seçimi, bölgeleri ve büyütme işlemi süresince farklı bölgelerdeki noktaları içine alan uygun özelliklerin seçimini belirtir.

- Görüntünün belirli özelliklerine dayanan pikselleri büyütme işlemi, iyi bölütleme belirlemeyebilir. Bağlanabilirlik veya bitişiklik bilgisi de bölge büyütme işleminde kullanılmalıdır.

- Benzerlik: Benzerlik, uzaysal bitişik iki piksel veya piksellerin bir kümesinin ortalama gri seviyesi arasında gözlenen, değişik bölgeler meydana getiren gri seviyedeki minimum farkı gösterir. Eğer bu fark benzerlik eşik değerinden daha düşükse, pikseller aynı bölgeye ait olur.

- Bölgenin Alanı: Minimum alan eşik değeri piksellerdeki en küçük bölge büyüklüğü ile ilişkilidir. Bölütlenmiş görüntüde, hiçbir bölge kullanıcı tarafından tanımlanan bu eşik değerinden daha küçük olmayacaktır.

Bölge Büyütme Son-İşlemi: Bölge büyütme optimum olmayan parametre ayarlarının bir sonucu olarak sıklıkla az büyüme ya da fazla büyüme ile sonuçlanır. Kenar tabanlı ve bölge büyütme tabanlı bölütlemeden elde edilen bölütleme bilgisini birleştiren bir son işlemci geliştirilmiştir. Daha basit son işlemciler, genel bulgulara dayanır ve bölütlenmiş görüntüdeki orijinal olarak uygulanan homojenlik kıstasına göre herhangi bir komşu bölgeyle birleştirilemeyen küçük bölgelerin sayısını azaltır.

1.2.3.2. Bölge Birleştirme ve Bölme:

Bir bölütleme algoritması olan bölge birleştirme ve bölmede, görüntüdeki tek bir büyük bölgenin parçalanması yüzünden birçok küçük bölge üretebilir. Böyle bir durumda, daha küçük bölgelerin benzerlik ve yoğunluğuna dayanarak birleştirilmesine ihtiyaç vardır. Basit bir bölge birleştirme algoritması aşağıda gösterilmiştir [1]:

1. Adım: Eşik değerlerinin bir kümesi kullanılarak görüntü $R_1, R_2, ..., R_m$ gibi bölgelere bölütlenir.

2. Adım: Görüntünün bölütlenmiş tanımından bir bölge bitişiklik grafı *(RAG)* oluşturulur.

3. Adım: Her R_i, $i = 1,2,.....,m$ için R_i nin R_j ye bitişik olduğu tüm R_j, $j \neq i$ ler RAG dan tanımlanır.

4. Adım: Her *i* ve *j* için R_i ve R_j arasında uygun bir benzerlik ölçüsü S_{ij} hesaplanır.

5. Adım: $S_{ij} > T$ ise R_i ve R_j birleştirilir.

6. Adım: 3 ten 5 e kadar olan adımlar benzerlik kıstasına göre birleştirilecek herhangi bir bölge kalmayana dek tekrar edilir.

Birleştirme için başka bir strateji iki bölgenin arasındaki kenarların yoğunluğuna dayanır. Bu metotta, komşu bölgeler arasındaki birleştirme işlemi, iki bölge arasında çizilen sınırın uzunluğu boyunca var olan kenar yoğunluğuna dayanır. Eğer kenar yoğunluğu küçükse kenar noktaları zayıftır, bu durumda eğer birleşme ortalama piksel yoğunluğu değerlerini fazla değiştirmiyorsa bu iki bölge birleştirilebilir.

Hatalı başlangıç bölütlemesi yüzünden çok küçük bölgelerin meydana geldiği bazı durumlar vardır. Bu durum farklı bölgelerin tek bir bölgeye hatalı birleşmesinden dolayı meydana gelir. Bu gibi bir durumda, bölütlenmiş bir görüntüdeki gri değerlerin değişimi bir eşik değerinin üzerinde olabilir ve bu nedenle bölgelerin her bir küçük bölgenin aynı küçük değişimlere sahip olduğu daha küçük bölgelere bölünmesi gerekmektedir.

Bölme ve birleştirme, bölme ve birleştirme işlemi uygulamalarının bir kurala dayandığı karmaşık görüntüleri bölütlemek için birleştirilebilir.

1.2.4. KÜMELEMEYE DAYALI BÖLÜTLEME:

Bilgi sürümlü bölütleme teknikleri histogram yönlendirmeli ve küme yönlendirmeli olabilir. Histogram yönlendirmeli bölütleme, çok özellikli bir bilginin her özelliği için ayrı bir bölütleme meydana getirir, daha sonra daha çok parçalanmış bölgeler meydana getirmek için her özellikten bölütleme sonuçlarını üst üste bindirir. Küme yönlendirmeli bölütleme görüntü piksellerini kümelere bölmek için çok boyutlu bilgi kullanır. Her pikselin bazı niteliklere sahip olduğu ve bir vektörle gösterildiği görüntüleri bölütlemede, küme yönlendirmeli teknikler histogram yönlendirmeli tekniklerden daha uygun olabilir. Küme analizleri 1960 lardan beri çok dikkat çekmektedir ve OCR (optik karakter tanıma) sistemi, parmak izi tanıma, hareket algılama, biyolojik görüntü bölütlemesi gibi pek çok alanda kullanılmaktadır.

Kümeleme, üyeleri bazı benzer özelliklere sahip olan nesnelerin gruplara ayrılması işlemi olarak tanımlanabilir. Aşağıdaki tanımlamalar kümeleme tabanlı bölütlemeyi açıklamak için faydalı olacaktır:

- Bir küme, benzer olan elemanların bir grubudur. Farklı kümelerdeki elemanlar benzer değildirler.

- Bir küme, kümedeki herhangi iki nokta arasındaki mesafenin, kümenin içindeki ve dışındaki herhangi bir noktaya olan mesafesinden daha kısa olduğu test alanındaki noktaların birleşimidir.

- Kümeler, oldukça yüksek yoğunlukta nokta içeren ve diğer bölgelerden, daha az yoğunluklu noktalar içeren bir bölge ile ayrılan çok boyutlu bir uzayın birleştirilmiş bölgeleri olarak tanımlanabilir.

Kümeleme algoritmalarındaki araştırmalar patern tanıma ve bilgi işleme gibi birçok alanda oldukça kullanışlıdır. Kümeleme metotları hiyerarşik [9] ve parçalı olmak üzere ikiye ayrılabilir [10].

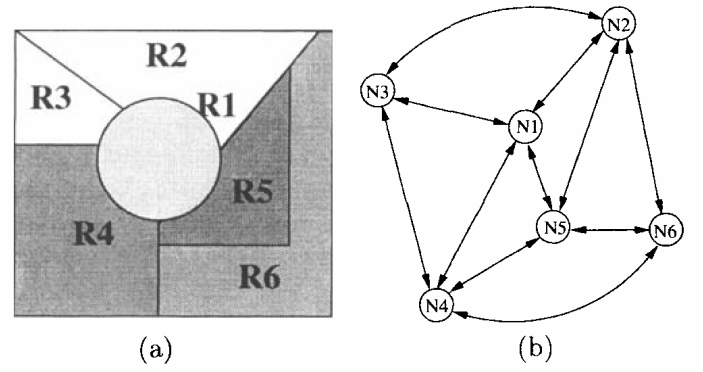
1.2.4.1. Graf Bazlı Kümeleme Metodu İle Bölütleme:

Bölütlemeye bir bakış açısı da, nesnenin bulunduğu imgenin piksellerinin, ya da nesnelere ait özniteliklerin oluşturduğu veri setindeki bileşenlerin aralarındaki ilişkilere göre, hangilerinin birer alt küme oluşturabileceğinin bulunmasıdır. Bu açıdan bakıldığı zaman problemimiz bir kümeleme problemi olmaktadır.

Kümeleme probleminin çözümüne çok çeşitli yaklaşımlar vardır. Bunlardan bir tanesi de graflar ile geliştirilmiş olan çözümdür. Her bir veri, graf üzerinde bir düğüm ile gösterilir. Dataların birbirleri ile olan ilişkileri ise, düğümler arasındaki kenarlardır. İki veri arasında kuvvetli bir ilişki varsa (bunun anlamı, bu iki verinin birbirlerine temel alınan özniteliğe göre benzer olduğudur), bu iki veri arasındaki kenar bilgisinin ağırlığı yüksek olarak belirlenir. Eğer verilerin birbirleri ile benzerlikleri az ise, bu iki veri düğümünün arasındaki kenar bilgisi daha düşük tutulacaktır. Veriler arasındaki ilişkiler ağırlıklı bir graf haline getirildikten sonra, problemimiz böyle bir grafı en iyi parçalara ayırma (graph cut) problemi haline gelecektir.

Ağırlıklı grafı, verilerin aralarındaki kenar maliyetlerinin en fazla olacağı şekilde bileşenlere bölümlemek, en düşük maliyetli kenarlardan kesmek ile mümkün olacaktır. Grafları, aralarındaki ağırlık bilgisi en fazla olacak şekilde bileşenlerine ayırmak için çeşitli algoritmalar ileri sürülmüştür [4].

Bölge Bitişiklik Grafı: Bir görüntüdeki bölgeler üzerindeki bitişiklik ilişkisi bir bölge bitişiklik grafı *(RAG)* ile gösterilebilir. Görüntüdeki bölümler RAG daki bir $N = \{N_1, N_2, ..., N_m\}$ düğümler kümesi ile gösterilir. Burada N_i düğümü görüntüdeki R_i bölgesini gösterir ve R_i bölgesinin özellikleri düğüm veri yapısı N_i de saklanır. N_i ve N_j arasındaki $e_{i,j}$ kenarı R_i ve R_j bölgeleri arasındaki bitişikliği gösterir. R_i ve R_j bölgeleri, R_i bölgesindeki bir piksel ve R_j bölgesindeki bir piksel birbirine bitişik ise bitişiktirler. Bitişiklik 4 komşuluklu veya 8 komşuluklu olabilir. Bitişiklik ilişkisi, yansıyan ve simetriktir fakat geçişken olmak zorunda değildir. Şekil 1.6'da altı ayrı bölgeden oluşan bir görüntünün bitişiklik grafı görülmektedir.



Şekil 1.6: (a) 6 farklı bölgeden oluşan bir görüntü, (b) Görüntün bitişiklik grafı

İkilik tabandaki bir *A* matrisi, bir bölge bitişiklik grafı (RAG) gösterdiği zaman bir bitişiklik matrisi olarak adlandırılır. RAG daki N_i ve N_j düğümleri bitişik olduğunda *A* matrisindeki $a_{i,j}$ değeri 1 dir. Bitişiklik ilişkisi yansıma özelliğine sahip olduğundan matrisin tüm diyagonal elemanları 1 dir. Çok bölgeli bir görüntü olan Şekil 1.8'in bitişiklik matrisi aşağıdaki gibidir:

$$A = \begin{bmatrix} & N1 & N2 & N3 & N4 & N5 & N6 \\ N1 & 1 & 1 & 1 & 1 & 1 & 0 \\ N2 & 1 & 1 & 1 & 0 & 1 & 1 \\ N3 & 1 & 1 & 1 & 1 & 0 & 0 \\ N4 & 1 & 0 & 1 & 1 & 1 & 1 \\ N5 & 1 & 1 & 0 & 1 & 1 & 1 \\ N6 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Şekil 1.7: Bitişiklik Matrisi

2. HÜCRESEL SİNİR AĞLARI

Hopfield Sinir Ağı'nın [11] özel bir kümesi olan Hücresel Sinir Ağları 1988 yılında Leon O. Chua ve Lin Yang tarafından önerilmiştir. Bu bölümde Hücresel Sinir Ağları kavramı ve temel tanımlamaları verilecektir.

2.1. Hücresel Sinir Ağlarının Keşfi

Sinir sistemimizdeki nöronların birbirine olan bağlantılarına yoğunlaşan iki bilim adamı Leon O. Chua ve Lin Yang 1988 yılında yeni bir yapay sinir ağı mimarisi ortaya koydukları makalelerini [12-13] yayınladılar ve bu makale ile Hücresel Sinir Ağı (HSA) doğmuş oldu.

Sinir sistemimizdeki nöronların diğer nöronlarla bölgesel olarak bağlı olması HSA yapısının dayandığı en önemli özelliktir. Hopfield sinir ağlarındaki bütün hücrelerin birbirleriyle doğrudan bağlılığına karşın HSA'nda her bir hücre en yakın komşuluğundaki hücre ile doğrudan, diğer hücrelerle dolaylı olarak bağlıdır.

Bölgesel bağlantı sonucu; hem devrenin kapladığı alan azalmakta hem de verilerin devre üzerindeki taşınım yolları kısa olduğundan toplam güç harcaması düşük olmaktadır. HSA nın önemli avantajlarından bir diğeri de kararlı bir sonuca yakınsama süresinin devrenin boyutundan bağımsız olması ve bir kaç nöron gecikmesi ile belirlenmesidir.

Prof. Roska ve Prof. Chua tarafından HYSA ağını merkezi işlem birimi alan yeni bir bilgisayar mimarisi ortaya atıldı [14]. Bu mimari aslında analog ve sayısal işlem yapma yeteneğine sahip bir merkezi işlem birimine sahipti. Bu nedenle HYSA-UM (CNN-UM) olarak adlandırılan işlemci, analog ve lojik işlem yeteneği nedeniyle bu iki terimin birleşimi olan analogic işlemci olarak nitelendirilmektedir. Mimarisi analog ve sayısal olmak üzere yerel ve evrensel (global) belleğe ayrıca HYSA nı programlamaya yarayan şablonların tutulduğu bellek alanlarına sahiptir.

HSA'nın bu özellikleri gelecekte pek çok uygulama alanı bulacağının göstergeleridir. HSA günümüzde hava kirliliği modellemesi, ozon konsantrasyonunun modellenmesi, retina modellemesi, biyonik göz, otomatik yönlendirme gibi birçok uygulamaya sahiptir [15–18]. HSA nın görüntü işlemedeki başlıca uygulamaları ise karakter tanıma, tümör tespiti, görüntü onarımı olarak sayılabilir [19].

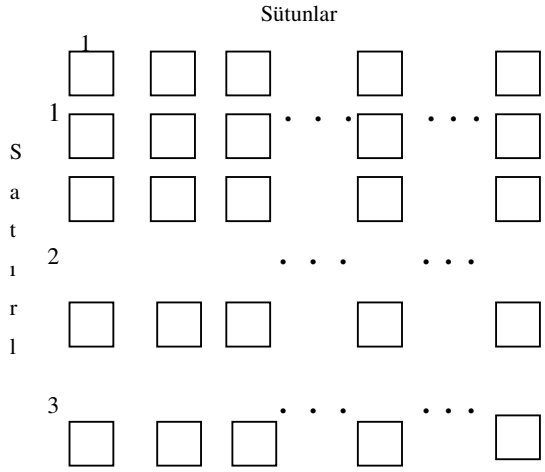
2.2. Hücresel Sinir Ağlarının Mimarisi

Bu kısımda hücresel sinir ağı mimarisi, ayrıntılı olarak incelenmiştir.

2.2.1 Temel Gösterilim ve Tanımlar

Tanım 2.1. Standart HSA Mimarisi

Standart HSA yapısı *M*x*N* boyutundaki dikdörtgen hücreler dizisinden oluşur. Burada *M*: satırı *N*: sütunu gösterir. HSA'nın en küçük birimine hücre adı verilir. Hücreler devrenin boyutuna göre iki boyutlu uzayda kartezyen koordinat sistemi düzeninde yerleştirilmişlerdir. *i. satır*, *j. sütun'*daki hücre *C(i,j)* ile ifade edilir. (*i=1,2,3...M, j=1,2,3...N*) (Şekil 2.1)



Şekil 2.1: Standart HSA mimarisi

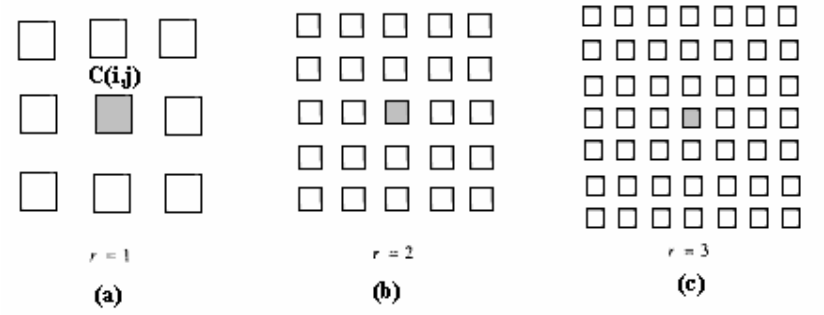
Uygulamalarda genellikle $M = N$ alınsa da $M \neq N$ de olabilir. Örneğin *5 x 512'* lik bir HSA yapısı tarayıcı, faks ya da fotokopi makinesi için uygundur.

Tanım 2.2. C*(i,j)* Hücresinin Etki Küresi [31]

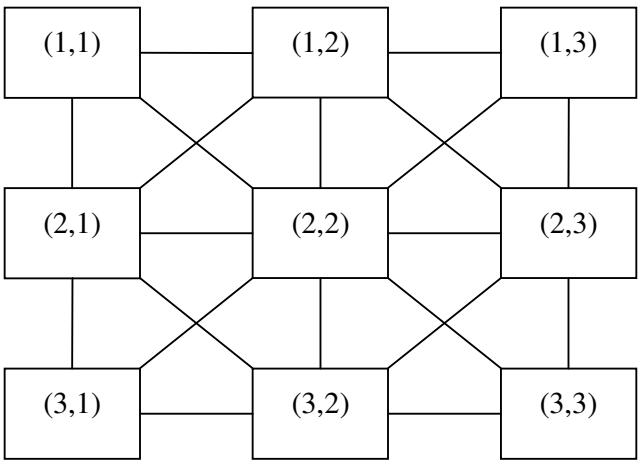
C(i,j) hücresinin r yarıçaplı etki küresi $S_r(i, j)$ olmak üzere;

$$S_r(i, j) = \left\{ C(k,l) \left| \max_{1 \le k \le M, 1 \le l \le N} \{ |k - i|, |l - j| \} \le r \right. \right\} \quad \textbf{(2.1)}$$

koşulunu sağlayan tüm komşu hücreler topluluğuna denir. Buradaki r pozitif bir tam sayıdır.



Şekil 2.2: (a)r=1 (3x3 komşuluk) , **(b)**r=2 (5x5 komşuluk), **(c)**r=3 (7x7 komşuluk)



Şekil 2.3: Bir HSA yapısındaki hücreler arası etkileşim

Komşuluk *r* yarıçapı ile belirtilebildiği gibi *(2r+1)*x*(2r+1)* şeklinde de gösterilebilir. Etki küresi içerisinde $(2r+1)^2$ kadar hücre bulunmaktadır.

Komşuluk sistemi simetri özelliğine sahiptir.

$$C(k,l) \in S_r(i,j) \Leftrightarrow C(i,j) \in S_r(k,l) \qquad \textbf{(2.2)}$$

Hücresel Sinir Ağı tanımlanırken, bir r değeri seçilir (Çoğu uygulamada r =1 alınır). Bu seçimden sonra her hücre sadece kendi komşuluk grubundaki hücrelerle doğrudan bağlantılı bulunur. Yani sadece komşuları ile sinaptik bağlantıları mevcuttur.

Tanım 2.3. Bir Hücrenin Matematiksel Tanımı

En genel halde bir HSA'nda *C(i,j)* hücresinin matematiksel tanımlamaları şu şekildedir;

Tanım 2.5. Durum Denklemi [31]

$$\frac{dx_{ij}}{dt} = -x_{ij} + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)\,y_{kl} + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u_{kl} + z_{ij} \qquad \textbf{(2.3)}$$

Burada;

$x_{ij} \in R$; *C(i,j)* Hücresinin durum değişkeni,

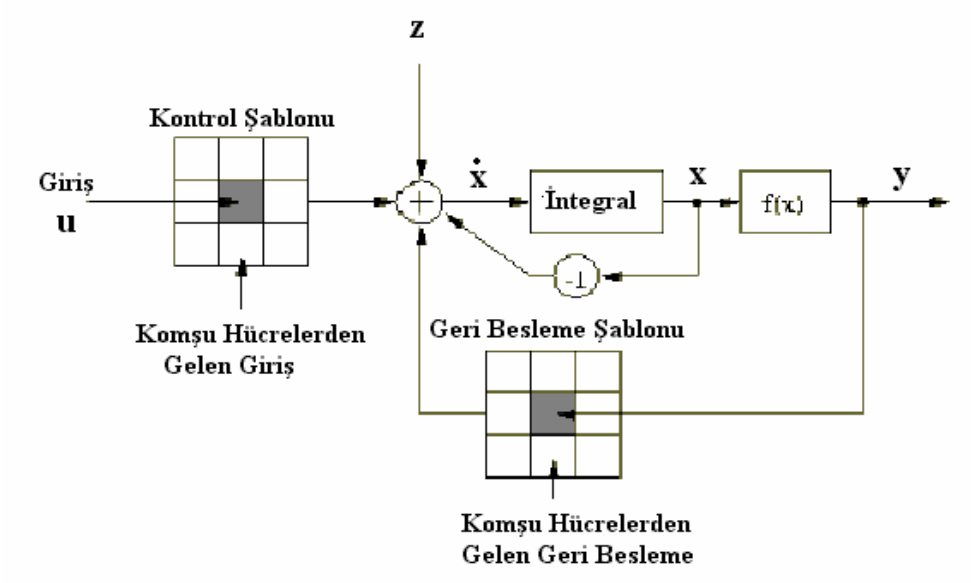
$y_{kl} \in R$; Etki küresi içerisindeki hücrelerin çıkışları,

$u_{kl} \in R$; Etki küresi içerisindeki hücrelerin girişleri,

$z_{ij} \in R$; Eşik değeri

A(i, j ; k, l) ; Geri Besleme operatörü

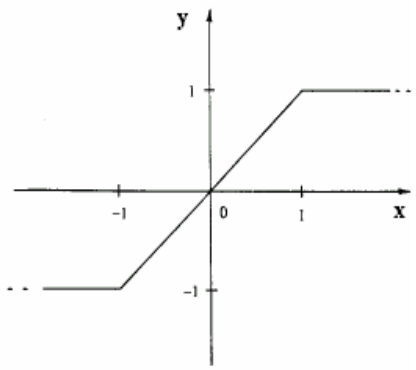
B(i, j ; k, l) ; Giriş (Kontrol) operatörüdür ve bu modele ilişkin blok diyagram Şekil 2.4 te verilmiştir. HSA ya ilişkin doğrusal olmayan fonksiyon *f(x,y)* Şekil 2.5 te gösterilmiştir.



Şekil 2.4: Bir hücrenin blok diyagram gösterilimi

Tanım 2.6. Çıkış Denklemi

$$y_{ij} = f(x_{ij}) = \frac{1}{2}\left|x_{ij}+1\right| - \frac{1}{2}\left|x_{ij}-1\right| \qquad \textbf{(2.4)}$$



Şekil 2.5: Eşik Aktivasyon Fonksiyonu

HSA'na lineer olmama özelliğini veren bu fonksiyondur.

Tanım 2.7. Sınır Koşulları

Sınır hücrelerinin etki küresi içinde bulunup da *M x N* boyutundaki hücre dizisinin dışında kalan hücrelerin durum ve çıkış değerleri koşuludur.

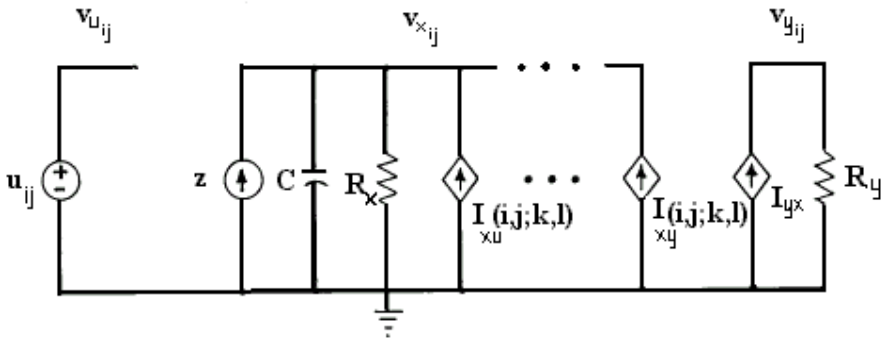
Tanım 2.8. Başlangıç Koşulları

Dizideki tüm hücrelerin ilk koşuludur.

$x_{ij}(0)$, i=1,2...M , j=1,2...N

2.3 Hücre Yapısı

HSA'nın bir hücresinin devre modeli Şekil 2.6 da görülmektedir.



Şekil 2.6: Bir Hücrenin devre şeması [20]

Hücrede sırasıyla giriş, durum ve çıkışa karşı düşen ***u, x*** ve ***y*** düğümleri bulunur. Hücrede bulunan elemanlar $\boldsymbol{u_{ij}}$; sabit gerilim kaynağı, ***Z***; sabit akım kaynağı, ***C***; lineer kapasite, Rx ve $\boldsymbol{R_y}$; lineer dirençler, m komşu hücre sayısına eşit olmak üzere en fazla 2m adet $\boldsymbol{I_{xu}(i,j;k,l)}$ ve $\boldsymbol{I_{xy}\ (i,j;k,l)}$ bağımlı akım kaynağı, son olarak da $\boldsymbol{I_{yx}}$ lineer olmayan gerilim kontrollü akım kaynağıdır. Komşu hücrelerle bağlantı, kontrol girişi $v_{u_{kl}}$ ve geri besleme gerilimi $v_{y_{kl}}$ 'nin ağırlıklı toplamları üzerine kurulmuştur.

$$I_{xy}(i,j;k,l) = A(i,j;k,l)v_{y_{kl}} \quad , \quad \forall\, C(k,l) \in S_r(i,j)$$

$$I_{xu}(i,j;k,l) = B(i,j;k,l)v_{u_{kl}} \quad , \quad \forall\, C(k,l) \in S_r(i,j)$$

Hücredeki tek lineer olmayan eleman ise I_{yx} parça-lineer gerilim kontrollü akım kaynağıdır.

$$I_{yx} = \frac{1}{R_y} f(v_{x_{ij}})$$

Buradaki *f(·)* fonksiyonu Tanım 2.6'da verilen çıkış fonksiyonudur.

HSA'nın VLSI gerçeklemelerinde *f(·)* fonksiyonunun keskin karakteristiğini elde etmek mümkün olmadığından daha yumuşak ve sürekli bir fonksiyon olan Sigmoid fonksiyonu kullanılır. Bu değişim teoriyi etkilemez [20].

2.4 Zamandan ve Konumdan Bağımsız HSA

Eğer geri besleme A(i, j ; k, l), giriş B(i, j ; k, l) operatörleri ve z_{ij} eşik değeri, hücrenin ağ içerisindeki yerleşiminden ve zamandan bağımsız olarak belirleniyorsa bu yapıya 'Zamandan ve Konumdan Bağımsız HSA' adı verilir. Uygulamaların neredeyse tamamı bu yapıyı kullanır.

Bu durumda aşağıdaki ifadeleri yazabiliriz;

$$\sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)\,y_{kl} = \sum_{|k-i|\le r} \sum_{|l-j|\le r} A(i-k,j-l)\,y_{kl}$$

$$\sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)\,u_{kl} = \sum_{|k-i|\le r} \sum_{|l-j|\le r} B(i-k,j-l)\,u_{kl}$$

$$z_{ij} = z$$

2.4.1 Klonlama Şablonu Gösterilimi

HSA'nın uygulamalarında en fazla *3x3* komşuluğun (etki küresi yarıçapı r =1) bulunduğu, konumdan bağımsız yapı kullanılır. Bu uygulamalarda analizini kolaylaştıracak bazı tanımlamalar yapılmıştır.

Geri besleme (*A(i-k , j-l)*), giriş (*B(i-k , j-l)*) operatörleri ve z_{ij} eşik değeri, hücrenin ağ içerisindeki yerleşiminden bağımsız olduklarından, bir şablon şeklinde gösterilebilirler.

2.4.1.1. Geri Besleme Operatöründen (A(i-k , j-l)) Gelen Katkı

$$\sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)\,y_{kl} = \sum_{|k-i|\le r} \sum_{|l-j|\le r} A(i-k,j-l)\,y_{kl} =$$

$$= a_{-1,-1}\,y_{i-1,j-1} + a_{-1,0}\,y_{i-1,j} + a_{-1,1}\,y_{i-1,j+1} +$$
$$+ a_{0,-1}\,y_{i,j-1} + a_{0,0}\,y_{i,j} + a_{0,1}\,y_{i-1,j+1} +$$
$$+ a_{1,-1}\,y_{i+1,j-1} + a_{1,0}\,y_{i+1,j} + a_{1,1}\,y_{i+1,j+1} =$$
$$= \sum_{k=-1}^{1} \sum_{l=-1}^{1} a_{k,l}\,y_{i+k,j+l} =$$

$$= \begin{array}{|c|c|c|}\hline a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ \hline a_{0,-1} & a_{0,0} & a_{0,1} \\ \hline a_{1,-1} & a_{1,0} & a_{1,1} \\ \hline \end{array} \otimes \begin{array}{|c|c|c|}\hline y_{i-1,j-1} & y_{i-1,j} & y_{i-1,j+1} \\ \hline y_{i,j-1} & y_{i,j} & y_{i-1,j+1} \\ \hline y_{i+1,j-1} & y_{i+1,j} & y_{i+1,j+1} \\ \hline \end{array} = A \otimes Y_{ij}$$

$$A = \begin{array}{|c|c|c|}\hline a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ \hline a_{0,-1} & a_{0,0} & a_{0,1} \\ \hline a_{1,-1} & a_{1,0} & a_{1,1} \\ \hline \end{array}$$

$$Y_{ij} = \begin{array}{|c|c|c|}\hline y_{i-1,j-1} & y_{i-1,j} & y_{i-1,j+1} \\ \hline y_{i,j-1} & y_{i,j} & y_{i-1,j+1} \\ \hline y_{i+1,j-1} & y_{i+1,j} & y_{i+1,j+1} \\ \hline \end{array}$$

Burada *3 x 3* boyutundaki ***A*** matrisi **'Geri Besleme Klonlama Şablonu'** olarak isimlendirilir.

$\boldsymbol{Y_{ij}}$ matrisi *M*x*N* boyutundaki çıkış görüntüsünün üzerinde *3*x*3* bir pencere gezdirilerek elde edilebileceğinden '***C(i,j)*** **Hücresindeki Çıkış Görüntüsü'** olarak adlandırılır.

$\otimes$ sembolü nokta çarpımların toplamını ifade eder ve **'Şablon Nokta Çarpım'** olarak isimlendirilir. Ayrık matematikte bu işlem uzaysal konvolüsyona karşı düşer.

Bazı uygulamalarda **A** şablonu şu şekilde ikiye ayrılabilmektedir:

$\boldsymbol{a_{00}}$**;** ***A*** şablonunun merkez elemanını göstermek üzere (a_{kl} ; *k=l=0*);

$$A = A^0 + \bar{A} \quad , \quad A^0 = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & a_{00} & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \quad , \quad \bar{A} = \begin{array}{|c|c|c|}\hline a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ \hline a_{0,-1} & 0 & a_{0,1} \\ \hline a_{1,-1} & a_{1,0} & a_{1,1} \\ \hline \end{array} \qquad \textbf{(2.5)}$$

A^0 ; Şablonun Merkez Bileşeni,

$\bar{A}$; Şablonun Çevre Bileşeni olarak adlandırılır.

2.4.1.2. Giriş (Kontrol) Operatöründen (B(i-k, j-l)) Gelen Katkı

Bir önceki bölümde yapılan işlemlere benzer işlemlerle;

$$\sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u_{kl} = \sum_{|k-i|\le r} \sum_{|l-j|\le r} B(i-k, j-l)u_{kl} =$$

$$\begin{aligned} &= b_{-1,-1}u_{i-1,j-1} + b_{-1,0}u_{i-1,j} + b_{-1,1}u_{i-1,j+1} + \\ &+ b_{0,-1}u_{i,j-1} + b_{0,0}u_{i,j} + b_{0,1}u_{i-1,j+1} + \\ &+ b_{1,-1}u_{i+1,j-1} + b_{1,0}u_{i+1,j} + b_{1,1}u_{i+1,j+1} = \\ &= \sum_{k=-1}^{1} \sum_{l=-1}^{1} b_{k,l}u_{i+k,j+l} = \end{aligned}$$

$$= \begin{array}{|c|c|c|}\hline b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ \hline b_{0,-1} & b_{0,0} & b_{0,1} \\ \hline b_{1,-1} & b_{1,0} & b_{1,1} \\ \hline \end{array} \otimes \begin{array}{|c|c|c|}\hline u_{i-1,j-1} & u_{i-1,j} & u_{i-1,j+1} \\ \hline u_{i,j-1} & u_{i,j} & u_{i-1,j+1} \\ \hline u_{i+1,j-1} & u_{i+1,j} & u_{i+1,j+1} \\ \hline \end{array} = B \otimes U_{ij}$$

$$B = \begin{array}{|c|c|c|}\hline b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ \hline b_{0,-1} & b_{0,0} & b_{0,1} \\ \hline b_{1,-1} & b_{1,0} & b_{1,1} \\ \hline \end{array}$$

$$U_{ij=} \begin{array}{|c|c|c|}\hline u_{i-1,j-1} & u_{i-1,j} & u_{i-1,j+1} \\ \hline u_{i,j-1} & u_{i,j} & u_{i-1,j+1} \\ \hline u_{i+1,j-1} & u_{i+1,j} & u_{i+1,j+1} \\ \hline \end{array}$$

*3*x*3* boyutundaki ***B*** matrisi **'Giriş (Kontrol) Klonlama Şablonu'** olarak isimlendirilir.

$\boldsymbol{U_{ij}}$ matrisi *M*x*N* boyutundaki giriş görüntüsünün üzerinde *3x3* bir pencere gezdirilerek elde edilebileceğinden '***C(i,j)* Hücresindeki Giriş Görüntüsü'** olarak adlandırılır.

Bazı uygulamalarda **B** şablonu şu şekilde ikiye ayrılabilmektedir.

$\boldsymbol{b_{00}}$**;** ***B*** şablonunun merkez elemanını göstermek üzere. (b_{kl} ; $k=l=0$)

$$B = B^0 + \bar{B} \quad , \quad B^0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & b_{00} & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad , \quad \bar{B} = \begin{bmatrix} b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & 0 & b_{0,1} \\ b_{1,-1} & b_{1,0} & b_{1,1} \end{bmatrix}$$

B^0**;** Şablonun Merkez Bileşeni

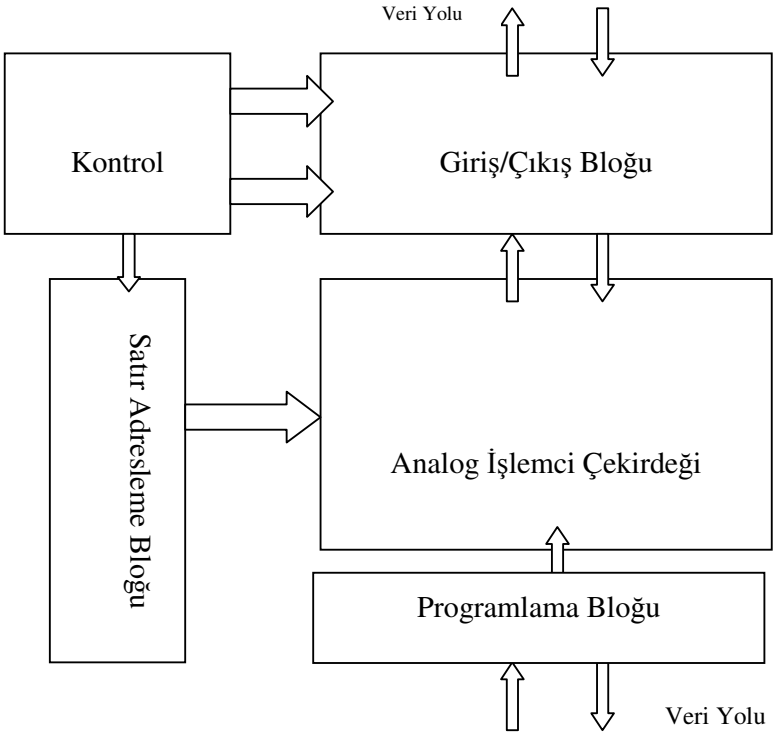
$\bar{B}$**;** Şablonun Çevre Bileşeni olarak adlandırılır.

2.5. Elektronik bir H.S.A. Gerçeklemesi: ACE16K

ACE16K [21] bir CNN-UM (CNN Universal Machine) gerçeklemesidir. CNN-UM [22] kendi ana işlemcisi üzerinde paralel olarak çalışan birbirine bağlı binlerce işlemsel birimden oluşan analog ve lojik bir bilgisayardır. Temel olarak birbiriyle etkileşimli, özdeş, doğruluk gerektiren yüksek hızlı görüntü işleme işlemleri için tasarlanan 128×128 analog işlemci birimlerinden(ACE16K) oluşan bir dizi olarak tanımlanabilir. Sistem dış ortamla iletişim kurmaya olanak sağlayan birçok devreye sahiptir. Bu devreler bilgisayarla bir dijital arabirime olanak sağlarlar ve algoritmaların yüklendiği programlama bellekleri aracılığıyla yüksek algoritmik yetenekler sağlarlar. CNN-UM bir analog işlemci olmasına rağmen tamamen dijital bir ortamda çalıştırılabilir. Bu amaçla prototip analogdan dijitale ve dijitalden analoğa dönüştürücüler içerir.

CNN-UM iki alternatif yolla kullanılmak üzere tasarlanmıştır. Birinci yol, kırmığın optik modül girişinden alınan görüntülerin direkt olarak işleme sokulduğu uygulamalardır. İkinci yol, elektriksel şekilde görüntüleri alıp göndermeye olanak sağlayan dijital bir bilgisayar sisteminin kullanılarak görüntüler üzerinde işlem yapılmasıdır.

CNN-UM ın mimari yapısı Şekil 2.7 de bloklar halinde görülmektedir.



Şekil 2.7: CNN-UM ın mimari yapısı [21]

Kırmık beş fonksiyonel bloğa bölünebilir:

Birinci blok, analog işlemci çekirdeği, 128×128 özdeş hücreler dizisi(ACE16K), görüntü işleme için uzaysal sınır koşullarını tayin etmek için kullanılan kenar hücrelerinin bir kümesi ve hücre dizisinde analog ve sayısal işaretleri süren bazı ara belleklerden meydana gelir. İkinci blok, programlama bloğu, kırmık tarafından çalıştırılacak algoritmaların yüklenmesi için kullanılan iki adet 64×32 SRAM sayısal bellek, sekiz bit formatta yükleme için kullanılan altı adet 32×32 sayısal bellek, hücrelerarası etkileşimi kontrol eden analog katsayıların farklı kümeleri, bazı genel bias işaretler ve optik giriş modülü tarafından kıllanılan bazı referanslardan meydana gelmektedir. Programlama belleği bu bellek bloklarına dışardan erişim ve programlanmış değerleri analog işlemci çekirdeğine iletmek için ihtiyaç duyulan devrelere de sahiptir. Bu blok sayısal komutlar için sayısal ara bellek, hücreler arası etkileşim ve referanslar için sayısaldan analoğa dönüştürücüler ve analog ara bellekler içerir.

Üçüncü bloktan beşinci bloğa kadar olan bloklar görüntü giriş/çıkış işlemleri için tahsis edilmiştir. Genel giriş/çıkış kontrol birimi, giriş/çıkış görüntülerine erişim için gerekli olan işaretleri üretir. Bu blok, satır ve sütun adresleme işaretlerini, analogdan sayısala ve sayısaldan analoğa dönüştürücülerin kontrolünü içerir.

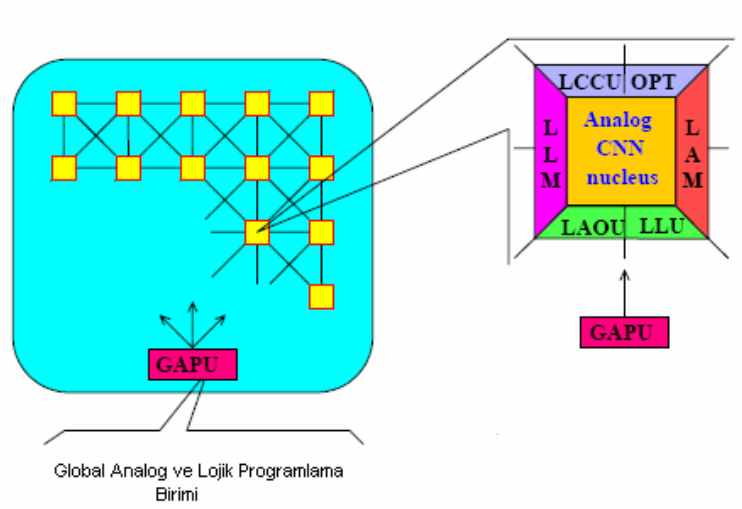
Kırmık dışarıyla iletişim için 32 bitlik çift yönlü bir veri yolu ve programlama belleği içinde değişik bloklar için bazı adres yolları kullanır.

CNN-UM yaklaşık 4 milyon transistörden meydana gelmiştir. Tablo 2.1 de CNN-UM ın bazı karakteristik özellikleri gösterilmiştir:

Tablo 2.1: CNN-UM ın karakteristik özellikleri [21]

Teknoloji	STM–0.35 µm 5M-1P
Tasarım Stili	Tamamen Özel (Analog Çekirdek) ve Standart Hücreler(Sayısal giriş/çıkış Blokları)
Paket	Seramik QFP144
Hücre Sayısı	16384(128×128)
Transistör Sayısı	3.478.170
Her Bir Hücredeki Transistör Sayısı	198
Hücre Boyutları	75.7 µm× 73.3 µm
Hücre Yoğunluğu	mm^2 de yaklaşık 180 Hücre
Durum İşaret Aralığı	[0.6, 1.4] V (Programlanabilir)
Yük İşaret Aralığı	[2.15, 2.95] V (Programlanabilir)
Zaman Sabiti	Yaklaşık 160 ns
Giriş/Çıkış Saat Frekansı	32 MHz
Besleme Gerilimi	3.3 V +/- %10
Güç Tüketimi	4 Watt tan daha az
Analog Komutların Sayısı	32
Sayısal Komutların Sayısı	64 × 64
Kalıp boyutları	11885 µm × 12230 µm

Programlanabilirliği garanti altına almak için analog işlemci çekirdeği kullanılmıştır. Arada bulunan sonuçların tekrar kullanılmasına olanak sağlamak için her hücre yerel belleklerle donatılmıştır. Yerel belleklere ek olarak her hücre yerel algılayıcılarla ve akıllı hücre analog ve lojik işlemleri gerçekleştirmek için ek devrelerle donatılabilir. İşlemci çekirdeğindeki bu hücresel yapı ve hücrelerin sahip olduğu bellekler Şekil 2.8 de görülmektedir.



Şekil 2.8: CNN-UM ın hücresel yapısı ve her bir hücrenin özellikleri [23]

Bir görüntü, algılayıcı girişi üzerinden alınabilir. Yerel bellekler her hücre içindeki analog (LAM: Local Analog Memory) ve lojik (LLM: Local Logical Memory) değerleri saklarlar. CNN-UM üzerinde 8 adet LAM bellek ve 2 adet LLM bellek vardır ve bu bellekler yardımıyla görüntü işleme işlemleri oldukça etkin bir şekilde gerçekleştirilebilmektedir. Bu bellekler sınırlı bir sayıda olsalar da, üzerinde aritmetik ve lojik işlemler kolaylıkla gerçekleştirilebilmektedir. LAM bellek üzerinde toplama, çıkarma gibi işlemler yapılabilirken, LLM bellek üzerinde lojik VE/VEYA, DEĞİL, DIŞLAMALI VE/VEYA gibi lojik işlemler kolaylıkla yapılabilmektedir. Ayrıca sonuçta elde edilen görüntüler LAM ve LLM bellekler arasında iletilebilmektedir. Bir yerel analog çıkış birimi (LAOU) ve bir yerel lojik birim (LLU) saklanan bu değerler üzerinde analog ve lojik işlemleri gerçekleştirir. Elde edilen çıkışlar her zaman bu yerel belleklerden birine gönderilir. Yerel iletişim ve kontrol birimi (LCCU) erişilen hücre ve makinenin (GAPU) merkez programlama birimi arasında iletişimi sağlar. GAPU dört fonksiyonel bloğa sahiptir. Analog program belleği (APR) analog program komutlarını ve CNN şablonlarını saklar.

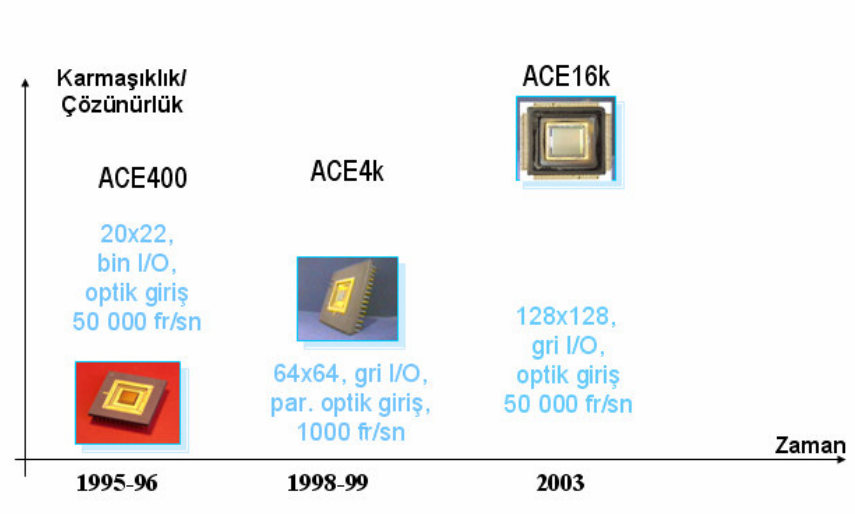
GAPU nun içindeki diğer tüm birimler hücre dizisinde işlem yapmak için gerekli kontrol kodlarını içeren lojik belleklerdir.

CNN-UM ın en önemli birimi iki boyutlu 128×128 özdeş hücreler dizisinden meydana gelen ACE16K dır. Görüntü işleme, görüntü bölütleme gibi işlemlerde kullanılan lojik ve analog işlemler ACE16K üzerinde paralel olarak çok hızlı bir şekilde gerçekleştirilebilmektedir. Bu hızı bir IBM süper bilgisayarla karşılaştıracak olursak;

$10 * 10^{12}$ işlem gücü için süper bilgisayar **6.9468 m^2** alana ve **491 kW** güce ihtiyaç duyarken 128×128 analog hücreden meydana gelen ACE16K sadece **1.4 cm^2** alana ve **4.5 W** güce ihtiyaç duymaktadır. Görüldüğü gibi aradaki fark kıyaslanamayacak kadar büyüktür. Bugün bilgisayar yardımıyla saatler süren bir işlem ACE16K ile saniyelerle gerçekleştirilebilmektedir.

ACE16K üzerinde bulunan 128×128 hücrenin her biri bir piksele karşılık gelecek şekilde maksimum 128×128 piksele sahip bir görüntü üzerinde işlem yapmak mümkündür. ACE16K sadece gri seviyeli görüntüler üzerinde işlem yapabilmektedir. Bir görüntü ACE16K ya bilgisayar üzerinden hazır olarak verilebileceği gibi optik algılayıcı yardımıyla dışarıdan direkt olarak da alınabilir.

ACE16K nın bir önceki versiyonu ACE4K dır. ACE4K 64×64 hücreye sahiptir. ACE4K da ACE16K gibi sadece gri seviyeli görüntüler üzerinde işlem yapabilmektedir. ACE4K ya da görüntüler iki farklı şekilde yüklenebilir. Bunlar, optik giriş ve arabirimler vasıtasıyla kullanılan bir bilgisayardır. Görüntüyü optik algılayıcı ile yakalama esnasında ACE4K nın ACE16K ile farkı ACE4K nın saniyede yakalayabileceği görüntü sayısının daha düşük olmasıdır. CNN teknolojisinin yıllara göre gelişimi Şekil 2.9 da gösterilmiştir.



Şekil 2.9: CNN teknolojisinin yıllara göre gelişimi [24]

Sistem tasarımı ve kırmık gerçeklemeleri arasında her zaman bir boşluk vardır. Kırmık üzerinde güçlü algoritmalar çalıştırılmak istendiğinde bazı donanımsal sınırlamalarla karşılaşılmakta ve bu sınırlamaları aşabilmek için uygulanan metotlarda basitleştirilmeye gidilmek zorunda kalınmaktadır.

2.6. Bi-i (Bio-Inspired):

Bi-i sıra dışı bir yüksek hıza sahip, kompakt, bağımsız, akıllı bir kameradır [23]. İki farklı tipte algılayıcı ile donatılabilir: Biri 1.3 mega piksel çözünürlükte görüntü yakalayan CMOS algılayıcı, diğeri ise ultra hızlı ACE16K odaksal düzlemli dizi işlemcisidir. Sistemin dışarıdan görünüşü Şekil 2.10 da görülmektedir [25].

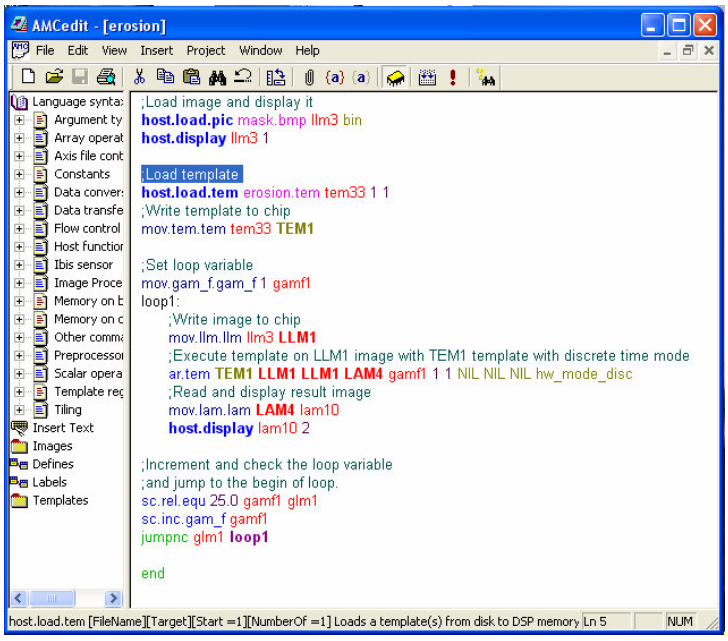
Şekil 2.10: Bi-i sisteminin dışarıdan görünüşü

Bölütleme algoritmalarının gerçeklenmesi Bi-i üzerinde gerçekleştirilmiştir. Bi-i üzerinde bulunan 128×128 hücreden meydana gelen ACE16K, görüntülerin şablonlarla konvolüsyonu, lojik işlemler, toplama çıkarma gibi birçok görüntü işleme işleminin gerçekleştirilmesinde kullanılır.

Bi-i'ın temel işlemci elemanı yüksek performanslı bir sayısal işaret işlemcisidir(DSP). Ayrıca değişik çevre birimlerini destekleyen bir iletişim işlemcisine sahiptir. Bi-i ın en önemli arabirimi 100 M-bit eternettir. Bi-i da çalıştırılacak programlar eternet üzerinden yüklenir ve bilgisayarla eternet üzerinden Bi-i a bilgi yazabilir ya da Bi-i dan bilgi okuyabiliriz.

Bi-i ı programlamanın başlıca yolları:

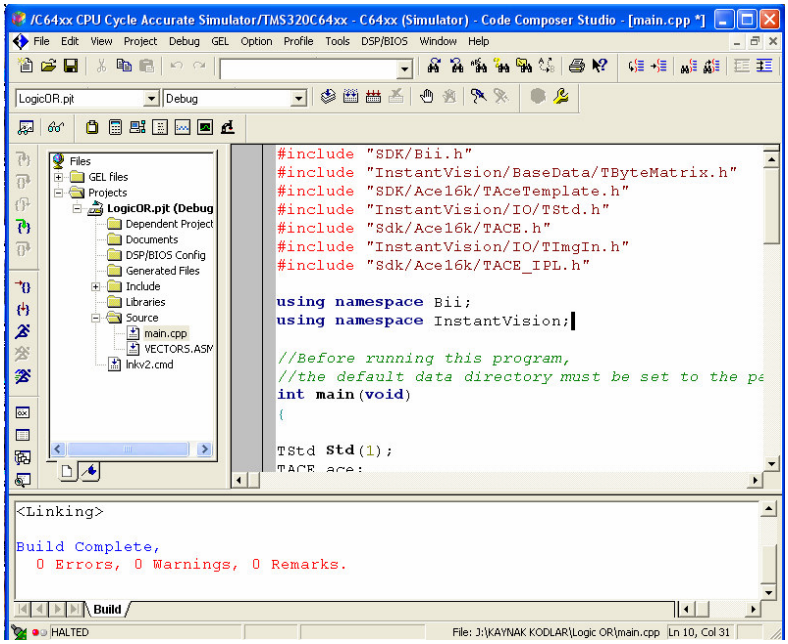
- AMC(Analogic Macro Code) Bi-i için doğal programlama dilidir. AMC dilinde yazılan kodlar ikilik tabana dönüştürülür ve Bi-i üzerinde çalıştırılır. Program geliştirme AMC derleyiciye başvuran dikkat çekici bir sentaks editörü tarafından desteklenir. AMC editörü ve AMC derleyici Bi-i yazılım paketinin bir parçası olarak yüklenirler. AMC büyük projeler için tasarlanmamıştır ama küçük uygulamalarda oldukça etkilidir. AMC ile yazılım geliştirme uygulamasına bir örnek Şekil 2.11 de verilmiştir.



Şekil 2.11: AMC programlama dilinde ACE16K için kod üretme

- Bi-i SDK(Yazılım Geliştirme Aracı) [26] ile birlikte Instant Vision kütüphaneleri

Bi-i uygulamalarını geliştirmek için kullanılan C++ programlama kütüphanelerinin bir kümesidir. Bu kütüphaneler geliştirme birimi Code Composer Studio ile birlikte DSP için kullanılabilir. Buradaki çalışmalarda da kullanılan Code Composer Studio ile birlikte bir yazılım geliştirme örneği Şekil 2.12 de görülmektedir.



Şekil 2.12: Code Composer Studio ile C++ dilinde ACE16K için kod üretme

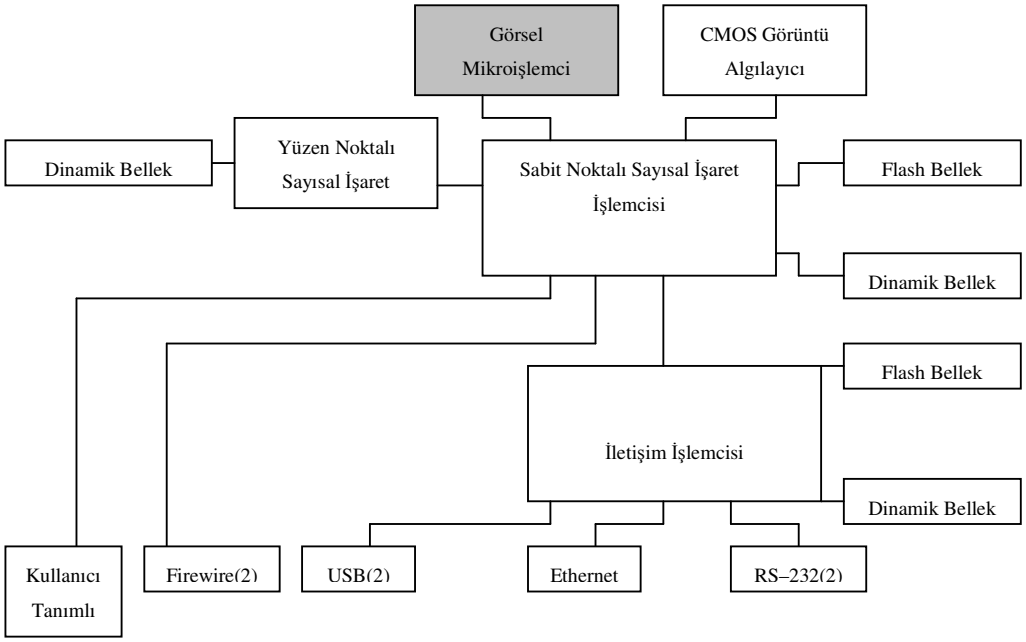
Bi-i ın v1, v2, v301 olarak temsil edilen bazı versiyonları vardır. Ancak tüm sistemler için temel mimari aynıdır.

Bi-i ın merkez elemanı, Bi-i programlarını çalıştıran yüksek performanslı bir sayısal işaret işlemcisidir. Diğer tüm önemli elemanlar buraya bağlıdır. DSP önyüklemede kullanılan bir flash belleğe ve program çalışması sırasında kullanılan bir dinamik belleğe sahiptir. Bi-i v1 250 MHZ de çalışan TMS320C6202 DSP ye, Bi-i v2 600 MHZ de çalışan TMS320C6415 DSP ye ve Bi-i v301 600 MHZ – 1 GHZ de çalışan TMS320C6415 DSP ye sahiptir.

Görüntüler farklı kaynaklardan yüklenebilir. Bu kaynaklardan birisi 1.3 mega piksel çözünürlükte IBIS5 CMOS algılayıcıdır. Diğer görüntü kaynağı ise 16k piksel formatındaki bir ACE16K görsel mikroişlemcidir. ACE16K DSP ye görüntü

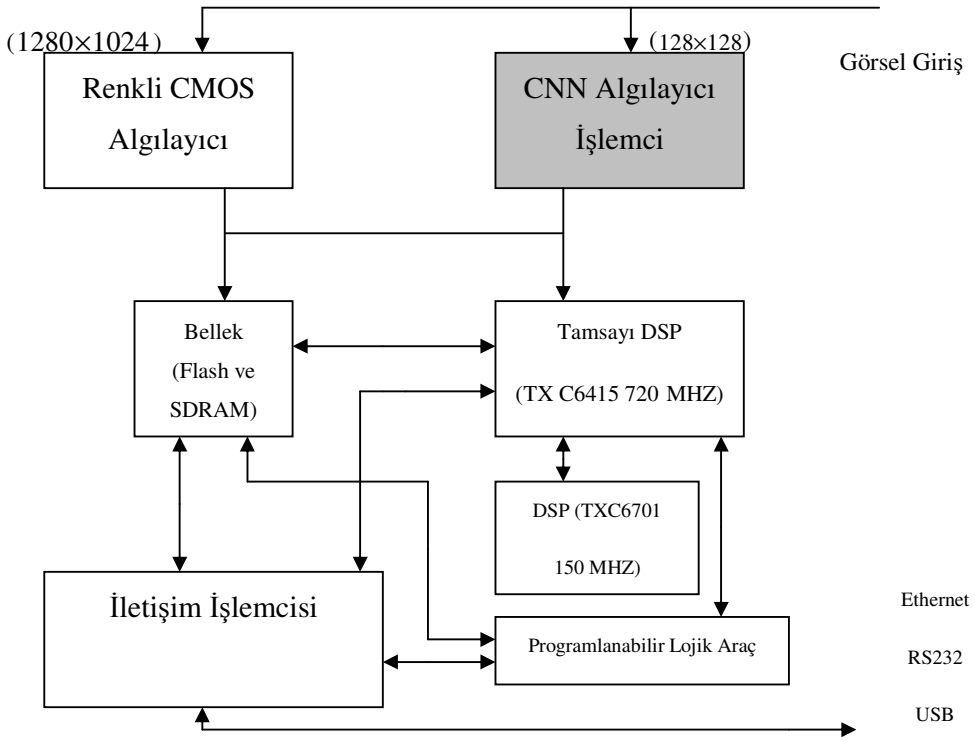
işlemenin sonucunu sağlayan sayısal olarak kontrol edilen bir dizi işlemcisidir. ACE16K hızlı görüntü işleme uygulamaları için tasarlanmıştır.

ACE16K(Bi-i v1) in donanım yapısı Şekil 2.13 de gösterilmiştir.



Şekil 2.13: Bi-i sisteminin mimari yapısı

Bi-i temelde CNN tipi mikroişlemci (ACE16K), DSP tipi mikroişlemcilere ve bunların farklı işlem adımları arasında geri besleme ve otomatik kontrol mekanizmalarına sahip algoritmik yapılarından oluşmaktadır. Bi-i yapısının daha genel bir gösterimi Şekil 2.14 te görülmektedir.



Şekil 2.14: Bi-i sistem mimarisinin temel bloklar halinde gösterimi [23]

3. BÖLÜTLEME ALGORİTMALARI:

Bu bölümde literatürde mevcut olan Early [27], Roska [28], Grassi [29] gibi bölütleme algoritmalarından Early ve Roska bölütleme algoritmaları üzerinde durulacak ve algoritmaların bazı görüntüler üzerinde uygulaması yapılacaktır. Bir sonraki adımda literatürdeki algoritmalardan yola çıkarak yeni bir bölütleme algoritmasının tasarımı yapılacak ve uygulamasından bahsedilecektir.

3.1. EARLY BÖLÜTLEME ALGORİTMASI:

Nesne tabanlı görüntü ve video işleme, bilgisayar dünyasının son dönemini temsil etmektedir. Görüntüler basit bir şekilde pikseller bloğu veya bir pikseller kümesi olarak adreslenemezken nesnelerin bir kümesi olarak adreslenebilir. Bu yaklaşım, otomatik gözetimden video kodlamaya kadar birçok uygulama alanına yeni çözümler sağlamaktadır. Video işlemede yeni bir interaktif boyut nesne tabanlı yaklaşımla elde edilmektedir. Nesnelerin dizi boyunca zamanda tutarlılığı, etkin bir şekilde oranların sıkıştırılması ve hata iyileştirmede kullanılır.

MPEG-4 standardı blok tabanlı, nesne yönelimli bir video kodlama planı tanımlar. Bu, görüntünün keyfi olarak biçimlendirilmiş nesnelerin bir kümesi olarak düzenlenebileceği anlamına gelmektedir. Her nesne, MPEG-2 standardında tanımlanana benzer şekilde, blok tabanlı bir teknik kullanılarak kodlanır. MPEG-4 standardı nesnelerin tanımlanması gereken yolu belirtmediği için nesneleri oluşturmak için kullanılan görüntü bölütleme tekniğinin seçimi konusunda herhangi bir kısıtlama mevcut değildir. MPEG-4 açık bir şekilde her nesnenin nasıl kodlanacağını tanımlar. MPEG-4 te her nesne blok tabanlı bir teknik aracılığıyla kodlanır. Bu, nesnenin 16×16 bloklara bölündüğü anlamına gelmektedir. Bu bloklar üzerinde bazı işlemler gerçekleştirilmektedir. Kodlama işlemi sırasında en maliyetli iş hareket yorumlamadır (toplam işlemin %60'ı). Bu işi kosinüs dönüşümü ve ölçme izlemektedir. Eğer görüntü ve video işlemedeki nesne kavramının tanıtımı yeni uygulama ve daha güçlü araçlara yol açarsa, bu aynı zamanda nesnelerin gerçek

zamanlı, otomatik üretim ihtiyacını arttırır. Bu işlem uzay-zamansal görüntü bölütleme olarak adlandırılır [30].

Literatürde birçok görüntü bölütleme algoritması yer almaktadır. Bir video dizisi bölütlenmek istendiğinde, dizinin uzay-zamansal bölütlemesini oluşturmak için uzaysal bilgi zaman bilgisiyle birlikte kullanılır. Uzay-zamansal görüntü bölütleme basit bir iş değildir. Bu iş genellikle, oldukça yüklü hesaplama gücü ve görüntü karakteristiklerinin geniş kapsamlı analizini gerektirir. Görüntü bölütlemede kullanılan dış hat belirleme ve hareket yorumlama gibi oldukça maliyetli işler, tam paralel bir yolla oldukça etkin bir şekilde yerine getirilebilir. Bu işler CNN-UM devre kırmığı gibi geleceğin mümkün olan donanım platformlarına uygulanabilir.

Tam paralel bir yapı CNN-UM mimarisindeki gerçek zamanlı video dizilerinin bölütlenmesine yardımcı olmaktadır. Birçok görüntü işleme konusu paralel mimaride etkin bir şekilde gerçekleştirilebilmektedir. Önerilen strateji aşağıdaki metotlara dayanmaktadır:

- Gradyan tabanlı dış hat belirleme ve morfoloji tabanlı bölütleme
- İstatistiksel değişim tespiti
- Anizotropik difüzyon ve ölçek uzay modelleri
- MRF(Markov Random Field) bölütleme modelleri

İlk metot, gerçek zamanda yapının üst üste bir bölütlemesini sağlamayı amaçlar. Elde ettiğimiz sonuçlar bölgelerin sayısını minimize etmese bile görüntüdeki grupların nasıl yeniden gruplandırılacağı kararının hem uzaysal hem de zamansal bilgiye bağlı olduğu daha sonraki işlemler için iyi bir başlangıç noktasıdır.

Hareketli nesnelerin bölütlenmesi işi standart iş istasyonları üzerinde iyi düzenlenmiş hiyerarşik programlama yapıları ve optimizasyon metotlarını kullanarak güçlükle çözülmektedir.

Hareketli nesne bölütlemesinin bağlantı analizi bazı global optimizasyonlara ihtiyaç duymaktadır. Hiyerarşik bir hesaplama modeli içinde grafik tabanlı tekniklerle çözülebilir. Buradaki tam paralel modelde benzer etkiler MRF(Markov Random Field) bölütlemesini kullanarak elde edilebilir. MRF, görüntü bölütleme için global ya da yarı-global optimum önermektedir ve hareket bölütlemesinde zaten

kullanılmaktadır. Buradaki durumda yapılması gereken iş optik akış alanını yüksek doğrulukla belirlemek değil, hareket eden farklı nesneleri veya bu nesnelerin parçalarını tespit etmektir. Bu esnada aşağıdaki işlerle karşı karşıya kalmaktayız [30]:

- Hareket alanının yorumlanması
- Hareketin ve orijinal görüntülerin bölütlenmesi
- Artıkların ve diğer yapıların çıkarılması
- Hareket ve uzaysal bilgiyi hareket eden nesneleri bölütlemek için birlikte elde etmek

Buradaki yaklaşım süresince asıl amaç, bir paralel modelde hareket bölütlemesinde kullanılan operatörlerin yapılabilirliğidir. Mümkün olan çözümler özel amaçlı VLSI mimarileriyle çözülebilen alt işlere kısıtlanır. Bu operatörlerin sağlaması gereken koşullar aşağıdaki şekilde sıralanabilir:

o Parametreler esasen yerel bilgide bulunmalı ve yerel olarak depolanmalıdır, komşuluk ilişkileri yerel komşulukta 1 veya 2 yarıçapla sınırlandırılmalıdır.

o Fonksiyonlar esasen yerel parametrelerle işlem yapmalıdır ve bu fonksiyonlar lojik fonksiyonlar, karşılaştırmalar, çarpma, toplama v.s. gibi basit fonksiyonlar olmalıdır.

o İşlem yapılan elemanlar ve işlem yapılan elemanların sayısı arasındaki veri akışı minimize edilmelidir.

Optik Akış Alanı Kestirimi:

Tam paralel mimarinin temel avantajlarından biri bazı blok veya nesneler üzerindeki parametreler yerine yoğun(piksel piksel) tanımıyla çalışmaktır. İyi bölütlenmiş nesneler ya da nesne parçaları elde etmeyi hedefleriz fakat bu bir amaç veya yarı amaçtır ve başlangıç koşulu değildir. Sonuç olarak buradaki model yoğun hareket alanını (DMF) hesaplar. İki değişik yaklaşım önerilebilir:

- Korelasyon tekniği: Bu yoğun hareket alanı elde etmek için basit ve oldukça paralel bir çözümdür.

▪ Gradyan tabanlı üst üste bölütleme: Geri beslemeli kalite ölçümüyle birlikte oldukça paralel bir hareket kestirimi ve düşük seviyeli bölütlemedir.

Hareketli Nesnelerin Bölütlenmesi:

- Artık temizleme: Bir kez DMF oluşturulduğunda sıradaki iş görüntüyü gürültü ve diğer yapılardan temizlemektir. Öncelikle oluşum ve artık etkileri hareket eden nesnelerin hareket alanlarından temizlenmelidir.

- Anizotropik yayınım: Hareket bilgisini ve uzaysal bilgiyi birlikte planlarken yapıdaki nesnelerin temel özelliklerinin kenar haritasına ihtiyacımız vardır. Bu adımı görüntü üzerinde çok ölçekli bir analiz içinde anizotropik yayınımla kontrol edebiliriz.

- Markov Random Alan Bölütlemesi (MRF): MRF, görüntüyü piksel piksel sınıflandıran bazı bölütleme problemlerinde global optimum verebilir. MRF tam paralel stokastik optimizasyon kullanarak daha yüksek seviyeli nesne tanımını elde etmeye yardımcı olabilir.

- Morfoloji: küçük yapıları kaldırmak ve temizlemek için, boşlukları doldurmak için, kenarları birleştirmek için, net iskeletler yapmak için uygun birçok adım vardır.

Sonuç olarak algoritmanın temel adımları aşağıdaki gibi olacaktır:

1. DMF'i, x ve y'yi ya da hız ve doğrultu bileşenlerini oluşturmak

2. Gradyan tabanlı bölütlemede üst üste bölütlenmiş alanlar elde ederken korelasyon tekniklerinde sınıflandırmak için işlenmemiş bir DMF elde ederiz.

3. DMF görüntüsü üzerinde artık temizleme ve morfolojik temizleme

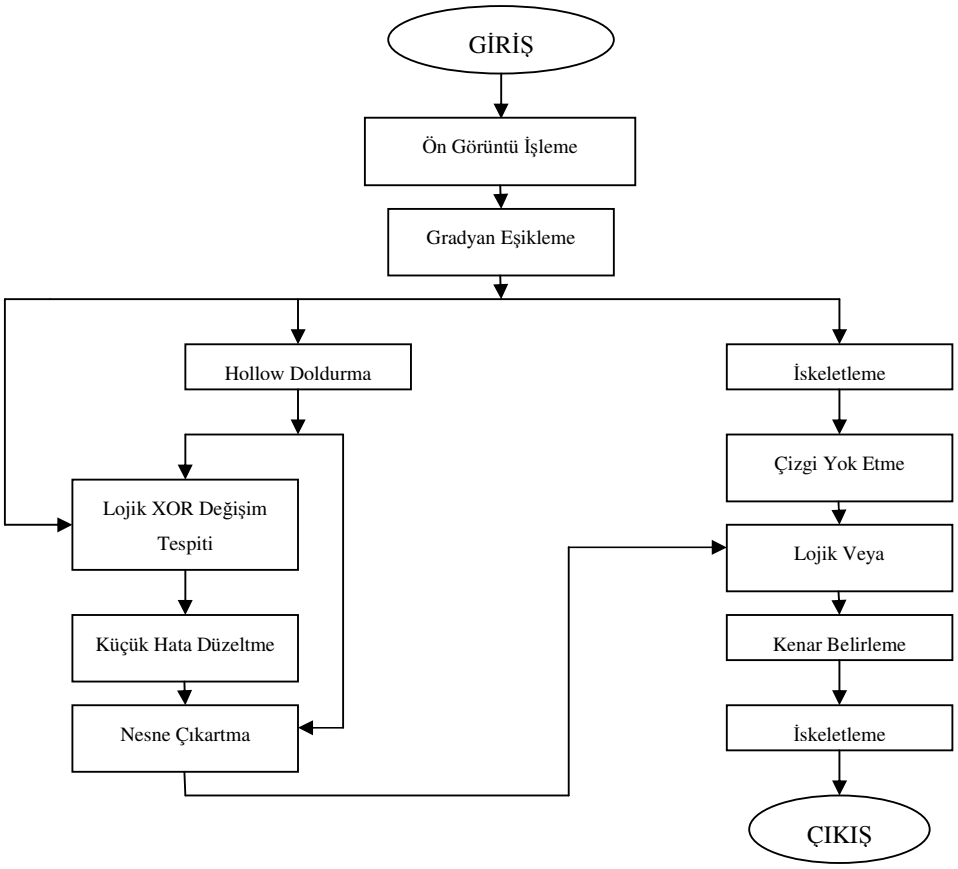
4. MRF kullanarak daha büyük tutarlı hareket alanları elde etmek için yükseltilmiş DMF görüntüsünün daha yüksek seviyeli bölütlemesi

5. Hareketlerinden bağımsız olarak temel parçaların birleştirilmiş kenarlarını elde etmek için uzaysal görüntünün kenar tespiti ve anizotropik yayınımı

6. Temel kenar haritasını ve kesin bir bölütlemeyle hareketli parçaları seçmek için DMF'nin MRF'sini kullanarak morfoloji

NOT: Burada yapılan uygulamalarda hareketli bir görüntü kullanılmadığından görüntünün hız bileşenleri ile işlem yapılmamıştır ve bu değerler hesaplanmamıştır.

Bu algoritma, parlaklık ve kontrast bilgisine dayanan uzaysal bölge koşullarındaki görüntünün bir tanımını sağlar [31]. Bu algoritma MPEG-4 standardının hükümlerini kabul etmektedir. Bir ön görüntü işleme safhasından sonra, dış hatların belirlenmesi hassasiyeti eşik değerine bağlı olan gradyan şablon işlemi ile gerçekleştirilir. Eşik değeri ne kadar küçük seçilirse, görüntüden o kadar fazla detay çıkartılır. Küçük gürültü etkileri sonradan giderilir ve aynı resmin farklı parçaları birleştirilir.



Şekil 3.1: Early bölütleme algoritmasının akış diyagramı

Şekil 3.1'in sol tarafındaki ilk kısım gradyan eşikleme işlemi ile görüntüden çıkarılamayacak nesneleri tespit eder. Gradyan tespiti, görüntüde gradyanın sıklıkla değiştiği bölümlerde üst üste bölütlemeye neden olur. Bu işlem sonucunda belirsiz dış hat çizgilerinde bazı benekler oluşabilir bu yüzden dış hat çizgilerini düzeltmek için bir boşluk doldurma işlemi uygulanır(Hollow şablonu). Bu işlemden sonra bir XOR işlemi değişiklikleri tespit eder, başka bir deyişle belirsiz dış hat çizgilerinin

yerini bulur. Bir küçük hata düzeltme şablonu, istenmeyen noktaları siler ve seçilen nesneyi çıkarma şablonu için bir maske oluşturur. Seçilen nesne çıkarma şablonu dış hattın belirsizliğini gidermek için Hollow doldurmanın sonucunu kullanarak maskelenmiş nesneyi düzeltir.

Şekil 3.1'in sağ tarafında görülen ikinci dal iskeletleme algoritmasını kullanarak iyi belirlenmiş dış hatları tespit eder ve anlamsız çizgileri bir çizgi yok etme şablonu ile ortadan kaldırır. Son olarak iki akış şemasından gelen belirlenmiş nesneler birbirine eklenir ve ikilik kenar belirleme ve yeni bir iskeletleme işlemi sonucu verir. Bu algoritma ACE4K dan farklı bir davranışı olan CNN-UM in karakteristik özellikleri göz önünde bulundurularak tasarlanmıştır.

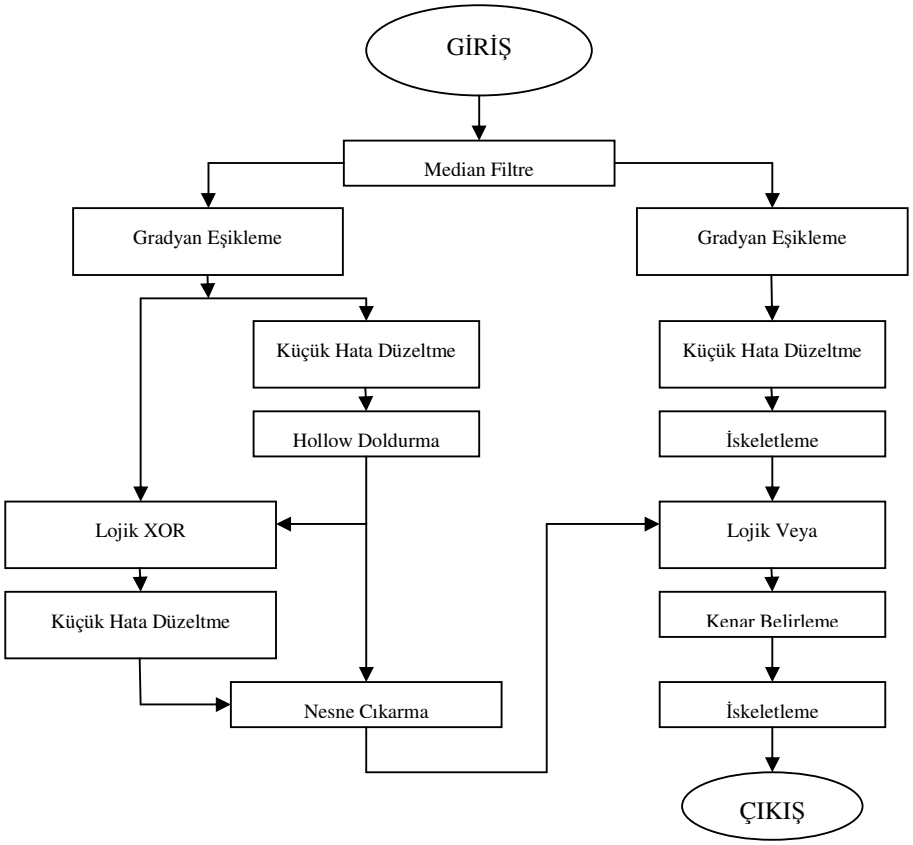
Kırmık ortamında bu işlemleri gerçekleştirebilmek için ilgili şablonlar için formülün yeniden düzenlenmesine ihtiyaç vardır. Bunun nedeni CNN simülatörü ve donanımsal VLSI gerçeklemesi arasındaki farktır. Aslında:

- ACExxK kırmıkları, sıklıkla uzayda değişen lineer olmayan şablonlar ile işlem yapabilmeye izin vermez. Bu yüzden kırmık üzerinde çalışabilmek için bu şablonları lineer olanları gibi tasarlamak için uzun bir deneme süreci geçirilir.

- Şablon kırmık üzerinde sadece belirli bir aralıkta çalışır. Bu aralık sırasıyla şablon sabitleri için $\left|-3,+3\right|$ ve bias terimleri için $\left|-6,+6\right|$ dır. Ayrıca kırmık sırasıyla gri seviyeli görüntüleri 7 bit, şablon terimlerini 8 bit çözünürlükle kullanır. Açıkça görülüyor ki, basit bir sabit ölçekleme başarılı sonuçlar vermediğinden bu sorun basit değildir. O nedenle, aralık dışı şablon işlemleri, yeni şablonlara veya yeni şablon tabanlı alt programlara orijinal aralıktaki şablon sonuçlarına göre oluşan hatayı minimize etmek için yol göstererek yeniden tasarlanır [32].

- Her şablonun sağlamlığını arttırmak ve tüm sonuçları geliştirmek için tüm ES algoritması üzerinde gerçekleştirilen ileriki bir konu deneye dayalı optimizasyondur.

O nedenle, kırmık davranışının iyice karakterize edilmesinden sonra kırmığa adapte edebilmek için orijinal ES [27] algoritması üzerinde bazı modifikasyonlar ve formülde yeni düzenlemeler yapılır. Kırmık üzerinde doğru bir şekilde çalışabilmek için yeniden düzenlenen ES algoritmasının yeni versiyonu Şekil 3.2 de görülmektedir.



Şekil 3.2: Modifiye edilmiş Early bölütleme algoritması

Bu yeni versiyondaki temel değişiklikler aşağıdaki gibidir:

- *Görüntü ön işleme işlemi:* Bu işlem tuz ve biber gürültü filtresi aracılığıyla ilk gürültüyü ortadan kaldırmak için uygulanır. Görüntünün kontrastını arttırmak için görüntü ön işleme kısmına bir alçak geçiren filtre işlemi eklenebilir.

- *Gradyan Eşikleme:* bu işlem akış şemasının her iki dalında birden uygulanır. Ancak hassasiyeti arttırmak için akış şemasının her dalında farklı eşik değerleriyle uygulanır.

- *Çizgi Yok Etme işlemi:* Bu versiyonda mevcut değildir. Bu işlemin yerine küçük hata düzeltme işlemi kullanılmıştır.

- *İki yeni Küçük Hata Düzeltme İşlemi:* Bu adımlar işlemimizi gürültü etkilerinden korumak için algoritmaya eklenmiştir.

İlerleyen alt bölümlerde bu işlemler detaylı olarak açıklanacaktır.

3.1.1. Median Filtre:

Görüntü akış kaynağının bir kamera olduğunu göz önünde bulundursak, istenmeyen yeni nesnelerden dolayı bölütleme işlemini başarısız kılan gürültülü bir görüntüye sahip olmak mümkündür. Gürültülü ya da gürültüsüz tüm görüntüleri bir ön filtreleme işleminden geçirmek zorundayız. Median filtre en önemli filtreleme işlemlerinden biridir. Median filtre 1970 yılında Tukey [33] tarafından tanıtılan nonlineer bir filtredir. Median filtreler gürültüyü etkili bir biçimde azaltır ve sinyalin kenarlarını iyi korur. Kenar koruma, görsel algılamanın doğasından dolayı görüntü işlemede temel nitelik taşır. Median filtrenin orijinal versiyonunun çıkışı komşu piksellerin orta değerinde yer alan bir pikseldir. Bu işlem tüm görüntüye uygulanır ve gürültülü pikseller üzerinde seçici değildir. Bundan dolayı median filtre gürültüyü iyi bir şekilde giderir, fakat görüntünün detaylarını yok eder. Bundan başka, çalışan bir median filtrenin performansı zamansal olarak kör olduğundan sınırlanır. Gözlem penceresindeki yerlerine rağmen tüm gözlem örnekleri eşit şekilde davranır. Fakat median filtre birçok optimal özelliklere sahip güçlü bir kestiricidir. CNN e uyarlanan median filtre klasik olarak nonlineer bir şablona ihtiyaç duyar.

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} d & d & d \\ d & 0 & d \\ d & d & d \end{bmatrix} \qquad \mathrm{Z} = 0 \qquad \textbf{(3.1)}$$

$$d = -Q * isaret(v_{xij}(t) - v_{ukl}(t)) \qquad Q = 1/2$$

Olmak üzere v_{xij} ve v_{ukl} sırasıyla iç hücrenin ve komşu hücrelerin girişinin durumlarıdır. Bu şablon nonlineer olduğundan kırmık üzerinde çalışmayacaktır.

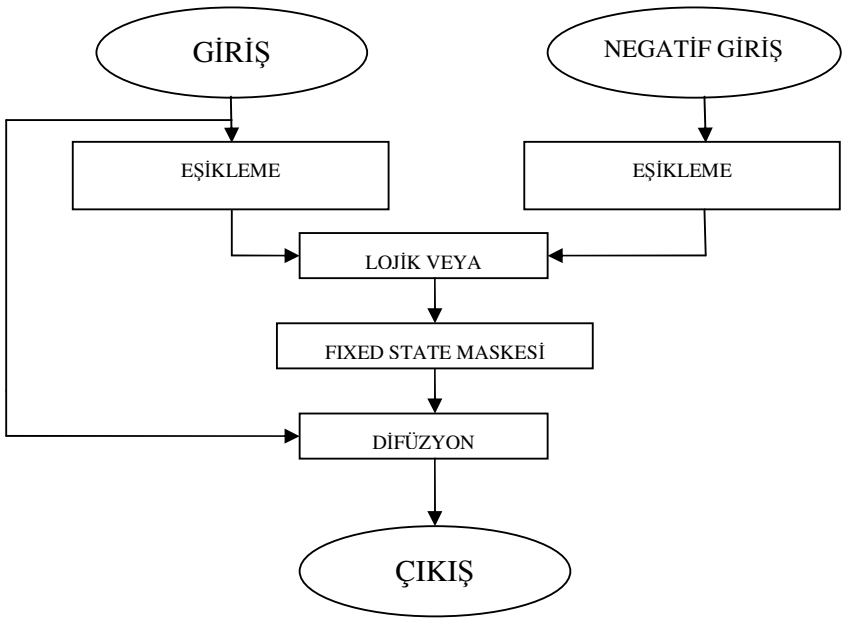
Bu sınırlamaları ve filtrenin asıl amacının tuz ve biber gürültüsünü ortadan kaldırmak olduğunu dikkate alarak, görüntünün diğer tüm parçalarını korurken yeni bir şablon algoritması tanıtılır ve direkt olarak ACE4K kırmığına uyarlanır. Bu yeni uyarlama sadece tuz ve biber gürültüsünü ortadan kaldırıp görüntü detaylarını bozmadığından kısmen istenilen hedefe ulaştırır. Bu algoritma sadece gürültülü pikselleri bulur ve bu pikselleri yerel orta değerleriyle yer değiştirir. Detaylı bir şekilde bakacak olursak, bu algoritma impuls gürültüsünden etkilenen piksellerin yerini belirlemek için giriş görüntüsü olarak orijinal görüntü ve orijinal görüntünün negatifini kullanır. Çıkarma işlemi her iki görüntüyü de eşikleme alt işlemine sokarak yapılır.

Eşikleme işlemi için belirlenen şablon aşağıdaki gibidir:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Z = -1 \tag{3.2}$$

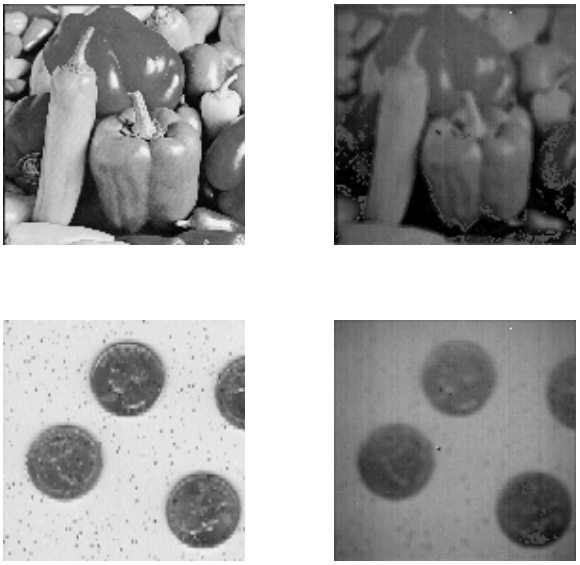
Sonraki işlem olarak eşiklenmiş bu iki görüntü arasındaki lojik veya işlemi orijnal karedeki tamamen siyah ya da tamamen beyaz piksellere karşılık gelen impuls gürültüsüne sahip tüm pikselleri tespit eder.

Bir sonraki adım tüm gürültülü pikselleri içeren elde edilmiş görüntünün lojik değil işlemine sokulmasıdır. Bu görüntü tüm gürültülü ve beyaz pikselleri içeren neredeyse tamamen siyah bir görüntüdür. Son adım ise yeni elde edilen görüntüyü ortalama şablonuyla konvolüsyona sokarak fixed state maskesini elde etmek ve elde edilen bu maskeyi orijinal görüntü üzerinde difüzyon işlemiyle uygulamaktır. Tüm bu adımları gösteren bir şema Şekil 3.3 te verilmiştir. Bu durumda klasik ortalama şablonu sadece tespit edilen gürültülü pikseller üzerinde uygulanmaktadır.



Şekil 3.3: Median filtre için akış diyagramı

Median filtre yerine alçak geçiren filtre de kullanılabilir. Alçak geçiren filtre işlemi Bi-i kütüphanesinde hazır olarak yer aldığından ACE16K kırmığı üzerinde çok daha kolay olarak uygulanabilmektedir. Şekil 3.4 te iki farklı görüntü için elde edilen median filtre çıkışları görülmektedir.



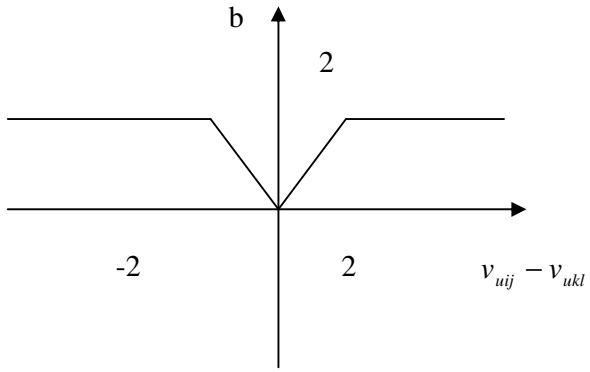
Şekil 3.4: İki farklı görüntü üzerinde median filtre uygulaması

3.1.2. Gradyan Eşikleme:

Gradyan eşikleme işlemi, resimde alanın gradyanının verilen bir eşik değerinden daha büyük olduğu bölgelerin bulunmasını sağlar. Simülasyon versiyonunda bu şablon aşağıdaki yapıdaki gibi nonlineerdir:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} b & b & b \\ b & 0 & b \\ b & b & b \end{bmatrix} \qquad \text{Z = eşik} \qquad \textbf{(3.3)}$$

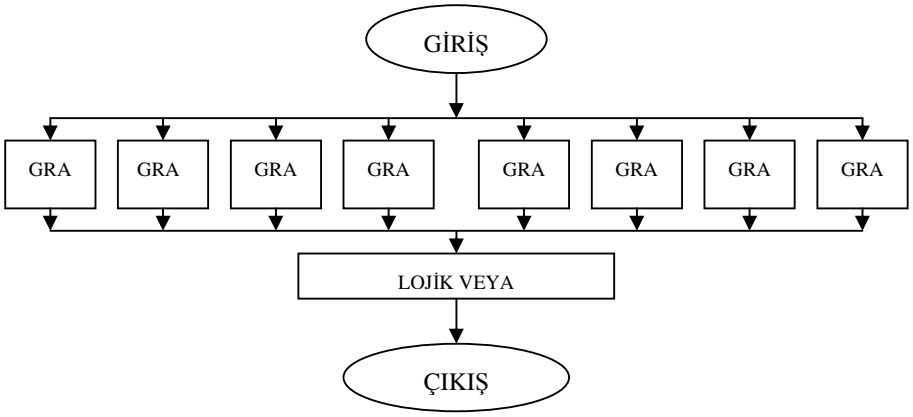
Burada eşik verilen bir eşik değeri ve *b* Şekil 3.5'te resmedildiği gibidir.



Şekil 3.5: Eşik değeri

Bu şablon işleminin CNN simülasyonu sonucunda çıkış görüntüsü, siyah piksellerin, giriş görüntüsünde gradyanın belirlenen eşik değerinden daha büyük olduğu bölgeleri gösterdiği ikilik bir görüntü olacaktır.

Bu şablonu lineer şablonların bir kümesine dönüştürmek için sekiz lineer şablonu kullanan, gradyanın tüm yönlerde paralel olarak değerlendirildiği bir algoritma oluşturuldu. ACE4K kırmığı için oluşturulan algoritma Şekil 3.6 daki gibidir.



Şekil 3.6: ACE4K için Gradyan eşikleme işleminin akış diyagramı

Örneğin kuzeybatı doğrultusu için gradyan işlemi basit olarak aşağıdaki şekildeki gibi bir şablon ile gerçekleştirilir:

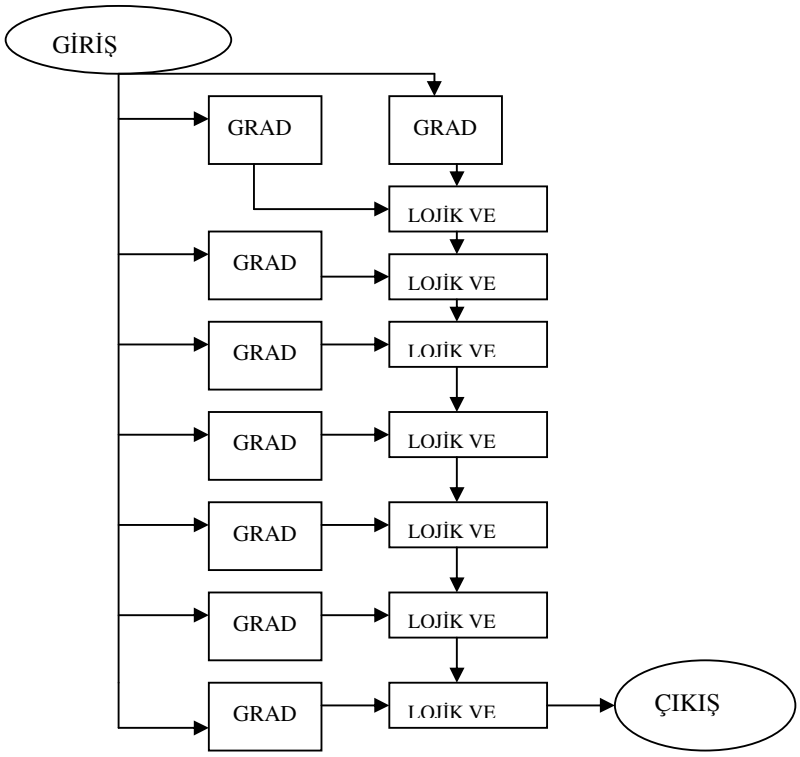
$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} -3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Z = 0 \qquad \textbf{(3.4)}$$

Kuzey doğrultusu için ise aşağıdaki şekildeki gibi olacaktır:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & -3 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Z = 0 \qquad \textbf{(3.5)}$$

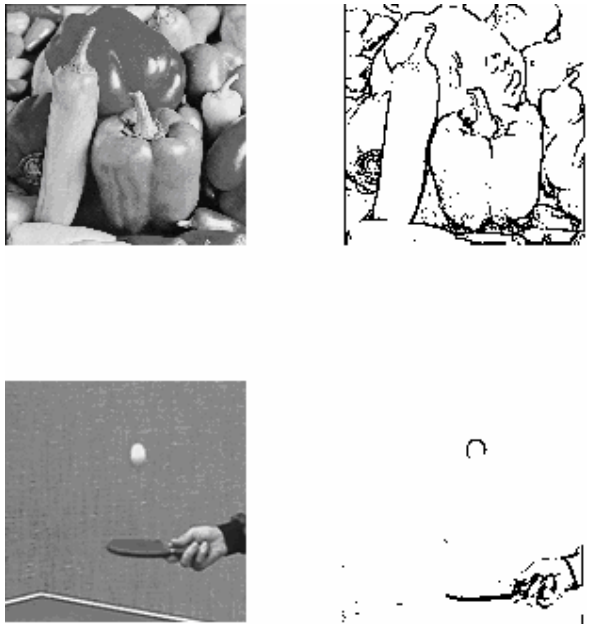
Kalan diğer altı şablon *B* matrisindeki sıfır dışı katsayının yerini matrisin merkez elemanı etrafında saat yönünde değiştirerek elde edilebilir. *A* matrisi tüm yönler için aynı kalacaktır. Burada bias bir eşik değeri gibi kullanılır. Sıfır bias durumu tüm pozitif eşitsizlikleri seçer. Burada iki farklı eşik değeri kullanılmaktadır. ACE4K kırmığı için bu değerler algoritmanın sol bölgesi için -1.2 ve sağ bölgesi için +0.5 tir.

Bu işlemi ACE16K kırmığı üzerine adapte etmek için şablon elemanlarının değiştirilmesine gerek yoktur. Ayrıca ACE16K nın yeni özelliklerinin avantajlarından yararlanabilmek için gradyan eşikleme algoritmasında yeniden düzenleme yapılmalıdır. Özellikle şablon ve lojik işlemler daha fazla LAM bellek elde etmek amacıyla birbirinin peşi sıra yapılmaktadır. Aslında ACE16K ACE4K ya oranla 4 tane daha fazla LAM belleğe sahiptir. Gradyan eşikleme algoritmasının ACE16K için düzenlenen yeni versiyonu Şekil 3.7 de görülmektedir.



Şekil 3.7: ACE16K için gradyan eşikleme işleminin akış diyagramı

ACE4K gerçeklemesinde olduğu gibi buradaki iki farklı eşik değeri de algoritmanın sol tarafı için -2.8 ve algoritmanın sağ tarafı için -2.4 olarak ayarlandı. Giriş, başlangıç durumuna eşit alınarak median filtreleme sonucunda elde edilen görüntü giriş olarak uygulandı. Gradyan eşikleme sonucunda elde edilen iki farklı görüntü aşağıdaki Şekil 3.8 de gösterilmiştir.



Şekil 3.8: Gradyan eşikleme işleminin iki farklı görüntüye uygulanması

3.1.3. Hollow Doldurma:

Bu işlem giriş görüntüsünün içbükey bölgesini doldurur. Bu işlem için modifiye edilmiş şablon ACE4K kırmığı için aşağıdaki gibidir:

$$A = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & -3 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2.5 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \text{Z} = 1 \qquad \textbf{(3.6)}$$

Bu şablon tüm nesneleri sadece yatay, dikey ve diagonal kenarlarla birlikte birleşik siyah dışbükey çokgenlere dönüştürür.

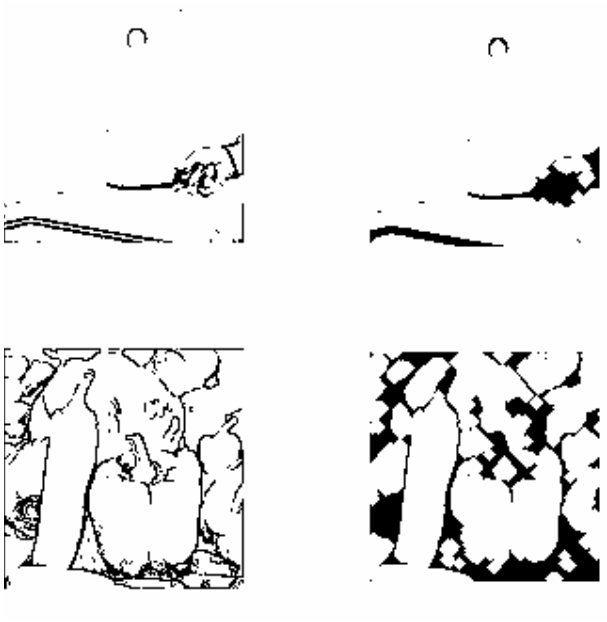
İçbükey bölgelerin doldurulması işlemini ACE16K kırmığına adapte etmek için iki küçük ayarlama yapılmalıdır: birincisi *ofset bias maskesi*nin kullanılmasıdır. Bu maske bir bias görüntünün karmaşık bir görüntüyle toplanmasına olanak sağlamaktadır. Bu işlemin avantajları ikilik işlemlerle ilgilidir. Yapılan ikinci ayarlama şablon modifikasyonudur. Yeni şablon aşağıdaki gibi olacaktır:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Z = 2 \tag{3.7}$$

Ancak belirtilen şablonlar sistem üzerinde iyi sonuç vermediğinden aşağıdaki şablonlar kullanıldı. Burada kullanılan şablonlar aşağıdaki şekildeki gibidir:

$$A = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 2 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Z = -3.4 \tag{3.8}$$

Hollow doldurma işlemi için sol kısımdaki gradyan eşiklemeden elde edilen görüntü, giriş başlangıç durumuna eşit kabul edilerek işlem yapıldı. Hollow doldurma işlemi şablonlarla yapılabileceği gibi Instant Vision kütüphanesindeki kapatma (Closing4) ya da HoleFiller fonksiyonları kullanılarak da yapılabilir. Kapatma fonksiyonunun kullanılması durumunda üç iterasyon adımı uygulanmaktadır. Hollow doldurma sonucunda elde edilen iki farklı görüntü Şekil 3.9 da gösterilmiştir:



Şekil 3.9: Hollow doldurma işleminin iki farklı görüntüye uygulanması

3.1.4. İskeletleme:

İskeletleme bir görüntüden siyah beyaz bir nesnenin iskeletini tespit eden bir algoritmadır. Görüntüde belirlenen kenarların kalınlığını bir piksele düşürür. Bu iskelet her biri bir ikilik morfolojik işlem gerçekleştiren sekiz şablona dayanan bir CNN algoritmasıyla elde edilir. Bu şablonlar dairesel olarak uygulanır. Alt program matrisinin simülasyon versiyonun bazı katsayıları limitlerin üzerinde olabilir. Örneğin aşağıdaki ilk dört şablon bu durumdadır:

$$A_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B_1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad Z_1 = -1$$

$$A_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B_2 = \begin{bmatrix} 2 & 2 & 2 \\ 0 & 9 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad Z_2 = -2$$

$$A_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B_3 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 5 & 1 \\ 0 & -1 & 0 \end{bmatrix} \qquad Z_3 = -1 \qquad \textbf{(3.9)}$$

$$A_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B_4 = \begin{bmatrix} -1 & 0 & 2 \\ -2 & 9 & 2 \\ -1 & 0 & 2 \end{bmatrix} \qquad Z_4 = -2$$

Buradan A şablonları aynı kalırken, tek (çift) numaralı B şablonlarının saat yönünde 90° döndürülmüş olduğu görülür. Kalan diğer dört şablon $(A_5, ...A_8)$ aynı şekilde bulunabilir. B şablonunun merkez elemanın $|-3,+3|$ aralığının dışında olduğu açıkça görülmektedir. Bu değeri uygun bir aralığa sokmak için basit bir yeniden ölçekleme her zaman en iyi seçim olmamaktadır. Yeniden düzenlenen tek şablonlar aşağıda gösterilmiştir:

$$A_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B_1 = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 3 & -0.5 \\ 0 & -0.5 & 0 \end{bmatrix} \qquad Z_1 = 0$$

$$A_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B_3 = \begin{bmatrix} 0 & 0.5 & 0.5 \\ -0.5 & 3 & 0.5 \\ 0 & -0.5 & 0 \end{bmatrix} \qquad Z_3 = 0 \qquad \textbf{(3.10)}$$

Çift şablonlar iki parçaya bölünmek zorundadır. İlk iki çift şablonun yeni yapısı aşağıda görülmektedir:

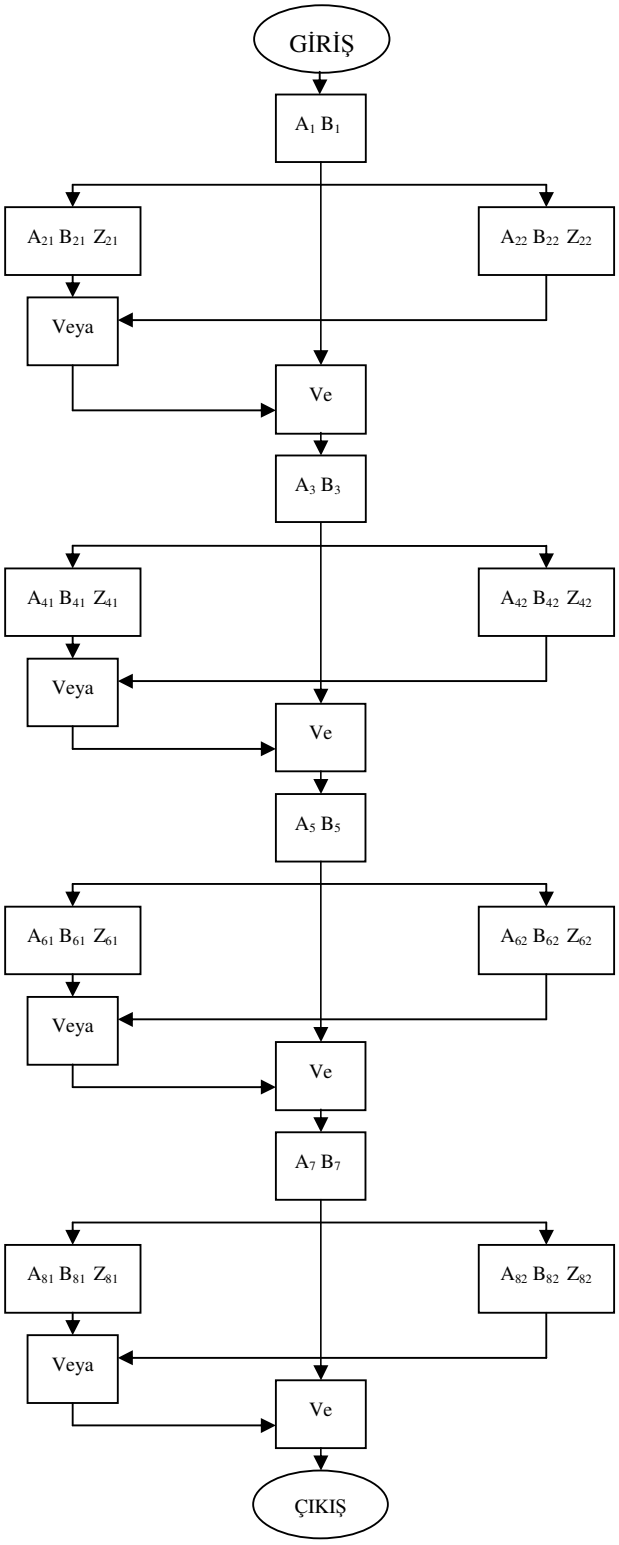
$$A_{21} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B_{21} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -3 & 0 & -3 \end{bmatrix} \qquad Z_{21} = -4$$

$$A_{22} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B_{22} = \begin{bmatrix} 1.5 & 1.5 & 1.5 \\ 0 & 0 & 0 \\ 0 & -1.5 & 0 \end{bmatrix} \qquad \begin{aligned} Z_{22\_1} &= 4.5 \\ Z_{22\_2} &= 6 \end{aligned}$$

$$A_{41} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B_{41} = \begin{bmatrix} -3 & 0 & 0 \\ 0 & 0 & 0 \\ -3 & 0 & 0 \end{bmatrix} \qquad Z_{41} = -4 \qquad \textbf{(3.11)}$$

$$A_{42} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B_{42} = \begin{bmatrix} 0 & 0 & 1.5 \\ -1.5 & 0 & 1.5 \\ 0 & 0 & 1.5 \end{bmatrix} \qquad \begin{aligned} Z_{42\_1} &= 4.5 \\ Z_{42\_2} &= 6 \end{aligned}$$

Burada ACExxK kırmığının bir özelliği olan çift biasın kullanıldığına dikkat edilmelidir. Bu durum gösterilmediği takdirde sıfırdır. Uygulanan iskelet tespit algoritmasının bir iterasyonunun akış şeması Şekil 3.10 da gösterilmiştir. Bu algoritma sırasıyla birçok kez uygulanır.

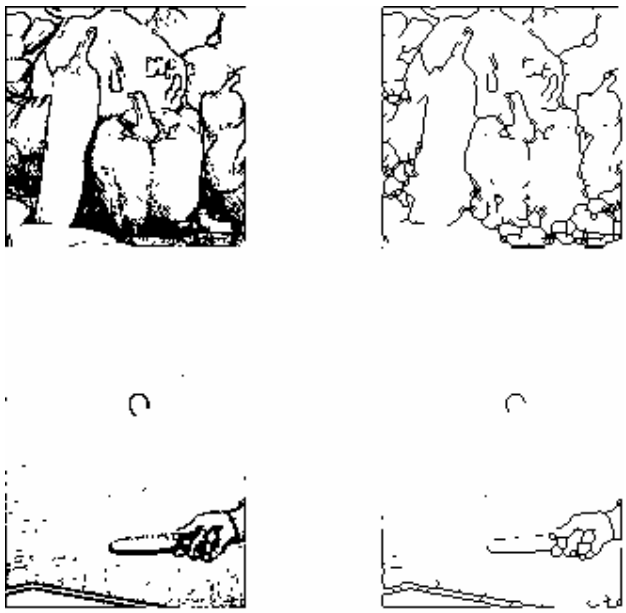


Şekil 3.10: İskeletleme algoritması akış diyagramı

ACE4K ve ACE16K kırmıklarında kullanılan iskelet belirleme algoritması aynıdır. ACE16K kırmığı için yapılan tek modifikasyon her adımda ofset bias maskesinin

tanıtılmasıdır. Her şablonun dinamikleri geri besleme şablonunun merkez elemanını değiştirerek azaltılır. Geri besleme şablonu pozitif olmalıdır.

Ancak belirtilen iskeletleme algoritmasının hem oldukça işlem yükü getirmesi hem de belirtilen şablonların kırmık üzerinde iyi çalışmamasından dolayı yapılan uygulamada, buradaki algoritma yerine Instant Vision kütüphanesindeki Skeleton fonksiyonundan yararlanıldı. İskeletleme işlemi için sağ taraftaki küçük hata düzeltme işlemi sonucunda elde edilen görüntü kullanılmıştır. Kütüphanedeki fonksiyon üç iterasyon adımı olarak uygulandı. Bu işlem sonucunda elde edilen görüntü yine Instant Vision kütüphanesindeki budama (Prune) fonksiyonu uygulanarak istenmeyen ucu açık kenarlar ortadan kaldırıldı. Kütüphanedeki fonksiyonlar siyah görüntüler üzerinde işlem yaptığından giriş görüntüsü ters çevrilerek işleme sokuldu. Bu işlemler sonucunda elde edilen iki farklı çıkış görüntüsü Şekil 3.11 de görülmektedir:



Şekil 3.11: İskeletleme işleminin iki farklı görüntüye uygulanması

3.1.5. Küçük Hata Düzeltme:

Bu işlem siyah arka fondan küçük beyaz nesneleri beyaz arka fondan küçük siyah nesneleri siler. Bu şablonun simülasyon versiyonu aşağıdaki gibidir:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Z = 0 \qquad \textbf{(3.12)}$$

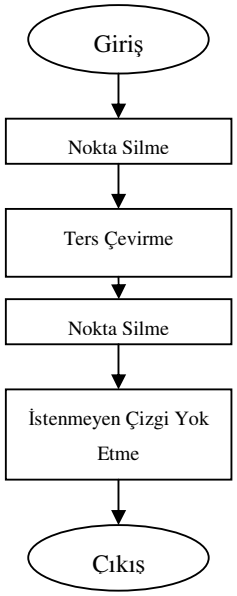
Bu şablon tüm elemanlarıyla birlikte belirlenen sınırlara uygun lineer bir şablondur. Yine de bu şablon ACExxK kırmığı üzerinde düzgün bir şekilde çalışmaz. Bu formülün yeniden düzenlemesiyle bias sıfırdan farklı bir değer alır:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0.2 & 0.2 & 0.2 \\ 0.2 & 3.0 & 0.2 \\ 0.2 & 0.2 & 0.2 \end{bmatrix} \qquad Z = -1 \qquad \textbf{(3.13)}$$

Bu şablonun ACE16K kırmığı için modifiye edilmiş hali aşağıda görüldüğü gibidir:

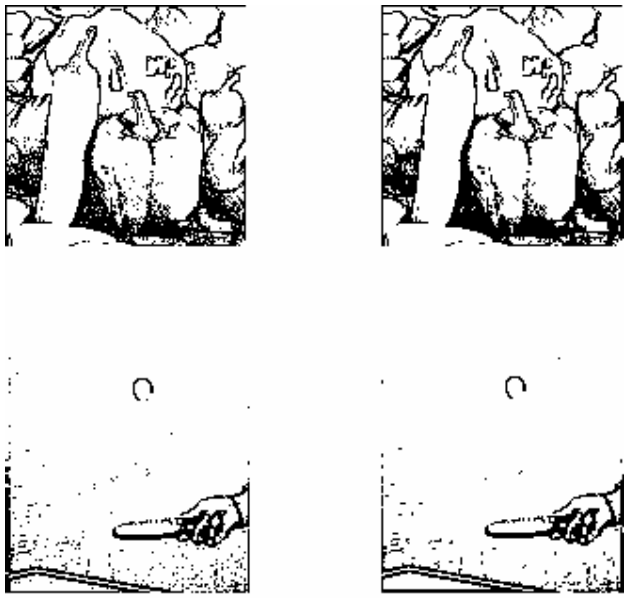
$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Z = -2.5 \qquad \textbf{(3.14)}$$

Hem simülasyon şablonları hem de ACE16K için modifiye edilen şablonlar kırmık üzerinde istenilen şekilde çalışmamaktadır. Bu sorunu gidermek için şablon uygulaması yerine Instant Vision kütüphanesinden de yararlanarak basit bir algoritma kullandık. Uygulanan küçük hata düzeltme algoritmasının akış diyagramı Şekil 3.12 de görülmektedir.



Şekil 3.12: Küçük hata düzeltme için tasarlanan algoritmanın akış diyagramı

Tasarladığımız yeni algoritmada giriş görüntüsü kütüphanedeki nokta silme fonksiyonu (PointRemove) kullanılarak istenmeyen (gürültü olarak gelen) siyah noktalardan temizlendi. Daha sonra görüntü ters çevrilerek bu kez istenmeyen beyaz noktalar temizlendi. Son olarak elde edilen görüntüde istenmeyen kenarlar Instant Vision kütüphanesindeki budama fonksiyonu (Prune) kullanılarak kenarlar daha düzgün bir biçime sokuldu. Kütüphanedeki fonksiyonlar siyah görüntüler üzerinde işlem yaptığından giriş görüntüsü ters çevrilerek işleme sokuldu. Bu algoritma sonucunda elde edilen iki farklı çıkış görüntüsü Şekil 3.13 de görülmektedir.



Şekil 3.13: Küçük hata düzeltme işleminin iki farklı görüntüye uygulanması

3.1.6. Seçili Nesne Çıkarma:

Bu işlem bir görüntüden siyah olarak işaretlenen nesneleri çıkartır. Kısaca *yeniden çağırma* ya da *şekil yeniden oluşturma* olarak adlandırılır. O yüzden iki tane ikilik görüntüye ihtiyaç vardır. Biri tüm siyah nesnelerle birlikte orijinal görüntü, diğeri ise yeniden düzenlenecek nesnenin bulunduğu alanda siyah pikseller içeren işaretleyici görüntüdür. Simülasyon şablonu hem A hem de B şablonlarında limit değerlerini aşan elemanlara sahiptir.

$$A = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 4.0 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Z = 3 \qquad \textbf{(3.15)}$$

Bu işlemi yeniden yazmak için basit bir yeniden ölçekleme çalışmayacaktır. Bu yüzden gürültülü piksellerin oluşmasını engellemek için *fixed state maskesinin* uygulanması önem kazanmaktadır. Bu amaçla orijinal görüntü, fixed state görüntüsü olarak LLM4 e yerleştirilir. Bu arada işaretleyici görüntü ilk durum görüntüsü gibi çalışır. Bu durumda oluşan yeni şablon aşağıdaki gibidir:

$$A = \begin{bmatrix} 0.41 & 0.59 & 0.41 \\ 0.59 & 1.80 & 0.59 \\ 0.41 & 0.59 & 0.41 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Z = 4 \qquad \textbf{(3.16)}$$

Bu metodolojideki bazı gözlemler şunlardır:

- LLM4'e yerleştirilen fixed state mask, hücre geçişinin meydana geldiği yerleri belirtmek için beyaz piksellere, donmuş hücreleri belirtmek için siyah piksellere sahiptir.

- Giriş görüntüsü sıfırla doldurulmalı ve bir LAM belleğe yerleştirilmelidir.

Nesne çıkarma işi üç giriş görüntüsü kullanmaktadır: asıl görüntü, işaretleyici görüntü, fixed state mask. Birinci ve üçüncü görüntü arasındaki tek fark biri birinin tersi olmalarıdır. Bu durumda sadece iki görüntüyü düşünmek mümkündür. ACE4K dan ACE16K ya geçişte en büyük kolaylık budur.

ACE16K için belirlenen şablonlar yine kırmık üzerinde istenilen şekilde çalışmamaktadır. Bu sorunu aşmak için yine Instant Vision kütüphanesinde bulunan yeniden çağırma (Recall) fonksiyonu kullanıldı. Kütüphane fonksiyonları siyah görüntüler üzerinde işlem yaptığından işaretleyici görüntü ve giriş görüntüsü ters çevrilerek işleme sokuldu. Seçili nesne giderme işlemi sonucunda elde edilen iki farklı çıkış görüntüsü Şekil 3.14 te görülmektedir.

Şekil 3.14: Seçili nesne çıkarma işleminin iki farklı görüntüye uygulanması (soldan itibaren işaretleyici görüntü, giriş görüntüsü, çıkış görüntüsü)

3.1.7. Kenar Belirleme:

Bu işlem siyah bir nesneden dış hattı çıkararak kenar bölgelerinde ince bir çizginin yer aldığı bir görüntü oluşturur. Simülasyon şablonu aşağıdaki gibidir:

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix} \qquad B=\begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix} \qquad \text{Z = -1} \qquad \textbf{(3.17)}$$

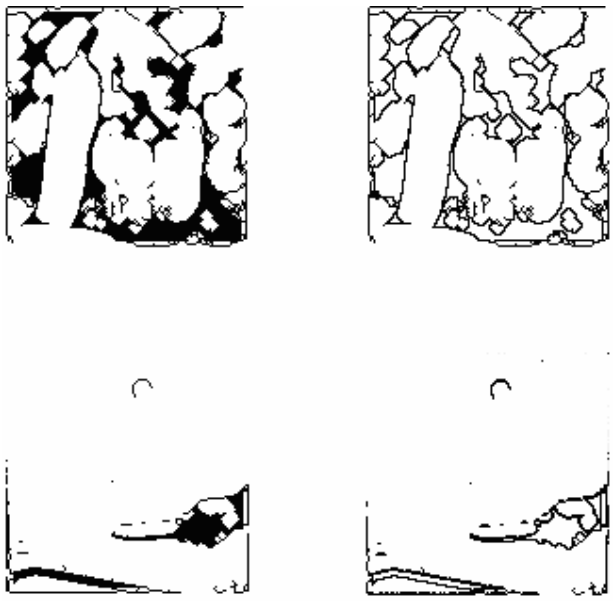
Burada aralık dışındaki merkez elemanının etkisi iki yüksek bias elemanı eklenerek sağlanmıştır. Bu durumda oluşan şablon aşağıdaki gibidir:

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 3 & 0\\ 0 & 0 & 0\end{bmatrix} \qquad B=\begin{bmatrix}-1 & -1 & -1\\ -1 & 0 & -1\\ -1 & -1 & -1\end{bmatrix} \qquad Z_1=6 \qquad Z_2=4.8 \qquad \textbf{(3.18)}$$

Bu işlem algoritmadaki en basit işlemlerden biridir. ACE16K kırmığı üzerinde işlem yapabilmek için şablon yeniden düzenlenmelidir, ayrıca bir lojik XOR işlemi ve ofset bias maskesi kullanılır. Bu şablon Hollow doldurma şablonunun aynısıdır. Tek fark burada şablon siyah nesnenin hatlarını tespit edebilmek için birkaç kez yinelenmiştir. Daha sonra lojik XOR operatörü sadece artan pikselleri seçer.

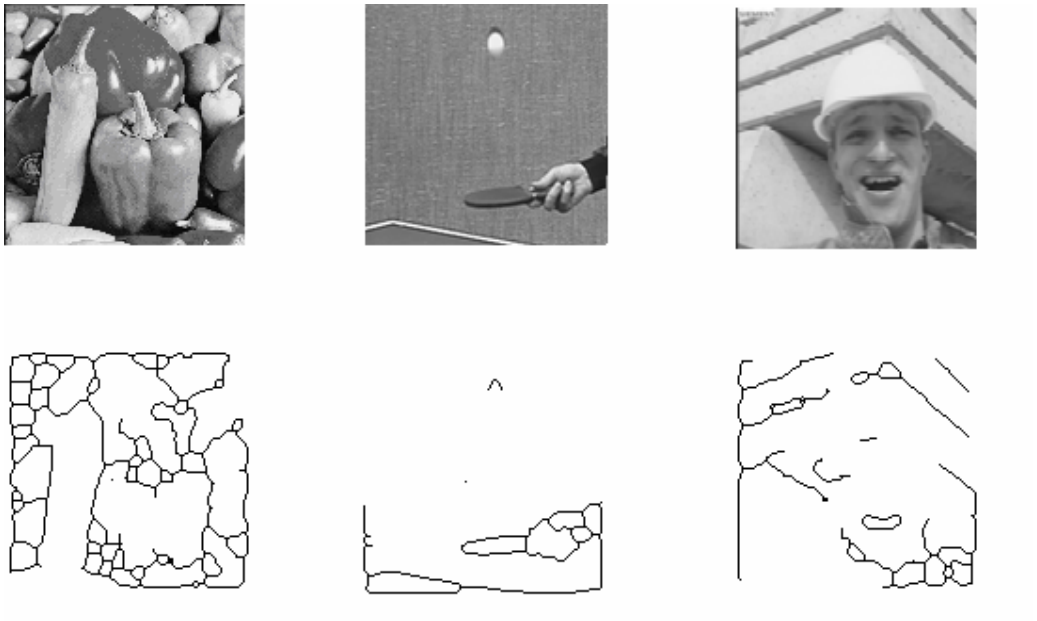
$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Z = 2 \tag{3.19}$$

ACE16K için belirlenen şablonlar kırmık üzerinde istenilen şekilde çalışmamaktadır. Bu sorunu aşmak için Instant Vision kütüphanesindeki Kenar Belirleme (Edge4) fonksiyonu kullanılmıştır. Kütüphane fonksiyonları siyah görüntüler üzerinde işlem yaptığından giriş görüntüsü ters çevrilerek işleme sokulmuştur. Kenar belirleme işlemi sonucunda elde edilen iki farklı çıkış görüntüsü Şekil 3.15 te görülmektedir.



Şekil 3.15: Kenar belirleme işleminin iki farklı görüntüye uygulanması

Tüm algoritmanın uygulanmasından sonra elde edilen giriş ve çıkış görüntüleri Şekil 3.16 daki gibi olmaktadır. Algoritmaya ilişkin program ekteki CD'de verilmiştir.



Şekil 3.16: Tüm algoritmanın üç farklı görüntüye uygulanması

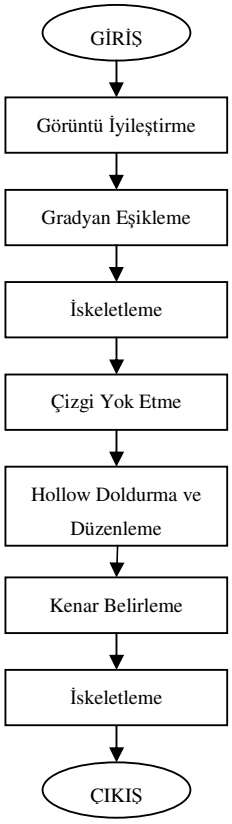
Algoritmanın her adımının gerçekleşmesi için geçen süreler Tablo 3.1 de verilmiştir.

Tablo 3.1: Early bölütleme algoritmasının işlem adımları için geçen zamanlar

Algoritma Adımı	**Süre(µs)**
Median Filtre	42169
Gradyan Eşikleme(Sağ Taraf)	37325
Gradyan Eşikleme(Sol Taraf)	37316
Küçük Hata Düzeltme(Sağ Taraf)	24542
Küçük Hata Düzeltme(Hollow Öncesi)	26394
Küçük Hata Düzeltme(XOR Öncesi)	27973
Hollow Doldurma	24192
Lojik XOR	24801
Seçili Nesne Çıkarma	27693
İskeletleme Birinci Adım	27985
Lojik OR	23388
Kenar Belirleme	24292
İskeletleme İkinci Adım	72379
Toplam Süre	308335

3.2. ROSKA BÖLÜTLEME ALGORİTMASI:

Literatürde yer alan bir diğer bölütleme algoritması Stoffels, Roska ve Chua'nın birlikte önermiş olduğu bölütleme algoritmasıdır [28]. Algoritmanın akış diyagramı Şekil 3.17 de görülmektedir. Şekilde görülen algoritma [28] da önerilen bölütleme ve nesne etiketleme algoritmasının bir adımını oluşturmaktadır. Buradaki algoritma, bölütleme işleminin sonuçlarının daha yüksek seviyeli sayısal görüntü işleme algoritmalarında kullanıldığı gösteren güzel bir örnektir.

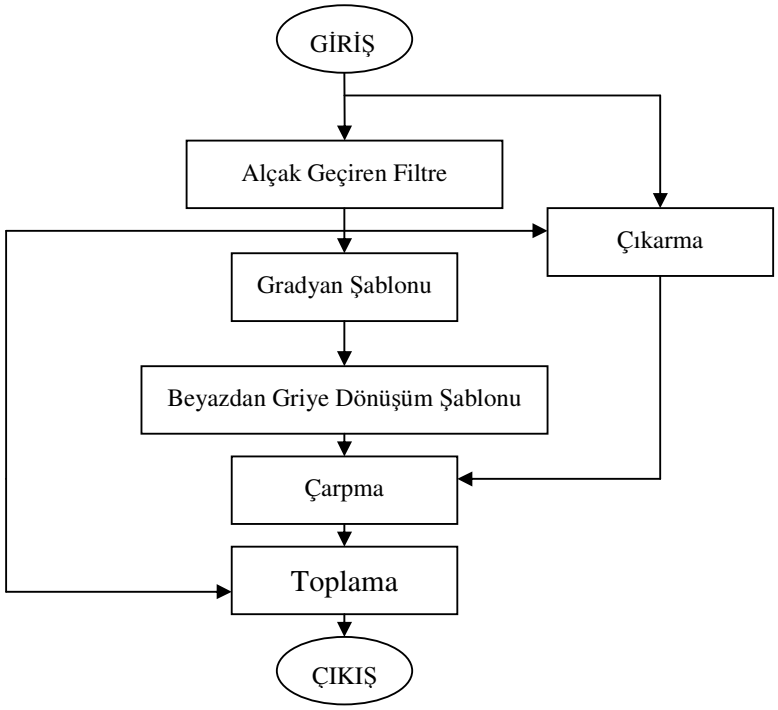


Şekil 3.17: Roska bölütleme algoritmasının akış diyagramı

Şimdi bu algoritmanın adımlarını detaylı olarak açıklayalım.

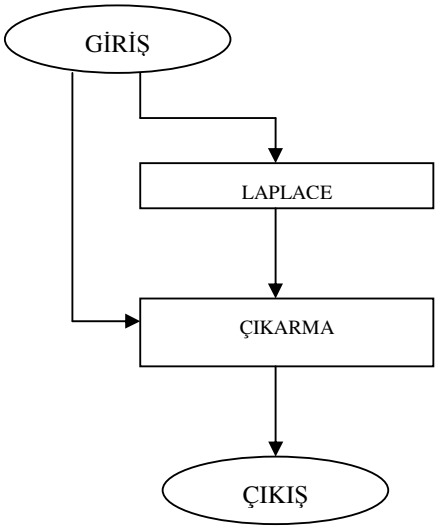
3.2.1. Görüntü İyileştirme:

Bu adım görüntüyü gürültü bileşenlerinden temizleyerek daha net bir görüntü elde etmek için uygulanır. Algoritmanın kendisinde önerilen işlem adımları Şekil 3.18 de görülmektedir.



Şekil 3.18: Görüntü iyileştirme işleminin blok diyagramı

Şekil 3.18 de görülen gradyan şablonu lineer olmayan bir şablondur. Bu özelliğinden dolayı ACE16K üzerinde çalıştırılamamaktadır. Bu şablonun ACE16K üzerinde çalıştırılabilecek şekilde lineer olarak yeniden düzenlenmesi gerekmektedir. Şablonu yeniden düzenlemek deneme yanılmaya dayanan oldukça zaman alan bir işlem olduğundan görüntü iyileştirme adımını bir median filtre kullanarak gerçekleştirmek mümkündür. Median filtre bir seçenek olabileceği gibi bir diğer seçenek yeni tasarlanan bölütleme algoritmasında da kullanılacak olan görüntünün Laplace işlemine sokulup asıl görüntüden çıkarılmasıdır. Bu işlem için gerekli olan adımlar Şekil 3.19 da görülmektedir.



Şekil 3.19: Yeniden düzenlenen görüntü iyileştirme adımının blok diyagramı

Burada görüntü öncelikle laplace işlemine sokulmaktadır. Laplace işlemi Instant Vision kütüphanesinde bulunan Laplace fonksiyonuyla gerçekleştirilmektedir. Laplace işlemi sonucunda elde edilen görüntü giriş görüntüsünden çıkartılarak sonuçta daha net ve nesnelerin kenarları daha belirgin bir görüntü elde edilmektedir. Yeniden düzenlenen görüntü iyileştirme adımının iki farklı görüntüye uygulanması sonucunda elde dilen çıkışlar Şekil 3.20 de görülmektedir.

Şekil 3.20: Yeniden düzenlenen görüntü iyileştirme işleminin iki farklı görüntüye uygulanması

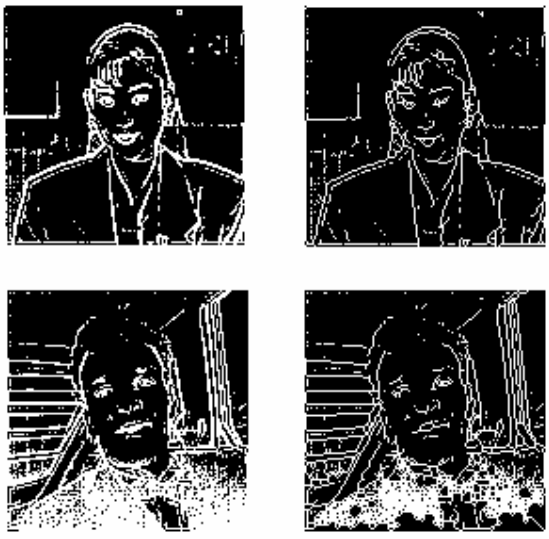
3.2.2. Gradyan Eşikleme:

Bu adım Early bölütleme algoritmasında bahsedilen adımın aynısıdır. Görüntü iyileştirme işlemi sonucunda elde edilen görüntü sekiz farklı doğrultudaki şablon kümesiyle konvolüsyona sokularak görüntüdeki nesnelerin kenarları tespit edilmiş olur. Buradaki tek fark uygulanan eşik değeridir. Burada gerçeklediğimiz algoritmada eşik değeri -2.4 olarak seçildi. Gradyan eşikleme işleminin iki farklı görüntüye uygulanması sonucunda elde edilen çıkış görüntüleri Şekil 3.21 de görülmektedir.

Şekil 3.21: Gradyan eşikleme işleminin iki farklı görüntüye uygulanması

3.2.3. İskeletleme:

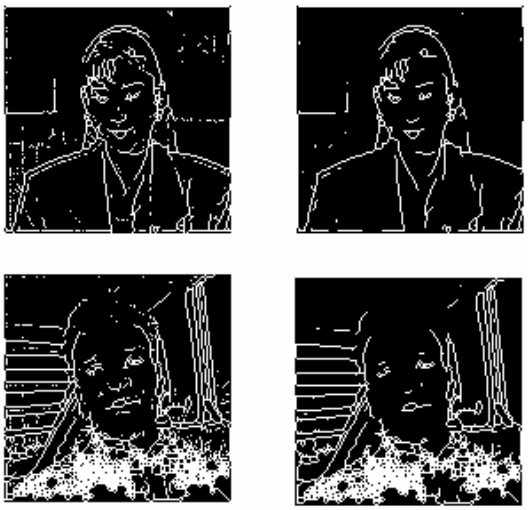
Gradyan eşikleme işleminden sonra elde edilen görüntünün kenarlarını bir piksel kalınlığına düşürmek için iskeletleme adımı uygulanır. Bu işlem kütüphanenin hazır fonksiyonu Skeleton ile gerçekleştirilmektedir. Skeleton fonksiyonu morfolojik bir işlem olduğu için sadece beyaz görüntüler üzerinde işlem yapılabilmektedir. Bu yüzden gradyan eşikleme işlemi sonucunda elde edilen görüntüye Skeleton fonksiyonu uygulanmadan önce görüntünün Inversion fonksiyonuyla ters çevrilmesi gerekmektedir. İskeletleme işleminin iki farklı görüntüye uygulanması sonucunda elde edilen sonuçlar Şekil 3.22 de görülmektedir.



Şekil 3.22: İskeletleme işleminin iki farklı görüntüye uygulanması

3.2.4. Çizgi Yok Etme:

İskeletleme işleminden sonra herhangi bir kenar belirtmeyen çizgileri ortadan kaldırmak için çizgi yok etme şablonu kullanılmaktadır. Burada şablon kullanmak yerine yine kütüphanenin Prune hazır fonksiyonundan yararlanılmaktadır. Daha önce de bahsedildiği üzere Prune fonksiyonunun iterasyon sayısı kritik bir noktadır. İterasyon sayısının fazla olması durumunda gerçekte bir nesnenin kenarını belirten çizgiler de ortadan kaldırılabilir. Çizgi yok etme işleminin iki farklı görüntüye uygulanması sonucunda elde edilen sonuçlar Şekil 3.23 te görülmektedir.



Şekil 3.23: Çizgi yok etme işleminin iki farklı görüntüye uygulanması

3.2.4. Hollow Doldurma ve Düzenleme:

Çizgi yok etme işleminden sonra Early bölütleme algoritmasında da yer alan Hollow doldurma işlemi uygulanarak nesnelerin içi doldurulur. Bu işlem için Early bölütleme algoritmasındaki şablon kullanılabileceği gibi kütüphanenin hazır fonksiyonları HoleFiller veya Closing8 de kullanılabilir. Burada şablon yerine HoleFiller fonksiyonu kullanılmıştır. Hollow doldurma işleminden sonra içi doldurulan nesnelerin kenarlarını düzlemek için Patch şablonu kullanılmaktadır. Bu şablon aşağıda görüldüğü gibidir:

$$A = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 4 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Z = -1.5 \qquad \textbf{(3.20)}$$

Hollow doldurma ve Patch şablonunun iki farklı görüntüye uygulanması sonucunda elde edilen sonuçlar Şekil 3.24 te görülmektedir.

Şekil 3.24: Hollow doldurma ve Patch işlemlerinin iki farklı görüntüye uygulanması

3.2.4. Kenar Belirleme:

Hollow ve patch işlemlerinden sonra elde edilen görüntünün kenarlarını belirlemek için görüntü [28] de verilen şablonlarla konvolüsyona sokulmaktadır. Bu şablonlar aşağıda görüldüğü gibidir:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} -0.25 & -0.25 & -0.25 \\ -0.25 & 2 & -0.25 \\ -0.25 & -0.25 & -0.25 \end{bmatrix} \qquad Z = -1.4 \qquad \textbf{(3.21)}$$

Bu şablonları kullanabileceğimiz gibi kütüphanenin hazır fonksiyonu Edge8 de kullanılabilir. Algoritmanın gerçeklenmesi esnasında kütüphanenin hazır fonksiyonlarından yararlanılmıştır. Kenar belirleme işleminin iki farklı görüntüye uygulanması sonucunda elde edilen sonuçlar Şekil 3.25 te görülmektedir.

Şekil 3.25: Kenar belirleme işleminin iki farklı görüntüye uygulanması

Kenar belirleme işleminden sonra son olarak iskeletleme işlemi tekrar uygulanarak görüntüdeki nesnelerin kenarları bir piksel kalınlığına indirilmektedir. Ancak bu işlem sonucunda da halen görüntünün kenarlarında açıklıklar mevcutsa kütüphanenin hazır fonksiyonlarından Dilate8 ya da Closing8 fonksiyonu Skeleton fonksiyonuyla art arda kullanılarak bu açık kenarların birleştirilmesi mümkündür. Tüm algoritmanın bazı resimlere uygulanması sonucunda elde edilen çıkış görüntüler Şekil 3.26 da görülmektedir. Algoritmaya ilişkin program ekteki CD'de verilmiştir.

Şekil 3.26: Roska bölütleme algoritmasının bazı görüntülere uygulanması

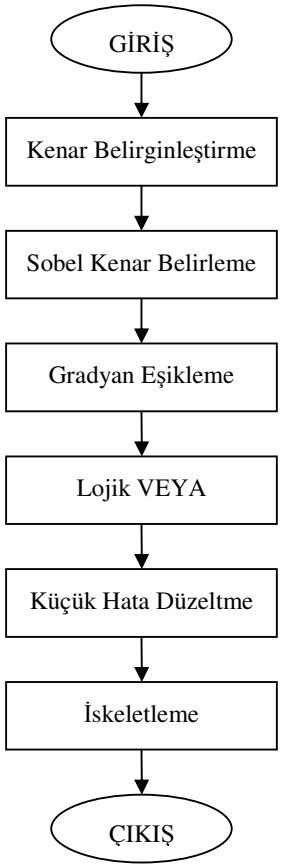
Algoritmanın her adımının gerçekleşmesi için gereken süreler Tablo 3.2 de verilmiştir.

Tablo 3.2: Roska bölütleme algoritmasının işlem adımları için geçen zamanlar

Algoritma Adımı	**Süre(µs)**
Görüntü İyileştirme	112075
Gradyan Eşikleme	35316
İskeletleme	22682
Çizgi Yok Etme	13542
Hollow Doldurma ve Düzenleme	24784
Kenar Belirleme	24361
İskeletleme	27511
Toplam Süre	260271

3.3. YENİ BİR BÖLÜTLEME ALGORİTMASI TASARIMI:

Literatürdeki bölütleme algoritmalarından esinlenerek yeni bir bölütleme algoritması tasarlanabilir. Bölütleme metotlarından yola çıkarak yeni bir algoritma oluşturuldu. Bu algoritmanın akış diyagramı Şekil 3.27 de görüldüğü gibidir.

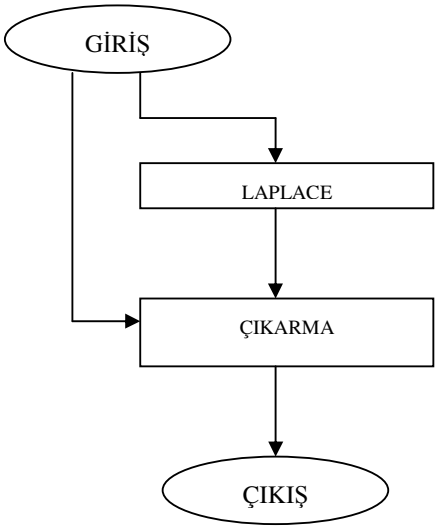


Şekil 3.27: Yeni bölütleme algoritmasının akış diyagramı

Buradaki adımlara ilave olarak, giriş görüntüsünün bir filtreden geçirilmesi adımı eklenmesi ya da kenar belirginleştirme adımından sonra bir filtreleme adımı eklenmesi mümkündür. Böylelikle giriş görüntüsünden gelen gürültüden kurtulabiliriz. Bu filtreleme işlemi için median filtre kullanılabilir. Şimdi bu algoritmadaki adımları tek tek açıklayalım.

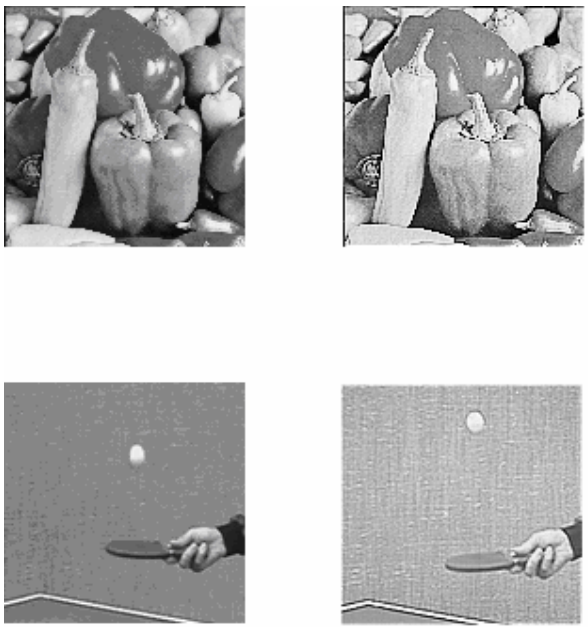
3.3.1. Kenar Belirginleştirme:

Kenar belirginleştirme işleminin akış diyagramı Şekil 3.28 de görülmektedir.



Şekil 3.28: Kenar belirginleştirme işleminin akış diyagramı

Akış diyagramında da görüldüğü gibi giriş görüntüsü ilk olarak kütüphanenin hazır fonksiyonu Laplace ile işleme sokulmaktadır. Laplace işlemi sonucunda elde edilen görüntü giriş görüntüsünden çıkarılarak kenarları daha belirginleşmiş çıkış görüntüsü elde edilir. Şekil 3.29 da iki farklı görüntü için kenar belirleme işlemi sonucunda elde edilen çıkış görüntüleri verilmiştir.



Şekil 3.29: Kenar belirginleştirme işleminin iki farklı görüntüye uygulanması

3.3.2. Sobel Kenar Belirleme:

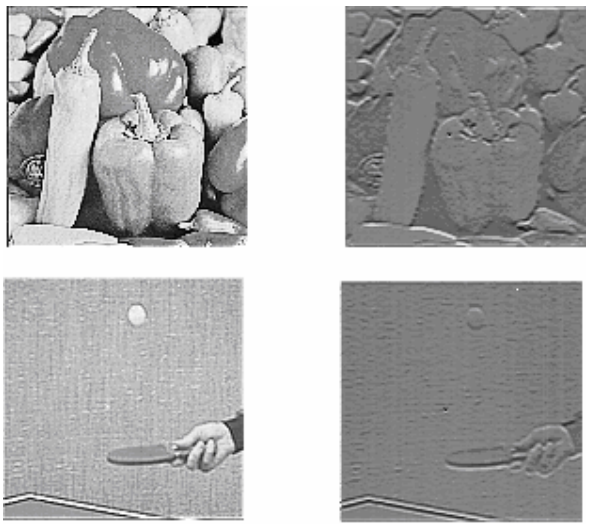
Algoritmanın bu adımında kenar belirginleştirme adımında elde edilen görüntü sobel operatörü ile işleme sokulur. Bu işlem sobel operatörünün yatay, dikey ve diagonal olarak uygulanmasıyla gerçekleştirilir. Bu işlem sırasında kullanılan sobel şablonları aşağıda görüldüğü gibidir:

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & -6 & 0\\ 0 & 0 & 0\end{bmatrix} \qquad B=\begin{bmatrix}-0.75 & 0 & 0.75\\ -1.5 & 0 & 1.5\\ -0.75 & 0 & 0.75\end{bmatrix} \qquad Z=3$$

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & -6 & 0\\ 0 & 0 & 0\end{bmatrix} \qquad B=\begin{bmatrix}-0.75 & -1.5 & -0.75\\ 0 & 0 & 0\\ 0.75 & 1.5 & 0.75\end{bmatrix} \qquad Z=3 \qquad \textbf{(3.22)}$$

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & -6 & 0\\ 0 & 0 & 0\end{bmatrix} \qquad B=\begin{bmatrix}-1.5 & -0.75 & 0\\ -0.75 & 0 & 0.75\\ 0 & 0.75 & 1.5\end{bmatrix} \qquad Z=3$$

Sobel operatörünün uygulanması sonucunda görüntünün kenarları belirlenmiş olur. Kenarların daha belirgin çıkması için sobel operatörü sekiz farklı doğrultudan da uygulanabilir. Ancak burada üç doğrultudan sobel operatörünün uygulanması kenarların belirlenmesi için yeterli olmaktadır. İki farklı görüntü için sobel operatörünün çıkış görüntüleri Şekil 3.30 de verilmiştir.



Şekil 3.30: Dikey doğrultuda sobel operatörünün iki farklı görüntüye uygulanması

3.3.3. Gradyan Eşikleme:

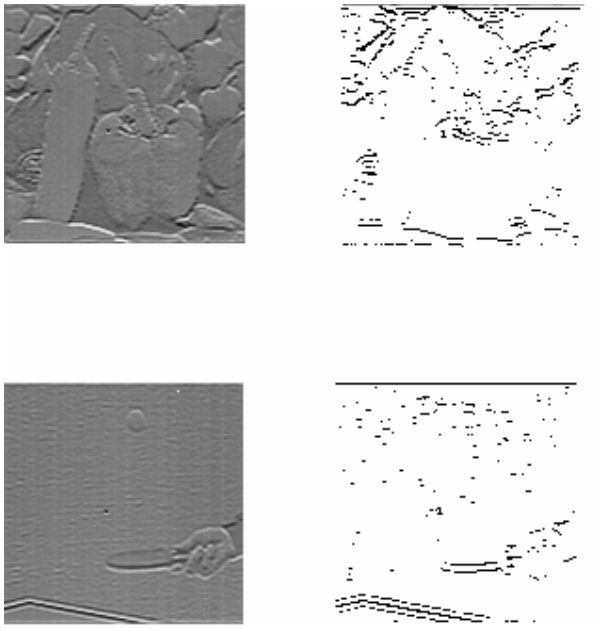
Bu adım Early bölütleme algoritmasındaki gradyan eşikleme işlemiyle tamamen aynıdır. Buradaki tek fark bu işlemin sekiz farklı doğrultudan değil, sobel operatörünün uygulandığı yatay, dikey ve diagonal doğrulardan uygulanıyor olmasıdır. Sobel adımında belirttiğimiz gibi sobelin sekiz farklı doğrultuda uygulanması durumunda gradyan eşikleme de sekiz farklı doğrultuda uygulanır ve böylece elde edilen çıkış görüntüsünün kenarları daha belirgin olur. Gradyan eşikleme işlemi için eşik değeri -1.1 olarak alınmıştır. Burada kullanılan gradyan eşikleme şablonları aşağıda görüldüğü gibidir:

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix} \qquad B=\begin{bmatrix}0 & 0 & 0\\ 0 & 3 & -3\\ 0 & 0 & 0\end{bmatrix} \qquad \text{Z = -1.1}$$

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix} \qquad B=\begin{bmatrix}0 & 0 & 0\\ 0 & 3 & 0\\ 0 & -3 & 0\end{bmatrix} \qquad \text{Z = -1.1} \qquad \textbf{(3.23)}$$

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix} \qquad B=\begin{bmatrix}0 & 0 & 0\\ 0 & 3 & 0\\ 0 & 0 & -3\end{bmatrix} \qquad \text{Z = -1.1}$$

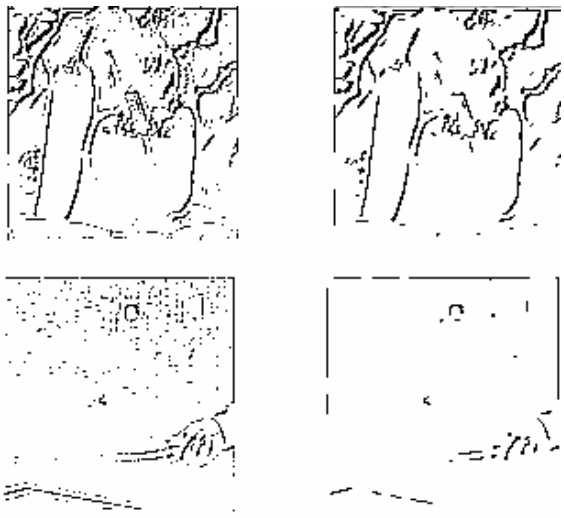
Daha sonra elde edilen yatay, dikey ve diagonal doğrultulardaki görüntüler lojik VEYA işlemine sokulur ve görüntüler toplanır. Böylelikle çıkış görüntüsü giriş görüntüsünün kenarlarını tam olarak verir. Gradyan eşikleme işleminin iki farklı görüntüye uygulanması Şekil 3.31 de gösterilmiştir.



Şekil 3.31: Gradyan eşikleme işleminin görüntüye yatay doğrultuda uygulanması

3.3.4. Küçük Hata Düzeltme:

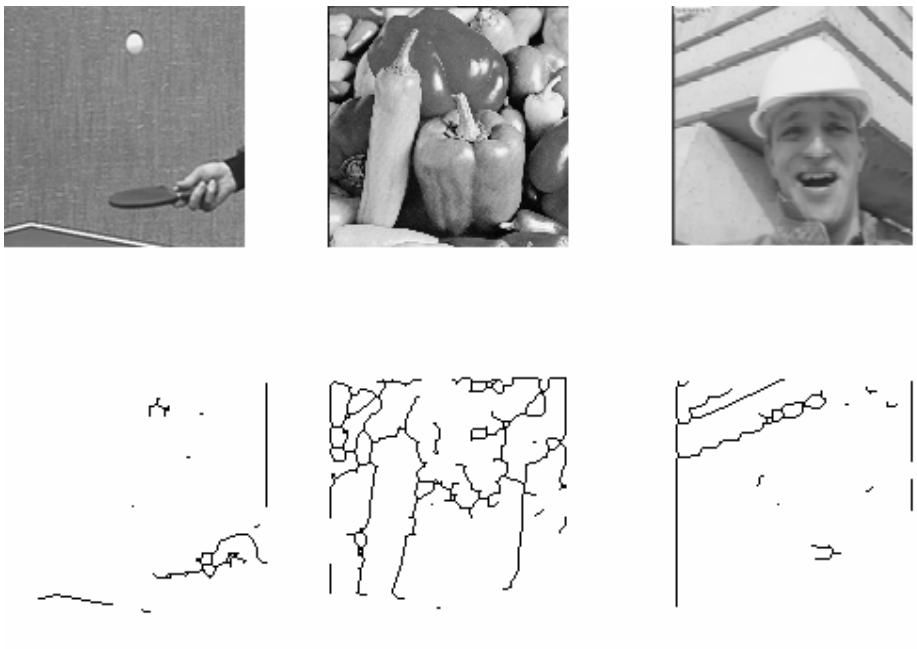
Bu adımda Early algoritmasındaki küçük hata düzeltme adımlarının aynısıdır. Gradyan eşikleme sonucunda elde edilen görüntü kütüphanedeki nokta silme (PointRemove) fonksiyonu kullanılarak görüntüye gürültü olarak gelen istenmeyen beyaz ve siyah noktalar görüntüden temizlenir. Küçük hata düzeltme işlemi sonucunda elde edilen görüntüler Şekil 3.32 de görülmektedir.



Şekil 3.32: Küçük hata düzeltme işleminin iki farklı görüntüye uygulanması

3.3.5. İskeletleme:

İskeletleme adımı da Early bölütleme algoritmasıyla aynı işlemlere sahiptir. Küçük hata düzeltme sonucunda elde edilen görüntü önce kütüphanenin hazır iskeletleme fonksiyonuyla (Skeleton) ile işleme sokulur. Bu işlem sonucunda hatların kalınlığı bir piksel olan yeni bir görüntü elde edilir. Ancak bu görüntüde kopuk kenarlar oluşmuş olabilir. Kenarları birleştirmek için kütüphanenin genişleme (Dilate4) fonksiyonu bir iterasyon adımı kullanılır. Daha sonra tekrar iskeletleme fonksiyonu kullanılır. Oluşan görüntüde halen kopuk kenarlar mevcut ise genişletme ve iskeletleme fonksiyonlarının art arda uygulanmasına devam edilir. Kenarların birleştirilmesi işlemi de tamamlandıktan sonra son olarak istenmeyen açık kenarlardan kurtulmak için kütüphanenin budama (Prune) fonksiyonu beş iterasyon adımı kullanılır. Budama işleminden sonra bölütlenmiş görüntü elde edilir. Şekil 3.33 de tüm algoritmanın üç farklı görüntüye uygulanması sonucunda elde edilen bölütlenmiş görüntüler verilmiştir. Algoritmaya ilişkin program ekteki CD'de verilmiştir.



Şekil 3.33: Yeni bölütleme algoritmasının üç farklı görüntüye uygulanması

Algoritmanın her adımının gerçekleşmesi için geçen süreler Tablo 3.3 de verilmiştir.

Tablo 3.3: Yeni bölütleme algoritmasının işlem adımları için geçen zamanlar

Algoritma Adımı	**Süre(µs)**
Kenar Belirginleştirme	124075
Sobel ve Gradyan Eşikleme	134677
Lojik VEYA	25990
Küçük Hata Düzeltme	25764
İskeletleme	50934
Toplam Süre	312715

4. SONUÇLAR:

Gelişen bilgisayar teknolojisi sayesinde görüntü işleme, görüntü bölütleme gibi problemlerin çözümü oldukça kolaylaşmış ve bu durum görüntü işleme, görüntü bölütleme araştırmalarında önemli gelişmelerin gerçekleşmesine yol açmıştır. Bölüm 1'de literatürde mevcut olan bölütleme metotlarından bahsedilmiştir.

Hücresel yapay sinir ağlarının görüntü işlemede kullanılmaya başlanması ve elektronik olarak gerçeklenmesiyle (ACE16K) bugün en hızlı bilgisayarlar yardımıyla bile saatlerce süren işlemler saniyenin de altında çözümlenebilmektedir. Burada hücresel yapay sinir ağlarının paralel ve analog yapıda olması bu hızın elde edilmesini sağlamaktadır. Bölüm 2'de hücresel sinir ağları yapılarından ve elektronik gerçeklemelerinden bahsedilmiştir.

Bölüm 3'de literatürde mevcut olan bölütleme algoritmalarından Early ve Roska bölütleme algoritmalarından detaylı olarak bahsedilmiş ve uygulama sonuçları verilmiştir. Early ve Roska bölütleme algoritmalarının gerçeklenmesi sırasında birçok sınırlamayla karşılaşılmıştır. Bunun sebebi ACE16K kırmığının kararlı olarak çalışmaması ve kırmığın mimari yapısından kaynaklanan sınırlamalar olarak söylenebilir. Algoritma adımları esnasında kullanılan şablonlar kırmık üzerinde belirli değerler aralığında çalıştırılabilmektedir. Ancak işlem adımları sırasında kullanılan küçük hata düzeltme, iskeletleme şablonlarındaki bazı terimler bu aralığın dışına çıkmaktadır. Bu yüzden bu şablonların kırmık üzerinde çalışabilmesi için modifiye edilmesi gerekmektedir. Ancak bu modifikasyon şablonların tam olarak işlevlerini yerine getirememesine neden olmaktadır. Bu sorunu aşabilmek için şablonlar yerine kütüphanenin hazır fonksiyonlarından yararlanıldı ve oldukça başarılı sayılabilecek sonuçlar elde edildi.

Yine bölüm 3'de yeni bir bölütleme algoritmasının tasarımından bahsedilmiş ve uygulama sonuçları verilmiştir. Bu algoritmanın tasarımı sırasında da şablonların kullanımında sınırlamalar yaşandığından algoritmanın bazı adımlarında kütüphanenin hazır fonksiyonlarından yararlanılmıştır.

Gerçeklenen üç algoritmanın farklı görüntülere uygulanması sonucunda elde edilen çıkış görüntüleri Şekil 4.1 de görülmektedir. Ayrıca literatürde yer alan bir diğer algoritma olan Grassi [29] bölütleme algoritmasının İstanbul Üniversitesi'nde Fethullah KARABİBER tarafından gerçeklenmesiyle elde edilen sonuçlar da Şekil 4.1 de görülmektedir.

Şekil 4.1: Bölütleme algoritmalarının karşılaştırılması (soldan itibaren giriş görüntüsü, Roska, Early, Grassi ve yeni tasarlanan bölütleme algoritmalarının çıkışları)

Elde edilen sonuçlardan da görüldüğü gibi Grassi bölütleme algoritması daha iyi sonuç vermektedir. Ancak Grassi bölütleme algoritmasının gerçeklenmesi esnasında ACE16K ile birlikte DSP de kullanılmaktadır. Bölütleme algoritmalarının gerçeklenmesinde karşılaşılan en büyük problem eşik değerinin seçimidir. Yeni tasarlanan bölütleme algoritmasında eşik değeri küçük seçildiğinde görüntünün kenarları belirlenememektedir. Tam tersine eşik değeri büyük seçildiğinde bu sefer

görüntüye kenar olmadığı halde kenar gibi gelen çok fazla gürültü gelmektedir. Bu sorunu aşmak için her görüntünün eşik değerinin otomatik olarak belirlendiği otomatik eşik seçimi algoritması tasarlanabilir. Böylelikle seçilen eşik değeri optimum olacak ve kenarlar en iyi şekilde belirlenmiş olacaktır.

Tüm algoritmaların gerçeklenmesi için geçen süreler Tablo 4.1'de verilmiştir.

Tablo 4.1: Bölütleme algoritmalarının gerçeklenmesi için geçen süreler

Bölütleme Algoritması	**Süre(µs)**
Roska Bölütleme Algoritması	224 548
Early Bölütleme Algoritması	240 741
Grassi Bölütleme Algoritması	2 928 355
Yeni Tasarlanan Bölütleme Algoritması	312 715

Tablo 4.1'den de görüldüğü gibi Grassi bölütleme algoritmasının gerçeklenmesi için geçen süre diğer algoritmalara oranla çok daha fazladır. Bunun nedeni algoritma adımlarının diğer algoritmalara oranla daha çok olmasının yanında gerçekleme esnasında ACE16K ile birlikte DSP nin de kullanılmasıdır. Görüldüğü gibi genel olarak algoritmaların gerçeklenmesi için gereken süreler bir saniyenin altındadır bu nedenle gerçek zamanda bölütleme yapabilmek mümkündür. Bu işlemin bilgisayar ortamında gerçeklenmeye çalışılması durumunda sadece bir konvolüsyon işlemi dakikalarca sürebilmektedir.

Elde edilen görüntülerden ve gerçeklenme zamanlarından yola çıkarak Grassi bölütleme algoritmasının daha iyi çalıştığı söylenebilir. Ancak Grassi bölütleme algoritmasının gerçekleme zamanına baktığımızda gerçek zamanlı bölütleme için pek uygun olmadığı görülmektedir. Ayrıca otomatik eşik seçimi algoritmasının oluşturularak yeni tasarlanan algoritmada kullanılması durumunda çok daha iyi sonuçlar elde edilebilir. Bu algoritmaların ara adımlarının modifiye edilerek daha da iyi sonuçlar alınması mümkündür.

Bugüne kadar gerçeklenen bölütleme algoritmalarının MATLAB ya da CNN simülatörleri üzerinde uygulandığını göz önüne alırsak elde edilen sonuçlar,

ACE16K üzerinde bölütleme işleminin yapılabileceğinin ve gerçek zamanlı bölütlemenin mümkün olduğunun ispatlanması açısından oldukça önemlidir. Sonuç olarak, bilgisayara oranla çok daha hızlı sonuç elde edilmesi dolayısıyla elektronik bir H.Y.S.A. gerçeklemesi olan ACE16K nın ilerleyen yıllarda etkin bir şekilde kullanılacağı söylenebilir.

KAYNAKLAR:

[1]**Tinku Acharya and Ajoy K. Ray**, 2005. Image Processing Principles and Applications**,** John Wiley & Sons, New Jersey.

[2]**Yatharth Saraf**, May 2006.Algorithms for Image Segmentation**,** Thesis, Birla Institute of Technology and Science**,** Pilani.

[3]**Mahinda P. Pathegama and Özdemir Göl**, 2004. Edge-end Pixel Extraction for Edge-based Image Segmentation, *Proceedings of the International Conference on Signal Processing*, Istanbul, Turkey, December 2004, vol 2 .

[4]**Boykov, Y. and Jolly, M.-P.**, 2001. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images, *In Proceedings of the International Conference on Computer Vision (ICCV)*, volume I, pages 105–112.

[5]**Bernd Jahne**, 2002. Digital Image Processing**,** Springer, Heidelberg.

[6]**C. E. Shannon**, 1948. A mathematical theory of communication, *The Bell System Technical Journal,* vol. **27**, pp. 379–423, 623–656, July, October 1948.

[7]**Sahoo, P., Wilkins, C., and Yeager, J.**, 1997. Threshold selection using Renyi's entropy, *Pattern Recognit.*, **30**, pp. 71–84

[8]**Özgür Tuncer**, 2003. Segmentation, Registration And Visualization Of Medical Images For Treatment Planning**,** MS Thesis, METU, Ankara.

[9]**James C. Tilton**, 2000. Method for recursive hierarchical segmentation by region Growing and spectral clustering with a natural convergence criterion, *Disclosure of Invention and New Technology:* NASA Case No. GSC **14**,328–1.

[10] **Ozan Ersoy**, 2004. Image Segmentation with Improved Region Modelling**,** MS Thesis, METU, Ankara.

[11] **Hopfield, J. J.**, 1982.Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences*, U.S.A., **79**:2554–2558.

[12] **L. Chua ve L. Yang**, 1988. Cellular Neural Networks: Theory, *IEEE Transaction on Circuits and Systems*, Vol **35**, No.10 ; s.1257-1272.

[13] **Chua, L.O ve Yang, L.**, 1988. Cellular Neural Networks: Applications, *IEEE Transaction on Circuits and Systems*, Vol **35**, No.10 ; s.1273-1290.

[14] **T.Roska and L.O. Chua**, 1993. The CNN Universal Machine: An Analogic Array Computer, *IEEE Trans. Circuits and Systems-II*., vol. **40**, pp. 163–173.

[15] **F.Werblin, T.Roska, and L.O. Chua**, 1995. The analogic cellular neural network as a bionic eye, *Int. J. Circuit Theory and Applications*, vol. **23**, no:6, pp. 541–569 (55 ref.).

[16] **Ülkü Şahin, Osman N. Uçan, Orhan Tolluoğlu, Cuma Bayat**, "İstanbul'un Hava Kirliliğinin Hücresel Yapay Sinir Ağ ile Modellenmesi", *The 13th Turkish Symposium on Artificial Intelligent and Neural Networks (TAINN),* İzmir, Türkiye, 10–11 Haziran 2004, pp.501–511.

[17] **Albora, A.M., Ucan, O.N., Ozmen, A.,ve Ozkan, T**., 2001. Separation of Bouguer Anomaly Map Using Cellular Neural Network, *Journal of Applied Geophysics*, Cilt **46**, 129-142.

[18] **Cimagalli, V.** 1993. Cellular Neural Networks A Review, *Proceedings of Sixth Italian Workshop on Parallel Architectures and Neural Networks*, Vietri Sul Mare, Italy, May 12–14.

[19] **E. Çelebi ve C. Güzeliş**, 1995. Hücresel yapay sinir ağları ile görüntü onarımı, *Sinyal İşleme ve Uygulamaları Kurultayı*, April 1995, Cilt-A, 37–41.

[20] **L. O. Chua and T. Roska**, Mar. 1993. The CNN paradigm, *IEEE Trans. Circuits Syst. I*, vol. **40**, pp. 147–156.

[21] **G. Liñàn, R. Domìguez-Castro, S. Espejo, and A. Rodrìguez-Vàzquez**, 2001. ACE16k:A programmable focal plane vision processor with 128_128 resolution, *Eur. Conf. Circuit Theory and Design*, Espoo, Finland, Aug. 28–31 vol.**1**, pp. 345, 348.

[22] **T. Roska and L. O. Chua,** 1993. The CNN Universal Machine: An Analogic Array Computer, *IEEE Transactions on Circuits and Systems- II: Analog and Digital Signal Processing*, Vol. **40**, pp. 163–173.

[23] **Ákos Zarándy and Csaba Rekeczky**, BI-I: A Standalone Ultra High Speed Cellular Vision System**,** *IEEE Circuits And Systems Magazine*, vol. **5**, no:2, pp. 36–45.

[24] **Tamás ROSKA,** 2004, CNN Technology for Brain-like Spatial-temporal Sensory Computing–Present and Future, *ISCAS-2004 Plenary Lecture*, Vancouver.

[25] **www.analogic-computers.com**

[26] Bi-i Vision System: User's Manual.

[27] **K. Laszlo, F. Ziliani, T. Roska, and M. Kunt**, 1998. Early segmentation in video compression using CNN processors, *Proceedings of the 1998 Fifth International Workshop on Cellular Neural Networks and Theirs Applications (CNNA'98)*, London, UK, pp. April 14–17, 175–180.

[28] **A. Stoffels, T. Roska and L. O. Chua,** 1997. Object-oriented image analysis for very-low-bitrate video-coding systems using the CNN Universal Machine, *Int. J. Circuit Theory Applic.*, vol. **25**, pp. 235-258.

[29] **Giuseppe Grassi, Eugenio Di Sciascio, Pietro Vecchio Luigi A. Grieco**, 2006. New Object-Oriented Segmentation Algorithm Based On The CNN Paradigm, *IEEE Trans.on Circuits and Systems - II: Express Briefs*, 2495/Sc., Volume **53**, Number 4, page 259 -263.

[30] **T. Szirànyi, K. Làszlò, L. Czùni, and F. Ziliani**, 1999. Object oriented motion segmentation for video-compression in the CNN-UM, *J. VLSI Signal Processing*, vol. **23**, pp. 479–496.

[31] **Paolo Arena, Adriano Basile, Maide Bucolo, and Luigi Fortuna,** 2003. An Object Oriented Segmentation on Analog CNN Chip, *IEEE Transactions On Circuits And Systems—I: Fundamental Theory And Applications,* Vol. **50**, No. 7.

[32] **T. Roska, L. Kèk, L. Nemes, A. Zaràndy, M. Brendel, and P. Szolgay**, 1998. CNN Software Library: Template and Algorithms, Computer and Automation Institute, Hungarian Academy of Sciences, Eds. Budapest, Hungary: Hungarian Acad. Sci.

[33] **J. W. Tukey**, 1971. Exploratory Data Analysis, Addison, Wesley, Menlo Park.

ÖZGEÇMİŞ:

Recep Uslu 01.05.1983 Kırklareli/Vize doğumludur. İlk, orta ve lise eğitimini Kırklareli'nde tamamlamıştır. 2001 yılında girdiği Yıldız Teknik Üniversitesi Elektronik ve Haberleşme Mühendisliği Bölümü'nden 2005 yılında mezun olmuştur. Aynı yıl İstanbul Teknik Üniversitesi Elektronik ve Haberleşme Mühendisliği Bölümü Elektronik Mühendisliği programında yüksek lisans eğitimine başlamıştır.