İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DOKTORA TEZİ

HÜCRESEL SİNİR AĞLARI İÇİN KARARLI ŞABLON TASARIMI VE GÖRÜNTÜ İŞLEME UYGULAMALARI

Bilg. Yük. Müh. Selçuk SEVGEN
Bilgisayar Mühendisliği Anabilim Dalı

Danışman
Prof. Dr. Sabri ARIK

Temmuz, 2009

İSTANBUL

Bu çalışma 16/09/2009 tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Anabilim Dalı programında Doktora Tezi olarak kabul edilmiştir.

Tez Jürisi

Prof. Dr. Sabri ARIK (Danışman)
İstanbul Üniversitesi

Prof. Dr. Ahmet SERTBAŞ
İstanbul Üniversitesi

Prof. Dr. Vedat TAVŞANOĞLU
Yıldız Teknik Üniversitesi

Doç. Dr. A. Halim ZAİM
İstanbul Üniversitesi

Doç. Dr. Serdar ÖZOĞUZ
İstanbul Teknik Üniversitesi

ÖNSÖZ

Tez çalışmam boyunca yardımlarını ve desteğini esirgemeyen, yol göstermesi sayesinde bu tezin hazırlanmasında büyük emeği geçen çok değerli hocam Prof.Dr. Sabri ARIK'a en içten teşekkürlerimi sunarım.

Bu çalışmam sırasında yardımlarını benden esirgemeyen Prof.Dr. Vedat TAVŞANOĞLU ve Doç.Dr. A.Halim ZAİM hocalarıma da teşekkürlerimi ve saygılarımı sunmak isterim.

Çalışmalarım esnasında bana çok yardımcı olan mesai arkadaşım Araş.Gör. Fethullah KARABİBER'e de teşekkür etmeyi borç bilirim.

Bu çalışma boyunca bana desteklerinden ötürü TUBİTAK – BİDEB'e de teşekkürlerimi sunarım.

Temmuz, 2009 **Selçuk SEVGEN**

İÇİNDEKİLER

ŞEKİL LİSTESİ

TABLO LİSTESİ

ÖZET

HÜCRESEL SİNİR AĞLARI İÇİN KARARLI ŞABLON TASARIMI VE GÖRÜNTÜ İŞLEME UYGULAMALARI

Bu tez çalışmasında bir HSA çok fonksiyonlu makineyi içinde barındıran Bi-i Hücresel görü sistemi incelenmiş ve bu sistem üzerinde çalışan IAŞT (**I**terative **A**nnealing ile **Ş**ablon **T**asarımı) olarak adlandırılmış şablon tasarımı gerçekleştiren bir yazılım geliştirilmiştir. Bu yazılımın geliştirilmesi ve şablonların tasarlanması için Iterative Annealing optimizasyon yöntemi kullanılmıştır. Geliştirilen yazılımın verdiği sonuçların doğruluğu kenar belirleme işlemi ile test edilmiştir. İkili ve gri seviyeli görüntüler üzerinde kenar belirleyebilen şablonlar eğitilmiştir. Daha sonra ACE16k yongası üzerinde çalışabilecek köşe belirleme şablonu eğitilmesi işlemi yapılmıştır. İç bükey ve dış bükey köşeleri ayrı ayrı bulabilen şablonlar eğitilmiştir. Geliştirilen yazılım, köşe belirleme işlemi aracılığıyla bir diğer şablon tasarım yazılımı olan CNNOPT ve HSA tabanlı olmayan iki adet köşe belirleme yöntemi ile karşılaştırılmıştır (Harris, He ve Yung). Karşılaştırma örnek görüntüler üzerinde köşe belirleme sonuçlarına ve işlem zamanlarına göre yapılmıştır.

Ayrıca tez çalışmasında, Ace16k yongası üzerinde çalışan bir nesne sayma uygulaması gerçekleştirilmiştir. Gri seviyeli giriş görüntüleri içinde bulunan nesneleri sayan bu uygulama öncelikle görüntüleri ikili hale çevirmekte, görüntü içindeki gürültüleri temizlemekte ve sonrasında nesneleri dörtgene tamamlayıp, IAŞT yazılımı ile eğitilmiş olan sol-üst köşe bulma şablonu kullanılarak her bir nesne bir nokta ile gösterilebilecek hale getirmektedir. Bu noktaların sayılmasıyla verilen görüntüdeki nesne sayısı bulunmaktadır. Uygulama hem DSP hem de MATLAB üzerine uygulanarak yürütme zamanları karşılaştırılmıştır.

Bu tez çalışmasında elde edilen sonuçlar ve yapılan uygulamalar gerçek zamanlı işlem yapabilen analog HSA çok fonksiyonlu makinenin etkinliğini göstermektedir. İçinde HSA çok fonksiyonlu makine bulunan Bi-i Hücresel görü sistemi görüntü işleme konusunda oldukça yüksek performans göstermektedir. Şablon kullanarak köşe belirleme işlemini HSA tabanlı olmayan yöntemlere göre 100 kat ve nesne sayma işleminin de MATLAB ortamına göre 60 kat daha hızlı yapabilmesi, ACE16k sisteminin görüntü işleme konusunda çok uygun bir platform olduğunu ve daha kapsamlı uygulamaların da rahatlıkla gerçekleştirilebileceğini göstermektedir.

SUMMARY

STABLE TEMPLATE DESIGN FOR CELLULAR NEURAL NETWORKS WITH APPLICATIONS TO IMAGE PROCESSING

In this thesis, Bi-i Cellular Vision System including a CNN-UM is examined and a template design software is developed on this system. Iterative Annealing optimisation method is used during development of the software and design of templates. Corner detection is used to test the accuracy of outputs of the software. Edge detection templates are trained on binary and grey level images. Then, corner detection template training is realized on ACE16k chip. Concav and convex corner detection templates are obtained. Developed software is compared with an another CNN based template training software called CNNOPT and two non-CNN based corner detection methods (Harris, He and Yung). Execution times and outputs of corner detection process are used for this comparison.

Besides, an object counting algorithm working on ACE16k chip is realized. This algorithm counts objects in grey level input images. Firstly, grey level images are converted to binary form and noises on the binary image are eliminated. Then, objects are transformed into rectangular shaped ones and each object is represented by one pixel using North-West corner detection template trained with developed software. Number of objects in an input image are calculated by counting these pixels. Execution times are compared by implementing the software on DSP and also on MATLAB.

Obtained results and developed applications show the efficiency of real-time working CNN-UM. Bi-i cellular vision system has a great performance in image processing tasks. Corner detection by using template is 100 times faster than non CNN based methods and object counting on ACE16k is 60 times faster than its simulation on MATLAB. These results indicate that ACE16k is a suitable platform for image processing and that it can realize more complex applications.

1. GİRİŞ

Bilimin pek çok alanında karşılaşılan problemlerin çözümü kullanılan bilgisayar sistemlerinin hesaplama gücüne dayanmaktadır. Ancak gittikçe karmaşıklaşan problemlerin çözülmesi, çok büyük sistemlerin benzetimi ve yüksek miktarlardaki verilerin işlenmesi daha fazla işlem gücüne gereksinim duymaktadır. Klasik sayısal mimarilerin gereken bu gücü sağlamayı sürdürmesi mümkün değildir. Yeni hesaplama mimarilerinin geliştirilmesi ihtiyaç duyulan işlem gücünü sağlamak için gereklidir.

Hesaplama kavramı 1936'da Alan Turing'in ortaya koyduğu soyut makine ve bu yapıya eklenmiş Von Neumann mimarisinden bu yana değişmeden günümüze kadar gelmiştir [1]. Genel olarak programın ve verinin bir bellek içerisinde saklandığı bu yapıda aritmetik ve lojik birimlerde komutlar sıralı olarak çalıştırılır [2]. Mikroişlemci devrinin başlamasıyla birlikte bu yapının kullanıldığı bilgisayarlar ucuzlayarak her alanda kullanılmaya başlanmıştır. Mikroişlemcilerin geliştirilmeleri sayesinde de bilgisayarların hızları sürekli artmış, boyutları ve enerji tüketimleri de azalmıştır. Ancak artık günümüzde mikroişlemci üretim tekniklerinin sınırlarına ulaşmaya başlaması mikroişlemcileri hızlandırmak için çok çekirdekli sistemlerin kullanılmasına neden olmuştur.

Son yılların gözde araştırma konularından olan *Görüntü İşleme* klasik bilgisayar sistemleri için oldukça zorlayıcı bir çalışma alanıdır. Ne kadar hızlı ve ne kadar çok çekirdekli de olursa olsun klasik bir bilgisayar sistemi, kameralar, alıcılar, çeşitli elektronik ve mekanik sistemler gibi pek çok bileşenden gelen analog sinyallerin işlenmesi için yetersiz kalmaktadır. Yeni yaklaşımların kullanılması kaçınılmazdır. İşte bu noktada, paralel hesaplama kavramı ön plana çıkmıştır. IBM'in ortaya koyduğu "Cell" isimli çok çekirdekli işlemci yapısı bu kavramın temsilcilerindendir [3]. Ancak Hücresel işlemci dizileri (HİD) adı verilen, daha basit işlemcilerden oluşan, yüksek paralel çalışma potansiyeline sahip ve belli bir iş yapma özelliği olan yapılar özellikle görüntü işleme uygulamaları için çok daha uygun bir seçenek haline gelmiştir. Bu

yapılarda bir yonga içerisinde bulunan pek çok işlemci birbirleriyle yerel olarak bağlantılıdır. Hücresel Sinir Ağlarının (HSA) uygulandığı Hücresel Sinir Ağı çok fonksiyonlu makine (CNN-UM : Cellular Neural Networks Universal Machine), HİD yapısının özel bir örneğidir. Çeşitli teknikleri kullanarak gerçek zamanlı görüntü işleyebilen sistemlerin geliştirilmesi artık mümkün hale gelmiştir.

Bi-i Hücresel görü sistemi (Bi-i inspired Cellular Vision System), HSA çok fonksiyonlu makine içeren bir sistemdir [4]. ACE16k adlı bir HSA tabanlı bir analog işlemci ile sayısal işaret işlemek için kullanılan özel bir sayısal işaret işlemcisine sahiptir. ACE16k işlemcisi ile görüntü işleme uygulamalarını çok hızlı bir şekilde yapmak mümkündür.

HSA çok fonksiyonlu makine sistemlerinin dayandığı Hücresel Sinir Ağları 1988 yılında Chua ve Yang tarafından geliştirilmiştir [7] [8]. Bir HSA, hücreler ve hücreler arasındaki yerel bağlantılardan oluşmaktadır. Hücreler bir ızgara (grid) yapısı oluşturmaktadır. Her bir hücre belli bir komşuluk değeriyle diğer hücrelere bağlıdır. Geliştirilmesinden sonra çeşitli HSA modelleri ortaya konmuştur. Bu modeller hücre karmaşıklığı, parametreler, hücre dinamikleri ve ağ yapısı gibi özelliklerle birbirlerinden ayrılırlar.

HSA'ların en önemli bileşenlerinden bir tanesi *şablondur (template)*. Şablon, bir HSA'nın çalışmasını düzenleyen parametreler setidir. HSA üzerinde herhangi bir uygulama geliştirilmesinin en önemli bölümü, uygun şablon değerlerinin elde edilmesidir. HSA şablonları programlama dillerindeki komutlar gibidir. HSA'nın yapacağı işlemleri belirlerler. Dolayısıyla günümüze kadar şablonların elde edilmesine yönelik pek çok yöntem geliştirilmiştir. İlk aşamalarda şablon elde edilmesi için benzetim sistemleri kullanılmıştır. SCNN [24], CANDY [59], MARGE [57], MATCNN [56] ve CADETWin [58] gibi birçok benzetim sistemi gerçekleştirilmiştir. Daha sonra HSA çok fonksiyonlu makine sistemlerinin geliştirilmesi ile şablonların HSA çok fonksiyonlu makine sistemleri üzerinde elde edilmelerine yönelik çalışmalar yapılmıştır [6] [30].

Bu tez çalışmasında HSA'ların en önemli bileşenlerinden olan şablonların tasarlanması üzerinde çalışılmıştır. IAŞT (**I**terative **A**nnealing ile **Ş**ablon **T**asarımı) isimli Iterative Annealing [19] optimizasyon yöntemine dayanan şablon eğitim yazılımı geliştirilmiştir. IAŞT yazılımının ürettiği şablonların çalışması kenar belirleme eğitimi yapılarak test edilmiştir. Siyah/beyaz ve gri seviyede sonuçlar elde edilmiştir. Daha sonra ACE16k için mevcut olmayan köşe bulma şablonunun eğitimine çalışılmıştır. İç bükey ve dış bükey olmak üzere 2 farklı şablon eğitimi yapılmıştır. Bulunan şablon değerleri Bi-i Hücresel görü sistemi üzerinde çeşitli giriş görüntülerine uygulanarak elde edilen sonuçlar literatürde bulunan köşe bulma tekniklerinin sonuçlarıyla karşılaştırılmıştır. Bu tekniklerin de bulunduğu performans değerlendirmesi yapılmıştır. Ayrıca başka bir şablon eğitim yazılımı olan CNNOPT [6] ile köşe bulma şablon sonuçları karşılaştırması ve performans değerlendirmesi gerçekleştirilmiştir. Elde edilen köşe bulma şablonları kullanılarak Bi-i Hücresel görü sistemi üzerinde uygulama geliştirilmiştir. Bu uygulamanın sonuçları verilmiştir.

Tez çalışmasının bölümleri şu şekildedir: 2. Bölümde, HSA ve HSA çok fonksiyonlu makine hakkında temel bilgiler verilmiştir. 3. Bölümde, şablon tasarımı üzerine genel bilgiler verilmiş, kullanılan Iterative Annealing ve bu yöntemin dayandığı Simulated Annealing optimizasyon yöntemleri ile Bi-i Hücresel görü sistemi tanıtılmıştır. 4. Bölümde, geliştirilen IAŞT yazılımı anlatılmıştır. Bu yazılım kullanılarak gerçekleştirilen kenar ve köşe belirleme eğitimleri sonucunda elde edilen şablon setleri ile bu setlerin örnek görüntüler üzerine uygulanmasıyla bulunan sonuç görüntüleri verilmiştir. Bulunan sonuçlar ve geliştirilen yazılımın etkinliğinin karşılaştırmaları yapılmış ve geliştirilen uygulama gösterilmiştir. Sonuçların değerlendirilmesi 5. Bölümde yapılmıştır.

2. GENEL KISIMLAR

Bu bölümde Hücresel Sinir Ağları (HSA) ve Hücresel Sinir Ağları çok fonksiyonlu makine (CNN-UM) hakkında bilgiler verilecektir. HSA'nın yapısına ilişkin temel kavramlar Bölüm 2.1'de açıklanacaktır, Bölüm 2.2 ise, HSA çok fonksiyonlu makine hakkında bilgiler vermektedir.

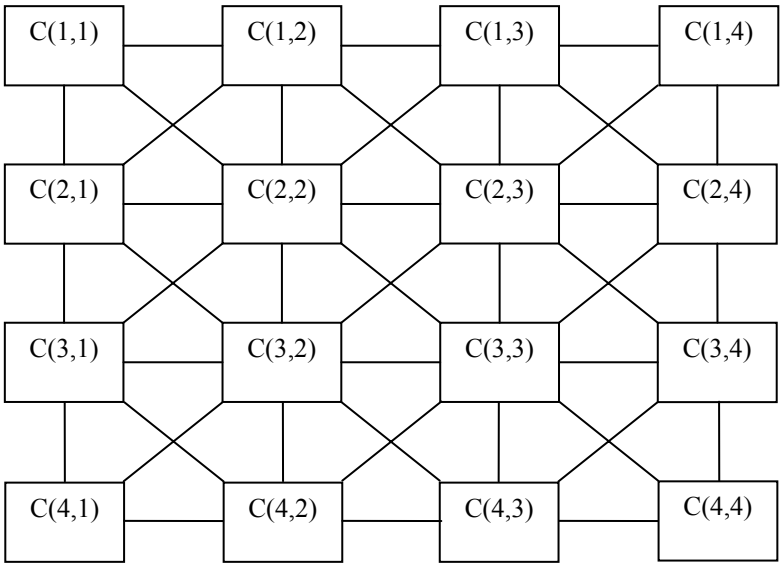
2.1. HÜCRESEL SİNİR AĞLARI

Hücresel Sinir Ağları (HSA), Chua ve Yang tarafından 1988 yılında önerilmiş olan yeni bir mimaridir [7] [8]. Paralel sinyal işleme, sürekli-zaman dinamikleri ve gerçek zamanlı sinyal işleme gibi yapay sinir ağlarının önemli özelliklerini içinde barındırmaktadır. HSA'ların bir diğer önemli özelliği de sadece komşularıyla iletişim halinde bulunan analog sinyal işleyebilen hücre yapısına sahip olmasıdır. Tüm hücrelerin birbirleriyle bağlantılı olmaları yerine sadece belli bir komşulukta bulunan hücrelerin birbirleriyle doğrudan, diğer hücrelerle ise dolaylı bağlantılı olması HSA devre yapısının karmaşıklığını azaltmakta, enerji tüketimini düşürmektedir. Bu sayede diğer klasik yapay sinir ağları modellerine göre donanımsal gerçeklemesi daha kolaydır.

2.1.1. Standart HSA Modeli

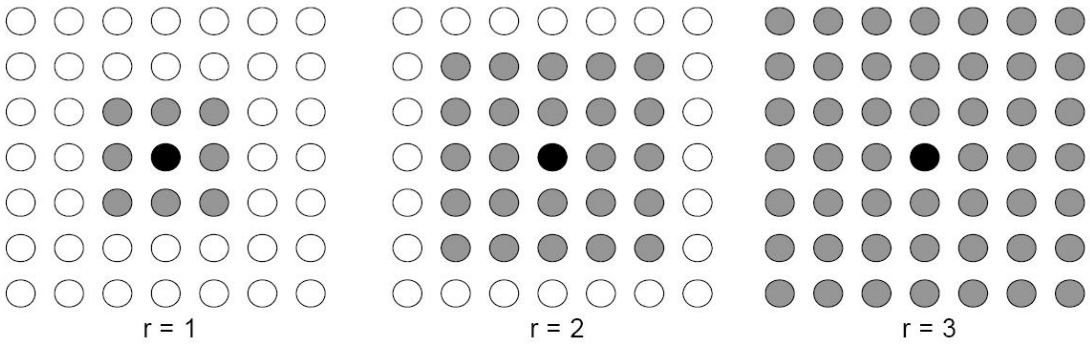
Standart bir HSA mimarisi $M \times N$ boyutlu hücreler dizisinden oluşmaktadır [7]. i. satır ve j. sütunda bulunan bir hücre $C(i, j)$ ile ifade edilmektedir ($i = 1,2,3,...,M \quad j = 1,2,3,...,N$). Her hücre sadece kendisine komşu olan hücrelerle bağlantılıdır ve komşu hücreler birbirleriyle direk etkileşimdedirler (Şekil 2.1). Bir hücresel sinir ağında $C(i, j)$ hücresinin *r*-komşuluğu aşağıdaki ifadeyle tanımlanabilir:

$$S_r(i,j) = \left\{ C(k,l) \;\middle|\; \begin{array}{c} maks \\ 1 \le k \le M, 1 \le l \le N \end{array} \{ |k-i|, |l-j| \} \le r \right\} \tag{2.1}$$



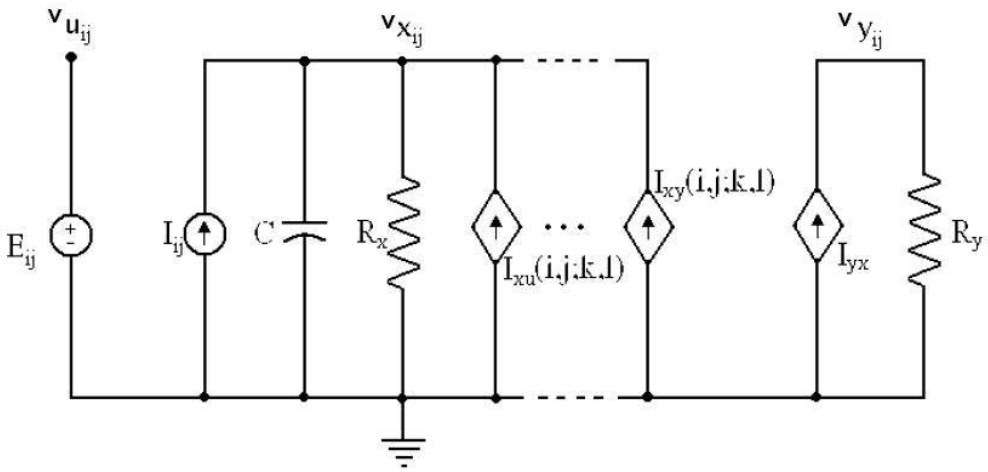
Şekil 2.1 : Standart HSA yapısı

$S_r(i, j)$, bir hücrenin *r* yarıçaplı etki alanı (sphere influence) olarak adlandırılır. *r*, bir pozitif tamsayı olmak üzere komşuluk derecesi olarak isimlendirilir. Hücresel sinir ağı tanımlanırken bu *r* değeri seçilir. Şekil 2.2, sırasıyla merkezde siyah ile gösterilen hücrenin $r = 1$, $r = 2$ ve $r = 3$ komşuluklarını göstermektedir. $r = 1$ komşuluğu için merkez hücrenin komşuları gri ile işaretlenmiş olan 8 hücredir ve 3×3 komşuluk olarak tanımlanır. Benzer şekilde $r = 2$ komşuluk değeri, 5×5 komşuluk ve $r = 3$ komşuluk değeri de 7×7 komşuluk olarak tanımlanabilir.



Şekil 2.2 : Çeşitli *r* değerleri için komşuluklar

Standart bir HSA hücresinin devre yapısı Şekil 2.3'te gösterilmektedir. Hücre içerisinde sırasıyla giriş, durum ve çıkışa karşı düşen *u, x* ve *y* düğümleri bulunmaktadır. v_{uij} düğüm gerilimi, *C(i,j)* hücresinin girişi, v_{xij} düğüm gerilimi, *C(i,j)* hücresinin durumu ve v_{yij} düğüm gerilimi, *C(i,j)* hücresinin çıkışı olarak isimlendirilmektedir. HSA'nın her bir *C(i,j)* hücresi, bir lineer kapasitör *C*, bir bağımsız akım kaynağı E_{ij}, iki adet direnç (R_x ve R_y), bir bağımsız akım kaynağı I_{ij} ve *m* komşu hücre sayısına eşit olmak üzere en fazla *2m* adet $I_{xu}(i,j;k,l)$ ve I_{xy} *(i,j;k,l)* gerilim kontrollü lineer akım kaynağından oluşmaktadır.



Şekil 2.3 : Standart HSA hücresinin devre yapısı

C(i,j) hücresinin x_{ij} durum değişkeni, I_{ij} bağımsız akım kaynağı, lineer *C* kapasitörü, lineer R_x direnci ve $I_{xu}(i,j;k,l)$ ve I_{xy} *(i,j;k,l)* gerilim kontrollü lineer akım kaynakları ile tanımlanır. Bu $I_{xu}(i,j;k,l)$ ve I_{xy} *(i,j;k,l)* kaynakları komşu hücrelerin gerilim değerleri ile kontrol edilirler ve böylece hücrelerin birbirlerine bağlanmaları (coupling) gerçekleşmiş olur. Lineer $I_{xu}(i,j;k,l)$ akım kaynaklarının komşu hücrelerin u_{kl} gerilimleriyle, $I_{xy}(i,j;k,l)$ akım kaynaklarının ise komşu hücrelerin y_{kl} çıkış gerilimlerinden gelen geri besleme ile kontrol edildiği iki tür bağlantı mecvuttur:

$$I_{xy}(i,j;k,l) = A(i,j;k,l)y_{kl} \tag{2.2}$$

$$I_{xu}(i,j;k,l) = B(i,j;k,l)u_{kl} \tag{2.3}$$

$A(i,j;k,l)$ geri besleme parametreleri ve $B(i,j;k,l)$ giriş (kontrol) parametreleri hücreler arasındaki etkileşimlerin kontrol edilmeleri ve değiştirilmeleri için kullanılabilirler.

Standart bir HSA matematiksel olarak aşağıdaki denklemle ifade edilebilir:

$$C\frac{dx_{ij}(t)}{dt} = -\frac{1}{R_x}x_{ij}(t) + \sum_{C(k,l)\in S(i,j)} A(i,j;k,l)y_{kl}(t) + \sum_{C(k,l)\in S(i,j)} B(i,j;k,l)u_{kl} + I_{ij} \qquad (2.4)$$

$$y_{ij} = f(x_{ij}) = \frac{1}{2}(|x_{ij}+1| - |x_{ij}-1|)$$

Bu denklemde,

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

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

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

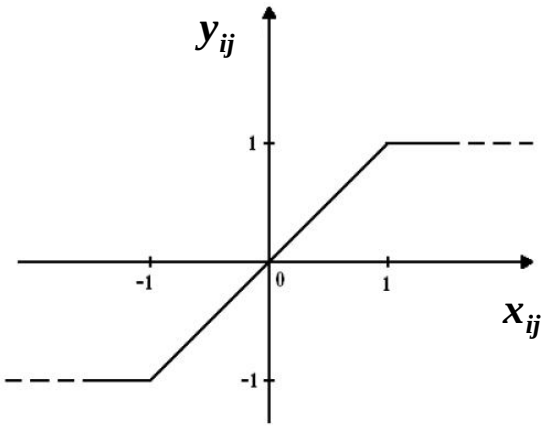
$I_{ij} \in R$; Eşik değeri veya hücre akımı,

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

B(i, j ; k, l) ; Giriş (Kontrol) operatörü,

olarak ifade edilmektedir.

Çıkış fonksiyonu y_{ij}, lineer olmayan ve hücrenin çıkışını tanımlayan karakteristik parçalı (piece-wise) bir fonksiyondur. Şekil 2.4'te bu fonksiyon görülmektedir:



Şekil 2.4: HSA'nın karakteristik parçalı çıkış fonksiyonu

$A(i,j;k,l)$, $B(i,j;k,l)$ ve I_{ij} parametreleri hücreler ve aralarındaki bağlantıları tanımlamaktadırlar. Bu parametreler *şablon* (*template*) olarak isimlendirilirler. Şablonlar değişik işlemlerin HSA üzerinde gerçekleştirilmelerini sağlamaktadırlar.

2.1.2. HSA Şablonları

(2.4) eşitliğinde görüldüğü gibi, bir hücrenin durumu kendisine komşu olan hücrelerle arasındaki ağırlıkların ilişkilerine bağlıdır. Bu parametrelerin $\{ A(i,j;k,l)$, $B(i,j;k,l)$, $I_{ij} \}$ oluşturduğu yapıya *şablon* adı verilir.

HSA'ların hücrelerinin sadece komşularıyla bağlantılı olması devre yapısını oldukça basitleştirmesinin yanında başka bir önemli özelliği de *A*, *B*, *I* ifadelerinin uzayda değişmez (space invariant) olabilmesidir. Bu özellik sayesinde, *A*, *B* ve *I* ifadelerinin farklı değerler almasıyla değişik HSA işlemleri gerçekleştirmek mümkündür. Pek çok durumda HSA'larda uzayda değişmez şablonlar kullanılır. Bu durum, $A(i,j;i+1,l)$ gibi bir ifadenin tüm *i* ve *j* koordinatları için aynı olması şeklinde açıklanabilir. Bu şekilde, iki boyutlu bir HSA yongası üstünde, tüm $A(i,j;k,l)$ bağlantıları 3×3 bir şablon ile tanımlanabilir. Benzer şekilde $B(i,j;k,l)$ bağlantıları da 3×3 bir şablon ile ifade edilebilir. Tüm sistemi bu iki şablon ve eşik değeri ile tanımlamak mümkündür. Toplam 19 parametreden oluşan bu {*A*, *B*, *I*} yapıyı (9 adet *A* değeri, 9 adet *B* değeri ve 1 adet eşik değeri) aşağıdaki gibi gösterilebilir:

$$A=\begin{bmatrix} a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ a_{0,-1} & a_{0,0} & a_{0,1} \\ a_{1,-1} & a_{1,0} & a_{1,1} \end{bmatrix},\quad B=\begin{bmatrix} b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & b_{0,0} & b_{0,1} \\ b_{1,-1} & b_{1,0} & b_{1,1} \end{bmatrix},\quad I \tag{2.5}$$

A ve *B* ifadelerinde $a_{0,0}$ ve $b_{0,0}$ değerleri merkez (center), geri kalanlar ise kenar (surround) şablon değerleri olarak isimlendirilirler. Bazı durumlarda *A* ifadesi aşağıdaki gibi ayrılabilir:

$$A = A^0 + \overline{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a_{0,0} & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ a_{0,-1} & 0 & a_{0,1} \\ a_{1,-1} & a_{1,0} & a_{1,1} \end{bmatrix} \tag{2.6}$$

A^0, merkez ve $\overline{A}$ kenar şablonları olarak isimlendirilirler. Benzer şekilde *B* ifadesi de yazılabilir [13]:

$$B = B^0 + \overline{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & b_{0,0} & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & 0 & b_{0,1} \\ b_{1,-1} & b_{1,0} & b_{1,1} \end{bmatrix} \tag{2.7}$$

Şablon elemanlarının sayısı komşuluk derecesi *r*'ye bağlıdır. Yukarıda ifade edilen 19 parametreli şablon $r = 1$ komşuluk değeri içindir. Eğer komşuluk değeri $r = 2$ olursa, 51 adet şablon parametresinin olması gerekirdi. Buradan HSA şablon uzayının sonsuz sayıda şablon içerdiğini söylemek mümkündür. Bununla birlikte, HSA yapısı içerisinde önemli olan 3 şablon sınıfını kısaca açıklamak gerekir:

- Sıfır-Geri besleme (Zero-feedback) Şablonu: Tüm geri besleme değerleri sıfırdır. Bu tür bir HSA hücresi aşağıdaki denklemle ifade edilebilir:

$$\dot{x}_{ij} = -x_{ij} + B * u_{ij} + I \tag{2.8}$$

- Sıfır-Giriş (Zero-Input) Şablonu: Tüm kontrol değerleri sıfırdır. Geniş bir alanda kullanılan bir şablon türüdür. Bu tür bir HSA hücresi aşağıdaki denklemle ifade edilebilir:

$$\dot{x}_{ij} = -x_{ij} + A * y_{ij} + I \tag{2.9}$$

- Bağlantısız (uncoupled) Şablon: tüm kenar kontrol değerleri sıfırdır. Yani $A = A^0$. Bu tür bir HSA hücresi aşağıdaki denklemle ifade edilebilir:

$$\dot{x}_{ij} = -x_{ij} + a_{00} f(x_{ij}) + B * u_{ij} + I \tag{2.10}$$

2.2. HÜCRESEL SİNİR AĞI ÇOK FONKSİYONLU MAKİNE

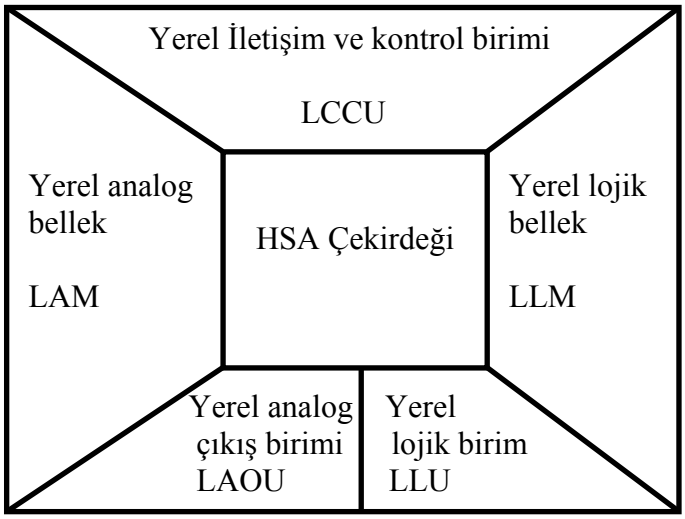
Hücresel Sinir Ağlarının teorisinin 1988 yılında Chua ve Yang tarafından ortaya konmasından sonra, Hücresel Sinir Ağı çok fonksiyonlu makine (CNN-UM) mimarisi 1993 yılında geliştrilmiştir [9]. Bu makine analog ve lojik mimariye ve program depolama yeteğine sahiptir. HSA'ların pek çok başarılı gerçeklemesinden [10] sonra, ilk HSA evrensel yongası 1994'te duyurulmuştur [11]. Duyurulmasından bu zamana kadar da ACE4k [12], ACE16k [4] ve EYE-RIS [5] gibi pek çok HSA çok fonksiyonlu makine yongası geliştirilmiştir.

2.2.1. HSA Çok Fonksiyonlu Makine Mimarisi

HSA çok fonksiyonlu makine sistemlerinin en önemli özelliklerinden biri hem analog hem de lojik işlemleri gerçekleştirebilmesidir. Analog işlem dizileri sadece EVET/HAYIR değişkenlerini içeren lojik işlemlerle birleştirilmiştir. Bu nedenle *analojik* mimari olarak isimlendirilmiştir. Mimari yapı içerisindeki hücre yapısı Şekil 2.5'te görülmektedir. Bu yapı içerisindeki tüm kısımlar Şekil 2.3'teki devreye benzer bir yapı içermektedir. Aradaki fark bu devrede bellekleri yönetmek için gerekli anahtarların varlığıdır. Hücreler analog ve lojik iki birimden oluşmaktadır. Hücreler yerel analog belleklere (Local Analog Memory : LAM) ve yerel lojik belleklere (Local Logic Memory : LLM) sahiptirler. LAM belleklerde gerçek değerler yani gri nokta değerleri, LLM belleklerde ise ikili değerler saklanır. LAM bellek üzerinde toplama, çıkarma gibi işlemler yapılabilirken, LLM bellek üzerinde lojik işlemler kolaylıkla yapılabilmektedir. Ayrıca sonuçta elde edilen görüntüler LAM ve LLM bellekler arasında iletilebilmektedir. Algoritmaların gerçekleştirilmesi esnasında pek çok işlem birden fazla şablon kullanılarak gerçekleştirildiğinde, ara sonuçların saklanması bu yerel bellekleri çok önemli hale getirmektedir. Yerel analog çıkış birimi (Local Analog Output Unit : LAOU) değişik LAM belleklerde bulunan analog değerleri tek bir değere dönüştüren aygıttır. Yerel lojik birimi (Local Logic Unit : LLU) ise LAOU 'nun yaptığı işlemleri lojik ifadeler için yapan birimdir. Yerel iletişim ve kontrol birimi de (Local Communicaiton and Control Unit : LCCU), global analog programlama biriminden (Global Analog Programming Unit : GAPU) program komutlarını alan birimdir [13]. GAPU sistem içerisindeki her bir hücreyi kontrol eden ve programlayan birimdir. Lojik komutlar, hüce çekirdeği için anahtarlama konfigürasyonu ve şablon değerleri gibi

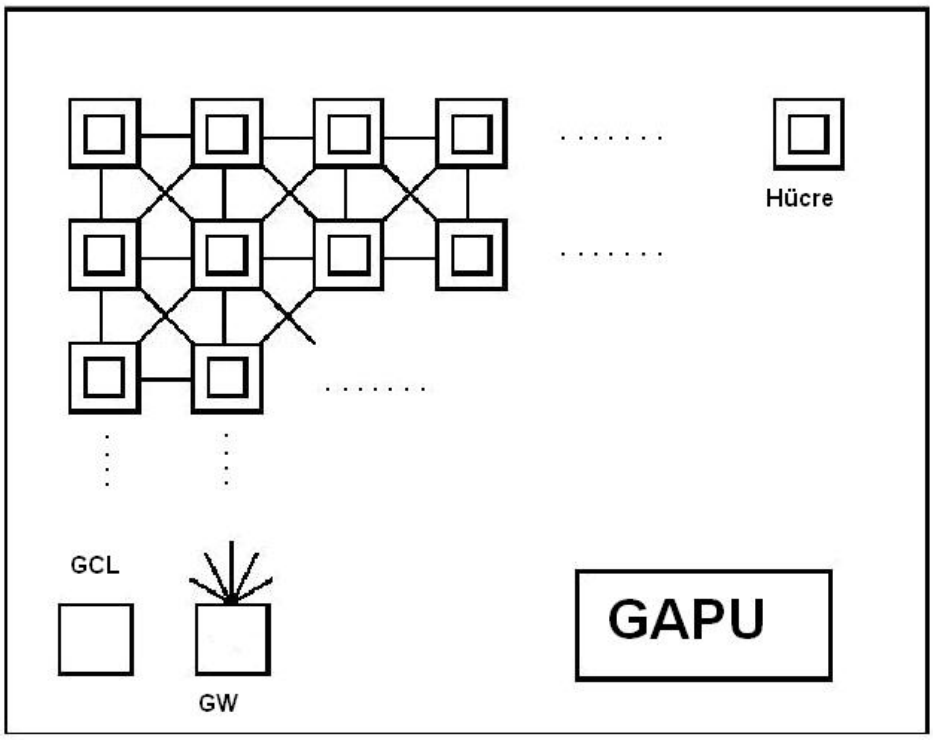
komutları hücrelere iletir. Bu nedenle 3 adet yazmaca sahiptir (Şekil 2.6'da tüm yapı görülmektedir.):

- Analog program yazmacı (Analog Program Register : APR)
- Lojik program yazmacı (Logic Program Register : LPR)
- Anahtar konfigürasyon yazmacı (Switch Configuration Register : SCR)



Şekil 2.5 : Hücre Yapısı

APR yazmacı şablonları saklar, lojik fonksiyonlar da LPR yazmacında bulunur ve SCR yazmacında ise anahtarların kontrol edilmesini sağlayan ve LCCU tarafından düzenlenen konfigürasyonlar bulunur. Bu yazmaçları kontrol eden ve işlemleri düzenleyen GAPU içerisindeki global analojik birimdir (Global Analogic Control Unit : GACU). Hücreler komşularıyla birbirlerine bağlı oldukları gibi doğrudan global birimle de bağlantılıdırlar. Ayrıca hücre içinde gerçekleşen işlemler çevrimlerle yürütüldüğü için bu işlemleri düzenleyecek bir de global saate ihtiyaç vardır [14].



Şekil 2.6 : HSA çok fonksiyonlu makine yapısı

HSA çok fonksiyonlu makine sistemlerinin bu zamana kadar çok sayıda gerçeklemesi yapılmıştır. Tarihsel olarak gelişimi aşağıdaki tabloda görülebilir:

Tablo 2.1 : HSA çok fonksiyonlu makine gelişim süreci

İsim	**Yıl**	**Boyut**
	1993	12×12
ACE440	1995	20×22
POS48	1997	48×48
ACE4K	1998	64×64
CACE1K	2001	$32 \times 32 \times 2$
ACE16K	2002	128×128
XENON	2004	128×96
EYE-RIS	2007	176×144

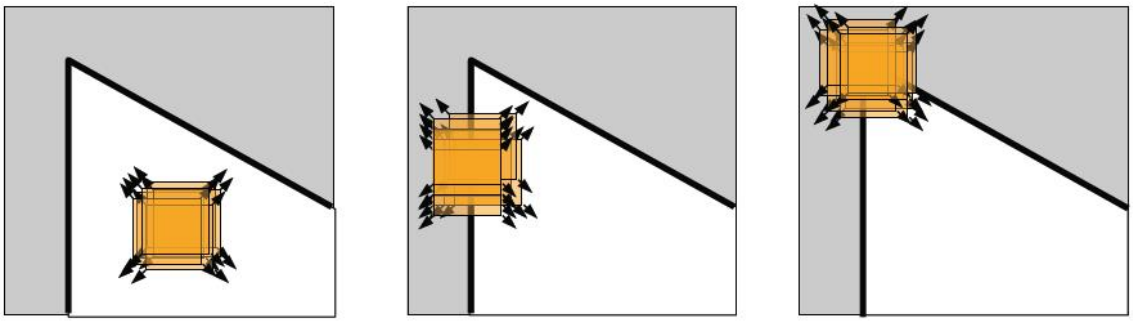
Bu sistemler görüntü işleme, robotik ve algılayıcıya (sensor) sahip sistemler gibi alanlarda uygulamalar geliştirmek için kullanılmaktadır [13]. Özellikle son yıllarda Tablo 2.1'den görülebileceği gibi yongaların boyutları sürekli artmıştır. Örneğin EYE-RIS [5] 176×144 adet işlemciye sahiptir ve her hücre 4 adet algılayıcıyı desteklemektedir.

Bu tez çalışmasında Bi-i Hücresel görü sistemi içerisinde bulunan ACE16k yongası kullanılmıştır. Bi-i özellikle görüntü işleme uygulamaları için geliştirilmiş bir sistemdir. İçerisinde HSA yongası ile bağlantılı olan yüksek hızlı bir sayısal sinyal işlemcisi de bulunmaktadır. Resim algılayıcılarıyla beraber tüm sistem hızlı bir kamera olarak kullanılabilir. Saniyede binlerce kare görüntü yakalayabilen sistem bunları HSA yongası üstünde gerçek zamanlı olarak işleyebilmektedir [15]. Bi-i Hücresel görü sistemi ve ACE16k yongası hakkında ayrıntılı bilgi sonraki bölümlerde verilecektir.

2.3. KÖŞE BELİRLEME (CORNER DETECTION)

Köşe belirleme, görüntü işleme ve görüntü tanıma sistemlerinde önemli bir araştırma alanıdır. Özellikle içerik tabanlı multimedya indeksleme ve geri getirmede, köşe bulma en güçlü tekniklerden biridir. Görüntü içinde köşe bulmak, bölütleme (segmentation), görüntü tanıma işlemleri gibi alanlarda kullanılmaktadır. Başarılı bir köşe belirleme aracı tüm gerçek köşeleri yüksek bir doğrulukla bulabilmelidir. Bununla birlikte köşe olmayan noktaları yok etmeli ya da minimize etmelidir. Gürültüye hassas olmalıdır. Çözünürlük, yer değiştirme değişiminden ve ölçeklemeden etkilenmemelidir [49] [53].

Bir köşede, herhangi bir yönde hareket yoğunlukta büyük bir değişime sebep olur:



a- Düz bölge: Tüm yönlerde bir değişim olmaz.

b- Kenar: Kenar yönünde değişim olmaz.

c- Köşe: Her yönde değişimler olur.

Şekil 2.7: Köşe Gösterimi

Köşelerin önemli avantajlarından bir tanesi ölçekleme, döndürme gibi dönüşümlerden etkilenmemeleridir. Köşe açısı nesnenin döndürülmesi durumundaki değişimden etkilenmez. Köşenin bulunduğu yer ile nesnelerin pozisyonunu belirtilebilir. Kenar belirleme ve nesne tanımada köşelerin bu özelliklerinden faydalanılabilir. Köşelerin aksine kenarlar yoğunluk değişimlerinden etkilenmezler ancak diğer görüntü dönüşümlerinden etkilenirler [51].

Köşe bulma araçları genel olarak geometri tabanlı ve şablon tabanlı olmak üzere ikiye ayrılabilir. Geometri tabanlı yaklaşım daha çok kullanılmaktadır. Geometrik özellikleri analiz ederek köşeleri belirleyen bu yaklaşımda görüntünün kenarları çıkarılır ve yerel maksimum eğim derecesine sahip noktalar köşe olarak bulunur. Bu tür yöntemlerin dezavantajı açı ve yerleşim gibi köşelerin özelliklerini kaydetmemeleridir. Şablon

tabanlı yöntemler ise köşeleri belirlemek için görüntü şablonları kullanırlar. Bu işlemi de görüntü ve kayıtlı şablonları karşılaştırıp aradaki benzerliği ölçerek gerçekleştirirler. Bu yaklaşımın dezavantajı ise fazla miktardaki hesaplamalar nedeniyle işlem performansının düşük olabilmesidir [52] [53].

Köşe belirleme işleminin nasıl yapılabileceğini görmek için Moravec'in geliştirdiği yöntem [54] incelendiğinde, bir görüntü üzerinde olası bir köşeyi içeren bir bölge ile bu bölgenin değişik yönlerde ötelenmiş hali arasındaki karelerin farkının toplamı (SSD : Sum of Squared Differences) hesaplanır.

$$E(x,y) = \sum_u \sum_v w(u,v)(I(u,v) - I(u-x, v-y))^2 \tag{2.11}$$

w, görüntü penceresi şeklinde ifade edilebilir: değeri dörtgenin içi ise 1, aksi halde 0'dır.

Eşik değerini aşan *min{E}* içerisindeki yerel maksimum değeri köşeyi ifade eder.

Harris [48], SSD denkleminin ikinci türevine dayanan bir hesaplama yaparak Moravec'in yöntemini geliştirmiştir. 2. seviye Taylor yaklaşımını kullanarak (2.11) denklemi aşağıdaki hale gelir:

$$E(x,y) \approx \frac{1}{2}(x,y) A \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.12}$$

A, kısmi türev terimlerini içerir, $I_x = \partial I / \partial x$, $I_y = \partial I / \partial y$:

$$A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{2.13}$$

A'nın özdeğerleri λ_1 ve λ_2 aşağıdaki gibi hesaplanabilir:

$$\lambda_1 = \tfrac{1}{2}(a + c - \sqrt{(a-c)^2 + 4b^2})$$

$$\lambda_2 = \tfrac{1}{2}(a + c + \sqrt{(a-c)^2 + 4b^2})$$

Kareköklerin hesaplanması karmaşık olacağından, köşe karşılık fonksiyonu (corner response function) ile işlem yapmak daha pratik olacaktır:

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \tag{2.14}$$

k, deneysel olarak bulunur.

R nin değeri sonucun ne olacağını gösterir:

- Eğer *R*, pozitif büyük bir sayı ise, sonuç köşedir.
- Eğer *R*, negatif büyük bir sayı ise, sonuç kenardır.
- Eğer *R*'nin mutlak değeri sıfıra yakın küçük bir sayı ise, sonuç düz bölgedir.

Bir eşik değeri uygulandıktan sonra, *R*'nin yerel maksimumları köşeler olarak bulunmuş olur.

3. MALZEME VE YÖNTEM

Bu bölümde, HSA'ların en önemli bileşenlerinden olan şablonların tasarlanması ile ilgili bilgiler Bölüm 3.1'de verilecektir. Bölüm 3.2'de bu çalışmada şablon tasarımı için kullanılan optimizasyon yönteminden ve Bölüm 3.3' te kullanılan Bi-i Hücresel görü sisteminden bahsedilecektir.

3.1. ŞABLON TASARIMI

Hücresel Sinir Ağları günümüzde geleneksel mimarilere alternatif olan oldukça güçlü ve etkili bir mimaridir. Bu mimariyi kullanan HSA çok fonksiyonlu makineler süper bilgisayarlarla yarışabilecek hesaplama gücüne sahiptirler. HSA'ların yerel bağlantı yapısı, analog ve sayısal sinyal işleme özelliği ve tek bir yonga içerisine yerleştirilebilen çok sayıda HSA hücre dizisi gibi özellikler bu hesaplama gücünün oluşmasına katkıda bulunmaktadır. Ayrıca analog devre yapısı HSA çok fonksiyonlu makine sistemlerinin hızını ve işlem gücünü arttıran özelliklerdendir [20]. Ancak bu analog devre yapısı sistemin en zayıf noktalarından biridir. Üretim sırasında oluşan parametre sapmalarına karşı çok hassastır. Dolayısıyla HSA işlemleri sırasında bu sapmalardan oluşacak hataların ihmal edilmesi mümkün değildir [21].

HSA'lar üstünde gerçekleştirilen işlemler için şablonlar kullanılmaktadır. HSA kullanarak istenen bir uygulamanın yapılabilmesi için en uygun şablon parametrelerinin bulunması en önemli işlemlerdendir. HSA'ların davranışı tamamen bu şablon parametrelerine bağlıdır. Bu nedenle şablon parametrelerinin belirlenmesi en önemli araştırma alanlarından biri haline gelmiştir.

Şablonların tasarlanması ya da optimize edilmesi süreçlerinde dikkat edilmesi gereken noktalardan biri de yukarıda bahsedilen bu parametre sapmalarıdır. Bu sapmalar nedeniyle şablonların HSA çok fonksiyonlu makine sistemlerinde çalışmaları esnasında elde edilen sonuçlarda bozulmalar oluşabilir. Dolayısıyla kullanılan şablon tasarım

yöntemi HSA'nın robust olarak çalışmasını garanti etmelidir. Bir HSA'nın robust olması gerçeklemeden kaynaklanan hatalara maruz kaldığında bile HSA'nın istenildiği gibi işlem yapabilmesi olarak tanımlanabilir [22]. Bu bölümde HSA çok fonksiyonlu makine gerçeklemelerinde kullanılan şablonların parametre sapmaları dikkate alınarak tasarlanmasından bahsedilecektir.

İstenilen işlemi gerçekleştirecek şablon değerlerini elde etmek için değişik yöntemler mevcuttur. Bu yöntemler arasında en zor olanı sezgisel olarak tasarım yapmaktır. Deneyimli tasarımcılar için hızlı bir yöntemdir ancak istenen sonuca ulaşılabileceğinin garantisi yoktur [28]. Şablon tasarımı için bir ikinci yöntem ise öğrenmedir. Klasik yapay sinir ağları yöntemleri kullanılabilir. Günümüze gelene kadar çeşitli yerel ve global öğrenme algoritmaları ile uğraşılmıştır [32] [33]. Ancak buradaki sorun herhangi bir uygulama için uygun şablonun hiçbir zaman bulunamama ihtimalidir. Üçüncü bir tasarım yöntemi ise istenen fonksiyonun ne olduğunun belirlenmesine dayanan doğrudan şablon tasarımıdır. Bu yöntem ile gerçekleştirilmesi gereken işlem için bir veya birden fazla şablon bulunabilir. Bu şablon kümesi en robust olan şablonu da içerir. Diğer iki yönteme göre çok daha az zaman ve hesaplama gücü gerektirir [28].

3.1.1. Hatalı HSA Çok Fonksiyonlu Makine Davranışları

Üretim süreçlerinde oluşan değişimler ile sıcaklık değişimi ve elektriksel gürültü gibi çevresel etkenler, HSA çok fonksiyonlu makine gerçeklemelerinde oluşabilecek hatalı davranışların en önemli nedenleridir. Bununla birlikte şablon parametrelerinin alabileceği değer aralıkları da sınırlıdır [4]. Yukarıdaki bu nedenlere ek olarak, ideal HSA yapıları için geliştirilmiş bazı şablonların aynı giriş ve çıkış görüntüleri ile farklı sonuçlar verdiği de görülmüştür. Bunun nedeni HSA çok fonksiyonlu makine sistemine yüklenen hatalı ya da gürültülü giriş/başlangıç durumu görüntüsüdür. Yukarıda bahsedilen bu hatalı davranış nedenlerini şu şekilde özetlemek mümkündür [21]:

- Üretim süreçlerinden kaynaklanan parametre sapmaları
- Hücrelerin bileşenlerinden kaynaklanan gürültü
- HSA çok fonksiyonlu makine sistemine yüklenen hatalı ya da gürültülü giriş/başlangıç durumu görüntüsü
- Sıcaklık değişimleri

Parametre sapmalarının HSA çok fonksiyonlu makine gerçeklemeleri üzerine etkisi konusunda, Tetzlaf bir benzetim sistemi gerçekleştirmiştir [23]. SCNN adını verdiği evrensel benzetim sistemini [24] kullanarak HSA yongasının ve şablon değerlerinin gürültü karşısındaki davranışlarını simüle etmiştir. Geliştirilen bu model HSA Şablon Kütüphanesindeki [25] farklı şablonlara uygulanmıştır. Elde edilen sonuçlar çok küçük değişimlerin bile çıkış görüntülerini çok farklı hale getirebildiğini göstermiştir.

3.1.2. Yongadan Bağımsız Şablon Tasarımı

Robust şablonların tasarımı için analog HSA çok fonksiyonlu makine gerçeklemelerinin önceki bölümde bahsedilen etkilerinden kaçınan ya da en aza indirmeye çalışan pek çok yöntem geliştirilmiştir [28, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43]. Tanım olarak robust olma, çıkış değerlerinin üretilmesi esnasında, HSA şablonlarının değerlerinin değişimine karşı gösterdiği hassasiyettir. Robust şablonlardan, değerlerinde meydana gelebilecek değişimler karşısında değerlerini doğru işlem aralığının ortasında tutabilmeleri beklenir. Bu aralığın boyutu şablonun ne kadar robust olacağını tanımlar. Bu nedenle robust olma seviyesi yapılacak işlemin tipine bağlıdır. En robust işlemler genellikle parametre sapmalarını dikkate almadan HSA çok fonksiyonlu makine sistemleri üzerinde çalışırlar. Bu tür şablon tasarım yöntemleri özel HSA çok fonksiyonlu makine gerçeklemelerini dikkate almadan ideal HSA modellerini temel alarak geliştirilirler. Dolayısıyla, yongadan bağımsız yöntemler, robust şablonları üretirken belirli bir HSA çok fonksiyonlu makine yongasının parametrelerini değil de ideal HSA modellerini kullanan tasarım ya da öğrenme yöntemleridir [20].

Yongadan bağımsız yöntemlere örnek olarak ayrık zaman (discrete-time) [44] [45] ve sürekli zaman (continuous-time) [38] [46] HSA gerçeklemeleri verilebilir.

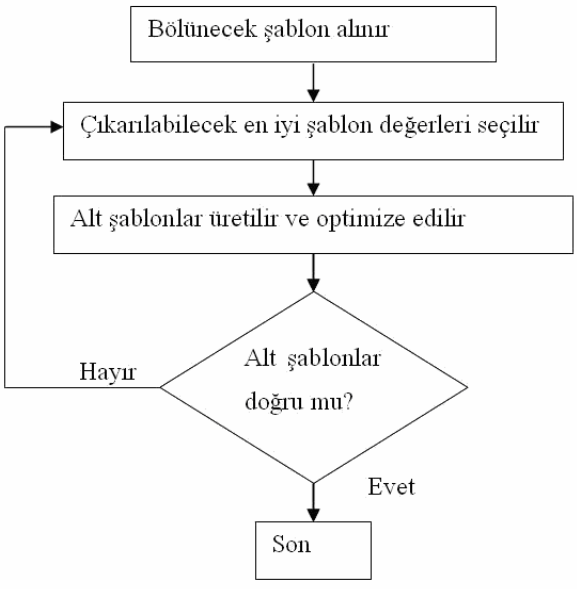
3.1.3. Yongaya Özel Şablon Tasarımı

Yongadan bağımsız yöntemlerin temel dezavantajı, yeterince robust olan bazı şablonların değişik uygulamalar üzerinde farklı robust olma dereceleri göstermeleridir. Ayrıca bu tür yöntemlerle üretilen birçok robust şablon düşük robust olma dereceli işlemlerde hatalı HSA çok fonksiyonlu makine davranışlarına engel olamayabilir. Bir HSA çok fonksiyonlu makine sistemi içerisinde bulunan yongadaki iki hücrenin en robust şablon kullanılsa dahi birbirlerine eş bir şekilde davranacaklarının garantisi

verilemez. Eğer bir HSA çok fonksiyonlu makine sistemi yongasının parametre sapması şablonun tolerans aralığından daha büyük olursa o şablonun düzgün çalışması mümkün olmayacaktır. Dolayısıyla, şablonların farklı yongalar üzerinde aynı robust davranışları gösterebileceklerini söylemek mümkün değildir. Yonga üzerinde işlem gerçekleştirebilmek için kullanılan yongadan kaynaklanan hataların yok edilmesi ya da en aza indirgenmesi gerekir. Bu işlemi kullanılan yonga üzerinde şablonların elle ya da deneysel olarak düzenlenmesi (tune) ile gerçekleştirmek mümkündür. Ancak bu oldukça fazla zaman alacak bir işlemdir. Dolayısıyla, yonga gerçeklemelerinin farklı özelliklerini dikkate alacak şekilde şablonların elde edilmesi gerekmektedir. Yongaya özel ya da yongaya bağlı yöntemler, HSA şablonlarını belirli bir yonga için üreten yöntemlerdir.

Yongaya özel yöntemlerden bir tanesi Földesy'nin geliştirdiği bir yaklaşımdır [22]. Bu yaklaşım En Küçük Ortalamalı Kareler (LMS: least mean squares) optimizasyon yöntemine dayanmaktadır. Temel amaç, belirli bir HSA çok fonksiyonlu makine sistemi üzerinde düzgün bir şekilde çalışabilen mevcut HSA şablonlarına eşdeğer şablon ya da şablonlar dizisi bulabilmektir. Bu yöntem, tek bir şablon ile gerçekleştirilemeyecek işlemlerin olduğu durumlarda, şablonların bölünmesi (decomposition) yaklaşımını kullanmaktadır.

Bölünme işlemi aşağıdaki gibi gerçekleşmektedir:



Şekil 3.1: Şablon bölünmesi algoritması

Bu yöntem doğru şablonların üretilmesini sağlayabilir. Ancak şablon bölünmesi işlemlerinin sayısı çok fazla da olabilir. Çünkü optimizasyon yöntemi global en iyiye ulaşılabilmesini garanti edemez. Ayrıca bu yöntem sadece bağlantısız (uncoupled) şablon durumunu dikkate almaktadır.

Yukarıda bahsedilen bu yaklaşımın global en iyiye ulaşmadaki zayıflığı şablonların bölünmesi sonucunu getirmektedir. Simulated Annealing [18] gibi bir global yöntemin uygulanması ile bu sorun ortadan kalkabilir ve bağlantılı şablonların da üretilmesi mümkün olabilir.

3.2. ITERATIVE ANNEALING OPTİMİZASYON YÖNTEMİ

Bu çalışmada, Bi-i Hücresel görü sistemi üzerinde çalışan IAŞT (**I**terative **A**nnealing ile **Ş**ablon **T**asarımı) isimli bir şablon eğitim yazılımı geliştirilmiştir. Bu yazılımın programlanması esnasında şablon eğitimi için bir optimizasyon yöntemi olan *Iterative Annealing* (IA) [19] kullanılmıştır. IA yöntemi, HSA için geliştirilmiş bir çeşit *Simulated Annealing* (SA) [18] yöntemidir. Bu kısımda her iki optimizasyon yöntemi hakkında bilgiler verilecektir ve IA yönteminin şablon tasarımında nasıl kullanıldığı anlatılacaktır.

3.2.1. Simulated Annealing Optimizasyon Yöntemi

Simulated Annealing (SA), bir çeşit yerel arama tekniğidir [18]. Herhangi bir yerel arama tekniğinden farkı ise yerel en iyilere de sıçrama olasılığının olmasıdır. Yani, normal bir yerel arama işleminde daha iyi bir sonuç ortaya çıkmadığı sürece ilerleme yapılamaz. Oysa SA belli olasılıklarla kötü sonuçları da kabul etme imkanı tanır. Böylelikle yerel en iyiden sıçrayarak global en iyiye ulaşmak mümkün olabilmektedir. Bu özellik SA'yi başlangıç noktasından bağımsız bir arama yöntemi haline getirmektedir [26].

Annealing işlemi metallerin, camın ya da kristalin yüksek sıcaklıkla eritilmesi sonra da yavaş yavaş katı hale gelene kadar soğutulmasıdır. Amacı minimum enerjili bir sonuç durumuna ulaşmaktır. Fiziksel materyaller yüksek enerji seviyelerinden düşük olanlara geçerler. Ancak belli bir olasılıkla daha yüksek enerji seviyesine de geçiş olabilir. Bu olasılık şu şekilde ifade edilebilir [27]:

$$p = e^{-\Delta E / kT} \tag{3.1}$$

ΔE : enerji değişimi, T: sıcaklık ve k: sabittir. Yukarı doğru bir hareket olasılığı aşağı doğru olandan düşüktür. Ayrıca yukarı hareketler sıcaklık düşüşünü de azaltacaktır. Bu hareketler genelde sıcaklığın yüksek olduğu işlemin ilk zamanlarında meydana gelir.

Sistemin soğuma hızı *annealing çizelgesi (schedule)* olarak isimlendirilmiştir. Eğer çok hızlı bir soğuma olursa yüksek enerji seviyelerinde kararlı durumlar oluşur, yani yerel

en iyilere ulaşılır. Yavaş bir soğumayla global en iyiye ulaşılabilir ancak fazla zaman alacaktır. En uygun annealing *çizelgesi* deneysel olarak belirlenmelidir.

SA algoritması aşağıdaki gibi ifade edilebilir:

S başlangıç çözümü alınır.
T>0 başlangıç sıcaklığını alınır.
Soğuma gerçekleşmediği sürece döngüye devam edilir.
    L defa aşağıdaki işlemi gerçekleştirilir.
        Rastgele bir *S'* oluşturulur. *S* nin komşusu olmalıdır.
        ΔE =maliyet(*S'*) – maliyet(*S*)
        $\Delta E \leq 0$ ise,
            $S = S'$
        $\Delta E > 0$ ise,
            $e^{-\Delta E / kT}$ olasılığı ile $S = S'$
    $T = rT$
S döndürülür.

Başlangıç çözümü rastgele seçilebilir ya da mevcut bir çözüm kullanılabilir. $T = rT$ sıcaklığın düşürülmesi içindir. *r* sabit bir sayı olup 0-1 aralığındadır. Değeri deneysel olarak belirlenir.

Yukarıda verilen algoritma SA'nin genel olarak elde edilmiş algoritmasıdır. SA yönteminin en önemli özelliklerinden birisi, parametrelerinin problemden probleme değişebilmesidir. Bu özelliği sayesinde değişik problemlere uygulanması mümkündür.

3.2.2 Iterative Annealing Optimizasyon Yöntemi

Iterative Annealing, SA'e dayanan ve HSA için geliştirilmiş bir optimizasyon yöntemidir. Yöntemin algoritması aşağıdaki şekildedir [19]:

1. $x_0^k, s_{\max}, j_{\max}, T_0, \tau, j = 0$ başlangıç değerleri seçilir,
2. $\nu = (\tau / T_o)^{j_{\max} / s_{\max}}$ adım boyutu seçilir,
3. $T = T_0$, $i = 0$
4. $y_i^k = x_i^k + u^k.T$ (u^k : [-0.5, 0.5] aralığında rastgele sayılar)
5. Eğer $f(\vec{y}_i) < f(\vec{x}_i)$ ise, $\vec{x}_{i+1} = \vec{y}_i$
6. $T = \nu.T$ (sıcaklık düşürülür)
7. $i = i + 1$
8. Eğer $i < (s_{\max} / j_{\max})$ ise 4.adıma geri dön
9. $j = j + 1$
10. Eğer $(j < j_{\max})$ ise 3.adıma geri dön.

$f(\vec{x})$ fonksiyonu minimize edilecek hatayı ifade etmektedir ve *x,* parametre vektörüdür. $s_{\max}$, maksimum yineleme sayısı ve $j_{\max}$ ise algoritmanın çalıştırılma sayısıdır. T_0, başlangıç sıcaklığıdır, minimum sıcaklık τ ile ifade edilmektedir. Minimum sıcaklığa ulaşılana kadar her adımda sıcaklığın düşürülmesi arama bölgesinin daralmasını sağlar. Minimum sıcaklığa ulaşıldığında tekrar başlangıç sıcaklığı ile algoritma çalıştırılır. Global bir minimum bulunmasına kadar devam edilir.

3.2.2.1. Iterative Annealing ile HSA Çok Fonksiyonlu Makine Üzerinde Eğitim

Iterative Annealing yöntemi HSA çok fonksiyonlu makine sistemini kullanarak bilgisayar üstünde çalışabilecek şekilde değiştirilmiştir. Değiştirilen bu algoritma Bi-i görü sistemi içindeki ACE16k yongası üzerinde çalıştırılmıştır. Eğitim sürecinde ACE16k yongası harici bir işlem ünitesi olarak değişik şablon setleri için çıkış görüntülerini elde eder. IA iki adet döngüye sahiptir. İç döngü annealing kısmını içerirken dıştaki döngü yinelemeli davranışı kontrol etmektedir [29].

Yonga üzerinde elde edilen çıkış görüntüsü (O) ile istenen çıkış görüntüsü (R) arasındaki hata minimize edilmek istenen fonksiyondur. Bu fonksiyon aşağıdaki şekilde ifade edilebilir [30]:

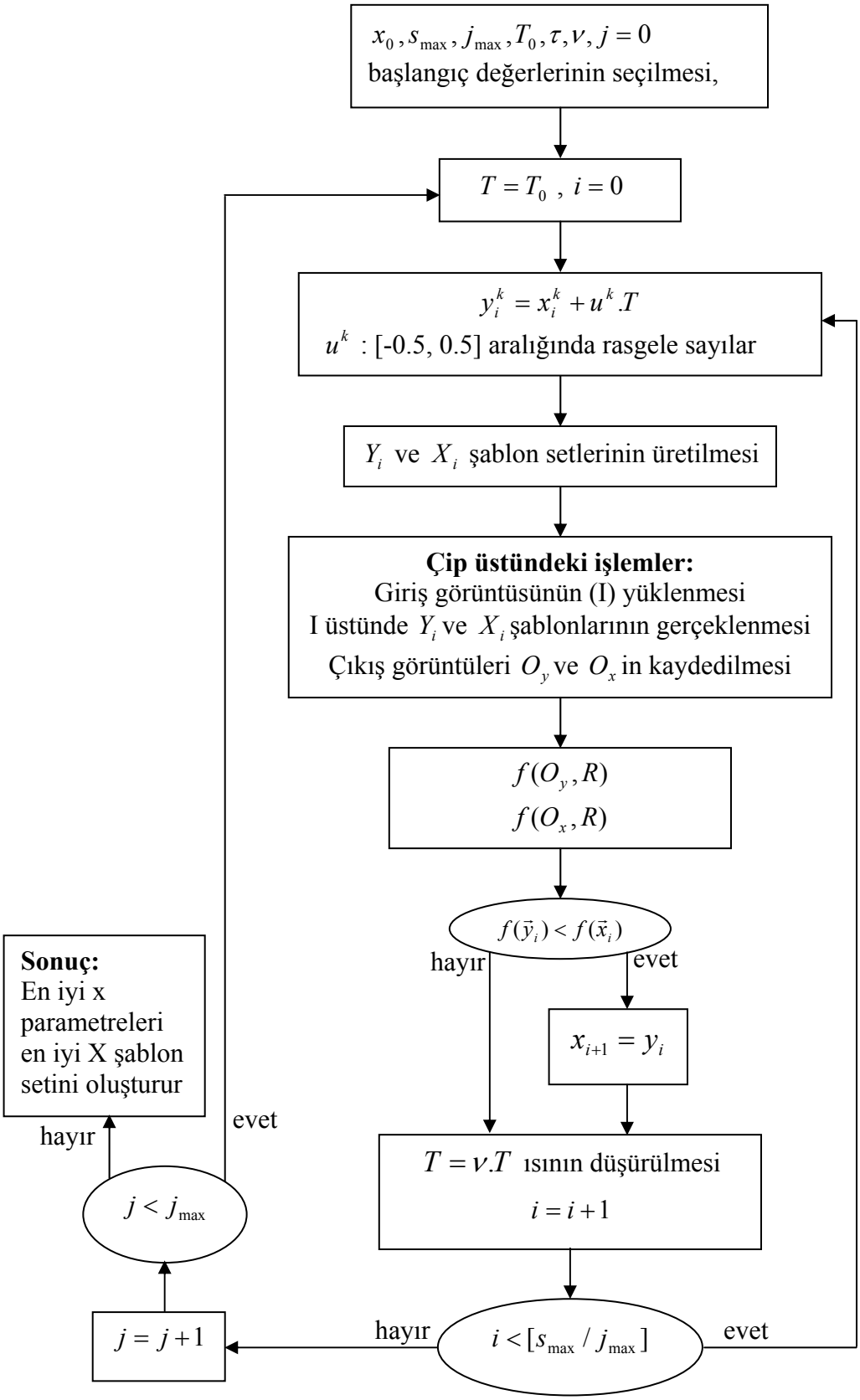
$$f(R,O) = \frac{\sum_{i=1}^{a} \sum_{j=1}^{b} \left| r_{i,j} - o_{i,j} \right|}{g * a * b} \tag{3.2}$$

a ve b görüntü boyutlarını, $r_{i,j}$ ve $o_{i,j}$ nokta değerlerini gösterir. g, hata değerini 0 – 1 aralığına getirmek için kullanılır.

IA yönteminin ACE16k yongası ile çalışabilmesi için algoritmanın içteki döngüsüne bazı adımlar eklenmiştir [31]:

1) Parametre vektörüne bağlı olarak üretilen şablonlar doğrudan yongaya yüklenir ve çalıştırılır.
2) Yonga üzerinde elde edilen çıkış görüntüleri istenen çıkış görüntüsü ile arasındaki hatanın hesaplanması için kaydedilir. Yonga üzerinde çalışan IA algoritması Şekil 3.2'de bulunan blok diyagram ile gösterilmiştir.

Algoritma rasgele seçilmiş değerler ile başlamaktadır. x değerleri kullanılarak rasgele üretilmiş sayılarla birlikte y değerleri elde edilir. Bu x ve y değerleri X ve Y şablon setlerini oluşturur. Elde edilen bu iki şablon ACE16k yongasına yüklenir. Giriş görüntüsüne uygulanan şablonların ürettiği çıkış görüntüleri kaydedilir ve istenen çıkış görüntüleri ile hata fonksiyonu kullanılarak karşılaştırılır. Elde edilen iki hata değerinin karşılaştırılması sonucunda da hangi sonuçla algoritmanın devam edeceği belirlenir. Daha sonra sıcaklık düşürülür. Minimum sıcaklığa ulaşılıncaya kadar iç döngüye devam edilir. İç döngü tamamlandıktan sonra başlangıç sıcaklığı ile algoritma yeniden çalıştırılır. En iyi şablon değerleri kaydedilerek işlem bitirilir.



Şekil 3.2: Yonga üzerinde çalışan IA algoritması

3.3. BI-I HÜCRESEL GÖRÜ SİSTEMİ

Bi-i çok hızlı, küçük boyutlu, bağımsız ve akıllı bir kameradır [15]. İki farklı tipte algılayıcıya sahiptir: Bunlardan biri 1.3 mega piksel çözünürlükte görüntü yakalayan CMOS algılayıcı, diğeri ise ultra hızlı ACE16k işlemcisidir.

Bi-i üzerinde bulunan 128×128 hücreden meydana gelen ACE16k, görüntülerin şablonlarla işlenmesi, lojik işlemler, toplama çıkarma gibi birçok görüntü işleme işleminin gerçekleştirilmesinde kullanılır.

Bi-i'ın temel işlemci elemanı yüksek performanslı bir sayısal işaret işlemcisidir (DSP : Digital Signal Processor). Ayrıca değişik çevre birimlerini destekleyen bir iletişim işlemcisine sahiptir. En önemli arabirimi 100 Mbit hızındaki ağ arabirimidir (ethernet). Sistem üzerinde çalıştırılacak programlar ağ arabirimi üzerinden yüklenir ve bilgisayarla ağ arabirimi üzerinden Bi-i' a bilgi yazabilir ya da Bi-i' dan bilgi okunabilir. Sistemin dışarıdan görünümü Şekil 3.3'te gösterilmiştir [16].

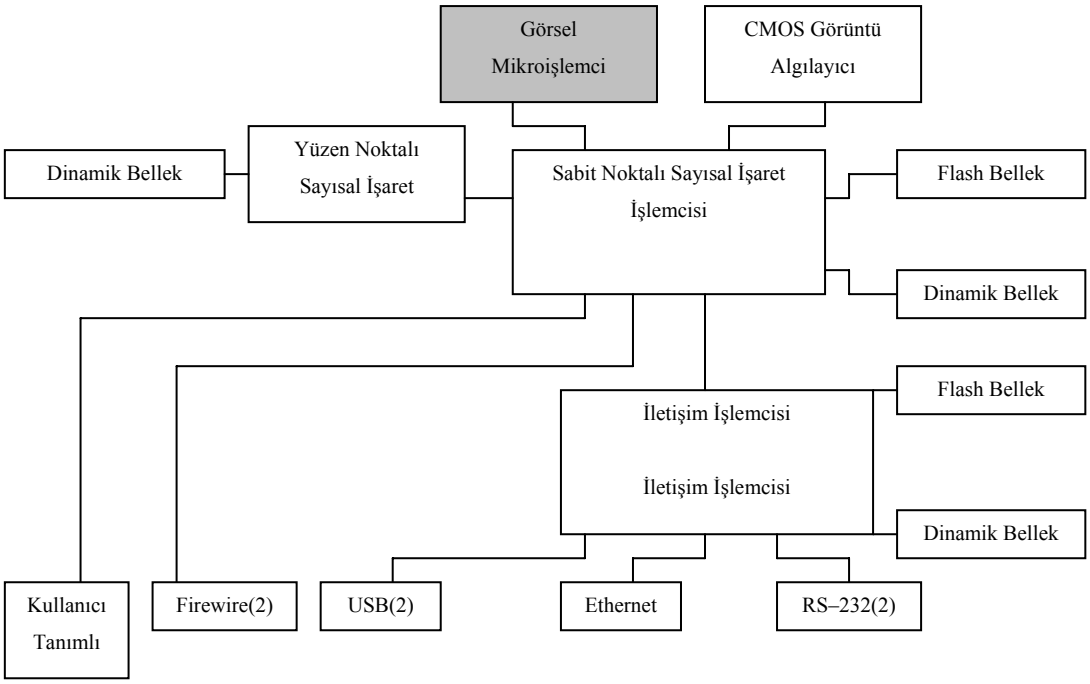
Bi-i'ın v1, v2, v301 olarak temsil edilen değişik sürümleri vardır. Ancak tüm sistemler için temel mimari aynıdır.

Şekil 3.3**:** Bi-i sisteminin dışarıdan görünüşü

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 bir TMS320C6202 model DSP'ye, Bi-i v2, 600 MHZ de çalışan TMS320C6415 model DSP'ye ve Bi-i v301 ise 600 MHZ – 1 GHZ de çalışan TMS320C6415 model DSP'ye sahiptir.

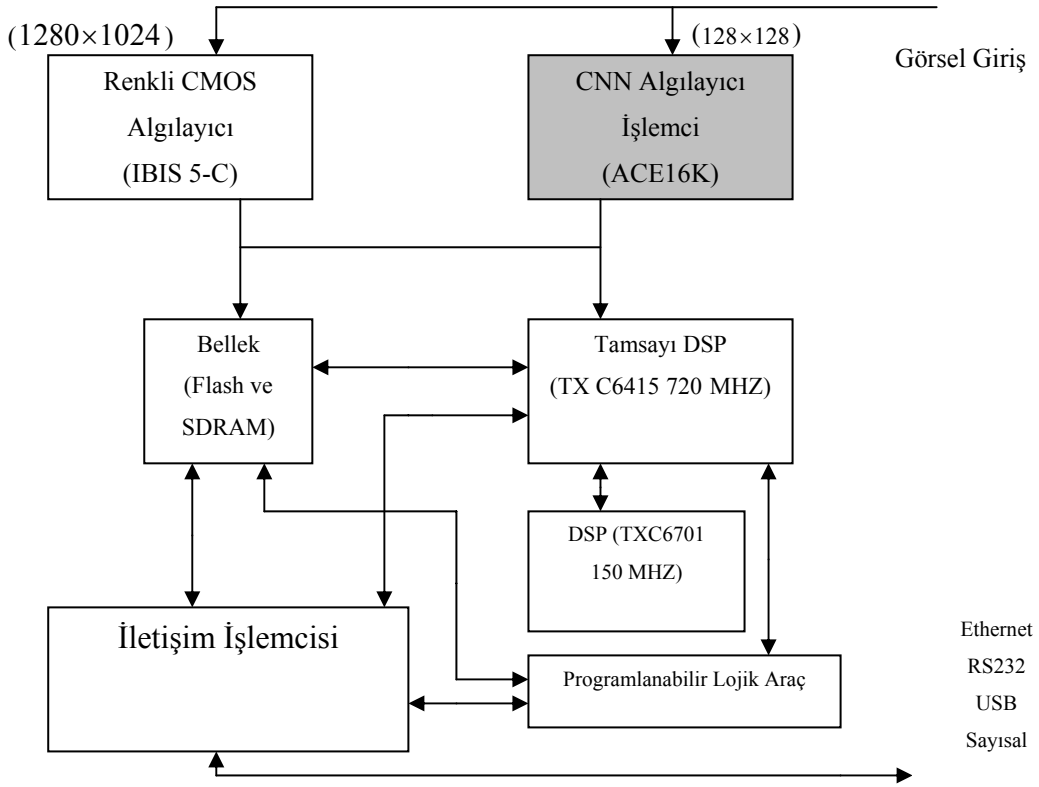
Bi-i sistemine görüntüler farklı kaynaklardan yüklenebilir. Bu kaynaklardan birisi 1.3 megapiksel çözünürlükte (1280*1024) renkli IBIS5 CMOS algılayıcıdır. Diğer görüntü kaynağı ise 16k piksel formatındaki (128*128) bir ACE16k görsel mikroişlemcidir. ACE16k, işlediği görüntüleri DSP ünitesine aktaran, sayısal olarak kontrol edilen bir HSA dizi işlemcisidir. ACE16k hızlı görüntü işleme uygulamaları için tasarlanmıştır.

Bi-i sisteminin mimari yapısı Şekil 3.4'de gösterilmektedir.



Şekil 3.4: Bi-i sisteminin mimari yapısı

Bi-i temelde HSA 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 3.5'te görülmektedir [15].



Şekil 3.5: Bi-i sistem mimarisinin temel bloklar halinde gösterimi

3.3.1 Bi-i Programlama

Bi-i sistemi üzerinde uygulama geliştirmek, şablonları çalıştırmak gibi işlemleri gerçekleştirebilmek için oluşturulmuş çeşitli programlama dilleri ve yazılım ortamları mevcuttur. Bunlar 3 başlıkta toplanabilir [17]:

I - AMC (Analogic Macro Code)

Bi-i için geliştirilmiş doğal bir programlama dilidir. AMC dilinde yazılan kodlar ikilik tabana dönüştürülür ve Bi-i üzerinde çalıştırılır. Program geliştirme AMC derleyici ile birlikte çalışan AMC söz dizim editörü tarafından gerçekleştirilir. AMC editörü ve AMC derleyicisi, Bi-i yazılım paketinin bir parçası olarak yüklenirler. AMC büyük projeler için tasarlanmamıştır ama küçük uygulamalarda oldukça etkilidir.

II - Bi-i SDK (Yazılım Geliştirme Aracı)

Bi-i uygulamaları geliştirmek için kullanılan *Instant Vision* isimli C++ kütüphanelerini içerir. Bu kütüphaneler bilgisayara yüklenen bir geliştirme arabirimi olan *Code Composer Studio* ile birlikte DSP için kullanılabilirler. Bi-i SDK ile AMC dilinde yapılabilecek yazılımlardan çok daha gelişmiş, büyük ölçekli ve optimize uygulamalar geliştirmek mümkündür.

III - API (Uygulama Programı Arabirimi)

Bi-i sistemiyle etkileşimli uygulamalar geliştirmeye yarayan bir yazılım arabirimidir. API kullanılarak geliştirilen uygulamalar bilgisayar üzerinde çalışmaktadırlar. Yani Bi-i sistemi içerisindeki ACE16K yongası kullanılmamaktadır. Bu tür uygulamalar geliştirmek için Microsoft Visual C++ gibi yazılım geliştirme araçları kullanılabilir.

3.3.2. ACE16k İşlemcisi

Bi-i görü sisteminin en önemli birimi iki boyutlu 128×128 özdeş hücreler dizisinden meydana gelen ACE16k yongasıdır. Görüntü işleme, görüntü bölütleme gibi işlemlerde kullanılan lojik ve analog işlemler ACE16k üzerinde paralel olarak çok hızlı bir şekilde gerçekleştirilebilmektedir. Bu hız bir IBM süper bilgisayarla karşılaştırılırsa;

- $10 * 10^{12}$ 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 doğrudan da alınabilir [4].

ACE16k'nın bir önceki sürümü 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 [12].

Yaklaşık 4 milyon transistörden meydana gelen ACE16k yongası ile ilgili Tablo 3.1'de bazı karakteristik özellikleri gösterilmiştir [4].

Tablo 3.1: HSA çok fonksiyonlu makine ın karakteristik özellikleri

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

3.3.3. Bi-i SDK (Yazılım Geliştirme Aracı)

Bu çalışmada gerçekleştirilen tüm uygulamalar Bi-i SDK kullanılarak gerçekleştirilmiştir.

Bi-i SDK içerisinde bulunan kütüphanelerin her birinin değişik işlevleri vardır. Bunlar içerisinde en önemlilerden biri olan TACE sınıfı, görüntü ve şablon transferi, lojik ve aritmetik işlemler ve şablon çalıştırmak gibi birçok fonksiyonu içerir. TACE_IPL ise ACE16k için oluşturulmuş bir görüntü işleme kütüphanesidir [17].

3.3.3.1. Giriş/Çıkış Arabirimi

ACE16k, şablon ve görüntü transferini DSP ile gerçekleştirmesi için 32-bit genişliğinde dijital bir arabirime sahiptir. 8 adet gri seviye (C_LAM1 - C_LAM8) ve 2 adet ikili seviye (C_LLM1 - C_LLM2) görüntü belleğine ve bununla birlikte 32 adet şablon (C_TEM1 - C_TEM32) belleğine sahiptir.

DSP de TBitMatrix olarak tanımlanan ikili görüntü ile TByteMatrix olarak tanımlanan gri seviye görüntüleri ACE16k'da tanımlanan fonksiyonlar veya operatörler kullanılarak transfer işlemi gerçekleştirilebilir. Ancak sadece 128×128 boyutundaki görüntüler ACE16k'ya transfer edilebilir.

3.3.3.2 Şablon Yapısı

Tace16ktem yapısı DSP belleğindeki bir şablonu göstermektedir. Bir veya birden fazla sayıda şablon yapısı aşağıda verilen formata uygun olarak bir metin dosyasına kaydedilir.

```
[Template_1]
Control = 9 1 -0.75 0 0.75 -1.5 0 1.5 -0.75 0 0.75
Feedback = 9 1 0 0 0 0 -6.0 0 0 0 0
Current = 3.0
Bias = 0
Mode = 3
References = 11 1 128 0 178 229 128 142 255 112 128 37 23
```

Köşeli parantez içerindeki sayı şablonu işaret etmektedir. Bir şablon aşağıdaki bilgileri içermektedir.

Control: 3×3 kontrol matrisi (B). İlk iki değer 9×1 eleman içerdiğini göstermektedir. Orta eleman hariç diğer elemanlar -3.0 ile 3.0 arasında olmalıdır. Ortadaki eleman ise -6.0 ile 6.0 arasında olmalıdır.

Feedback: 3×3 geri besleme matrisi (A). İlk iki değer 9×1 eleman içerdiğini göstermektedir. Orta eleman hariç diğer elemanlar -3.0 ile 3.0 arasında olmalıdır. Ortadaki eleman ise -6.0 ile 6.0 arasında olmalıdır.

Current: Akım değeri (I). Değeri -6.0 - 6.0 aralığında olmalıdır.

Bias: Uzaysal değişmeyen ofset görüntünün ağırlığı. Değeri -6.0 - 6.0 aralığında olmalıdır.

References: Şablon için tanımlanan 11 adet analog referans değerleri. İlk iki eleman 11×1 eleman içerdiğini göstermektedir.

Mode: Şablon çalıştırma kipi. 4 farklı kipi vardır.

Bunlar;

1. Geri beslemeli ve uzaysal değişen ofset değeri olmayan şablon.
   Geri besleme matrisinin bütün elemanları etkilidir. Kontrol matrisinin ise sadece orta elemanı etkilidir.
2. Geri beslemeli ve uzaysal ofset değeri olan şablon.
   1. kip ile aynı fakat şablonun bias elemanı uzaysal değişen ofsete sahiptir.
3. İleri beslemeli ve uzaysal değişen ofset değeri olmayan şablon.
   Geri besleme matrisinin ise sadece orta elemanı etkilidir. Kontrol matrisinin bütün elemanları etkilidir.
4. İleri beslemeli ve uzaysal ofset değeri olan şablon.
   3. kip ile aynı fakat şablonun bias elemanı uzaysal değişen ofsete sahiptir.

4. BULGULAR

Bu bölümde, 3. bölümde anlatılan Iterative Annealing optimizasyon yöntemi kullanılarak Bi-i Hücresel görü sistemi içindeki ACE16k yongası üzerinde şablon eğitimi gerçekleştiren IAŞT yazılımı hakkında detaylı bilgiler verilecek, ardından sözkonusu yazılım kullanılarak kenar ve köşe belirleme şablonlarının eğitimi gerçekleştirilecektir. Ayrıca elde edilen şablonların çeşitli örnek görüntüler üzerinde uygulanması ile bulunan sonuçlar verilecektir. Bulunan sonuçların bir diğer HSA tabanlı şablon tasarım yazılımı olan CNNOPT ve HSA tabanlı olmayan yöntemlerle (Harris, He ve Yung) karşılaştırması yapılmıştır. Ayrıca, ACE16k yongası üzerinde geliştirilen uygulama anlatılacak ve sonuçları verilecektir.

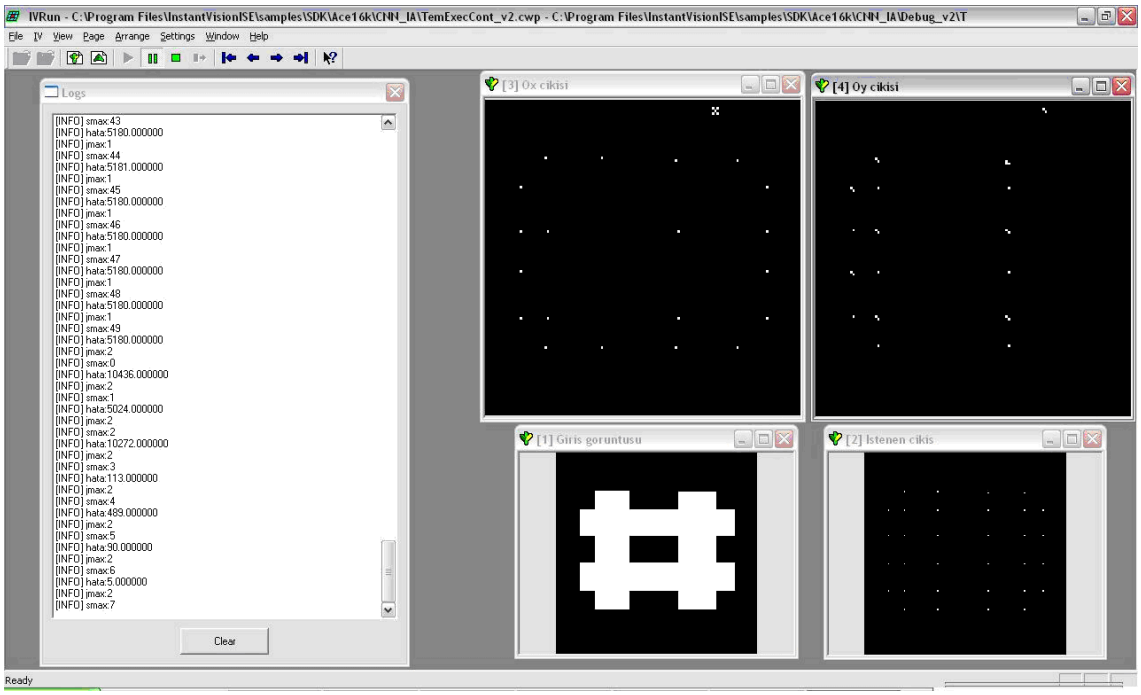
4.1. IAŞT (ITERATIVE ANNEALING İLE ŞABLON TASARIMI) YAZILIMI

Şablon eğitimi için geliştirilen yazılım, şablonların üretilmesi işlemini HSA'lar için geliştirilmiş olan Iterative Annealing optimizasyon yöntemini kullanarak gerçekleştirmektedir. Bu yöntemin ACE16k üzerinde çalışan algoritması tez çalışmasının 3. bölümünde anlatılmıştır.

Geliştirilen şablon tasarım yazılımın çalışması kenar belirleme şablonunun üretilmesi ile test edilmiştir. Hem ikili hem de gri seviyede görüntülerin kenarlarını tespit edebilen şablonlar elde edilmiştir. Daha sonra ACE16k yongası üzerinde çalışabilen bir şablonu bulunmayan köşe belirleme işlemine geçilmiştir. Köşe belirleme şablonu için değişik görüntülerle eğitim işlemleri gerçekleştirilmiştir. Elde edilen sonuçlar çeşitli görüntülere uygulanmıştır.

Yazılım süreci Bi-i SDK üzerinde Code Composer Studio yazılımı kullanılarak C++ dilinde yapılmıştır.

Aşağıda geliştirilen yazılımın arabirimi görülmektedir:



Şekil 4.1: Yazılım Arabirimi

Geliştirilen yazılıma giriş görüntüsü 128*x*128 piksel boyutlarında "input.bmp" isimli bir resim dosyası olarak verilmektedir. Bu giriş görüntüsü Şekil 4.1'de sol alttaki 1 numaralı küçük pencere içerisinde görüntülenmektedir. Elde edilmesi amaçlanan çıkış ise "d_output.bmp" isimli 128x128 piksellik bir resim dosyasıdır ve Şekil 4.1'de 2 numaralı pencerede görüntülenir. Algoritmaya bağlı olarak yongadan elde edilen sonuçlar 3 ve 4 numaralı pencerelerde görüntülenmektedir. 3 numaralı pencere *X* şablon değerlerine göre, 4 numaralı pencere ise *Y* şablon değerlerine göre elde edilen görüntüleri ekrana getirmektedir. Bu sonuç görüntüleri "output.bmp" ve "outputy.bmp" isimli dosyalara yazılır. Soldaki "logs" penceresinden de iterasyonlarla alakalı bilgiler görülebilir.

Giriş görüntüsü yazılım tarafından alındıktan sonra "bit" ya da "byte" tipindeki bir matrise yazılır. Eğer bit tipinde bir matrise kullanılırsa, görüntünün her bir pikseli "1" ya da "0" değeri ile gösterilir. Bi-i sisteminin yapısı gereği şablon değerleri "sablonadi.tem" şeklindeki bir metin dosyasından okunur. Algoritma yapısından dolayı hem *X* hem *Y* şablon değerleri oluşturulduğundan 2 adet ".tem" uzantılı dosya

kullanılmaktadır. Bu dosyaya ek olarak yazılıma kullanıcı tarafından ilk değerlerin kolayca verilebilmesi için bir metin dosyası daha ilave edilmiştir. “baslangic.txt” isimli bu dosya sistem tarafından bir kez okunur ve içinden şablonun ilk değerleri (A, B, z ve mod bilgisi) alınır ve yazılım içerisinde y_i değerlerinin elde edilmesinde kullanılır. Bunun dışında başlangıç sıcaklığı minimum sıcaklık, adım boyutu adı verilen soğuma parametresi ile iterasyon sayılarını belirleyen parametreler yazılım kodu içerisinde tanımlanmışlardır.

Yazılım, başlangıç değerleri ile giriş ve istenen çıkış görüntülerini aldıktan sonra yeni şablon değerlerini üretir. 2 adet şablon değer seti ile giriş ve istenen çıkış görüntüsü ACE16k yongasına gönderilir ve iki adet çıkış görüntüsü elde edilir. Çıkış görüntüleri ve istenen çıkış görüntüsü ile 2 adet hata değeri elde edilir. Bu sonuçların karşılaştırılması sonucunda yeni şablon değer seti belirlenir, sıcaklık düşürülür. Algoritma sıcaklık minimuma ulaşana kadar devam eder. Minimum sıcaklığa ulaşılınca başlangıç sıcaklığı ile yeniden algoritma çalıştırılır. Bulunan sonuçlar “cozumler.txt” isimli metin dosyasında saklanır.

4.2. KENAR BELİRLEME (EDGE DETECTION)

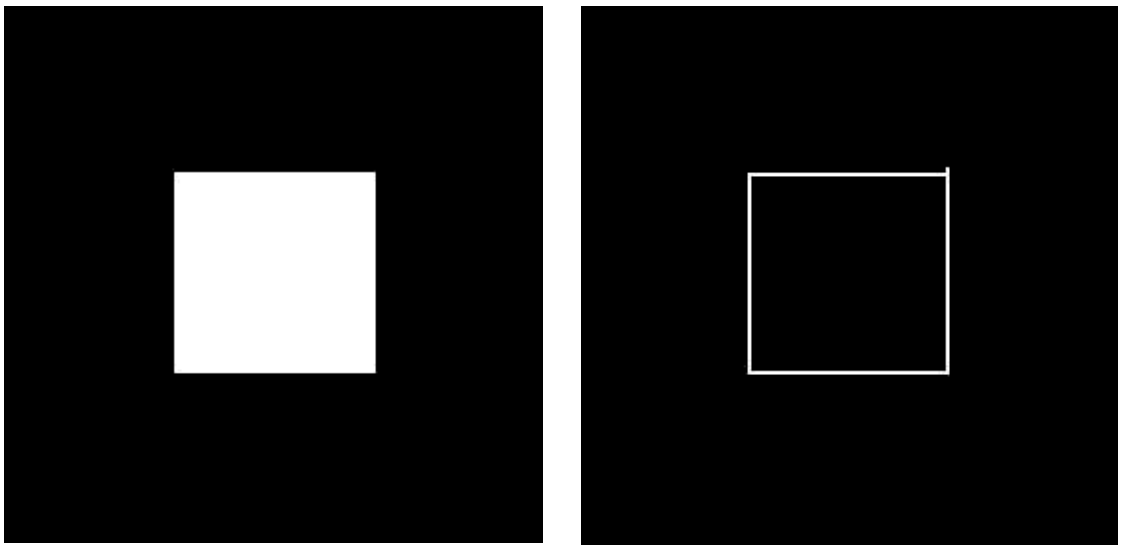
Kenar belirleme işlemi için başlangıç şablon değerleri olarak şu değerler kullanılmıştır:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad I = 2 \tag{4.1}$$

İlk olarak aşağıda gösterilen kare şekil kullanılarak eğitim yapılmıştır.

Bu işlem sonucunda elde edilen şablon değerleri ve elde edilen sonuç görüntüsü şöyledir:

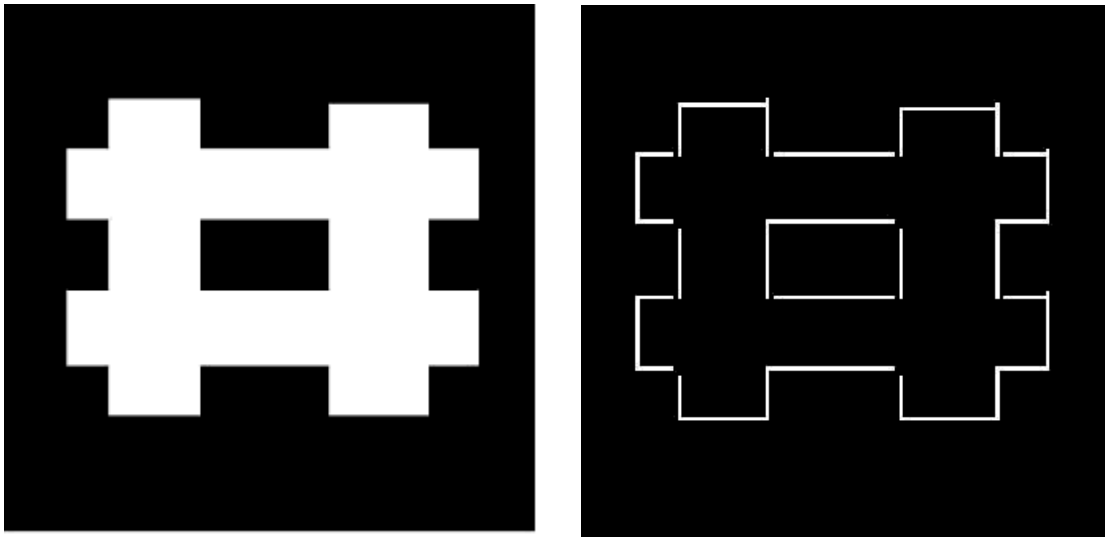
$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2.66 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} -1.81 & 1.96 & -1.6 \\ -0.02 & 3.49 & 0.20 \\ -0.2 & 1.59 & -3 \end{bmatrix} \quad I = 1.78 \tag{4.2}$$



Giriş görüntüsü Eğitim sonucu

Şekil 4.2: Kare şeklinin kenar belirleme eğitimi

(4.2) değerlerinin daha karmaşık bir şekle uygulanmasıyla elde edilen sonuç aşağıdadır.

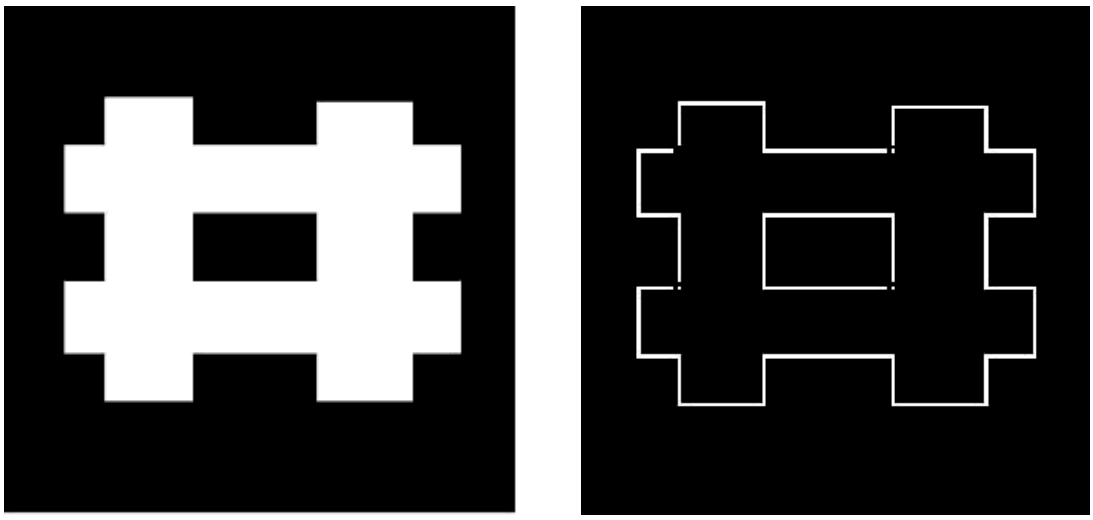


Giriş görüntüsü Çıkış görüntüsü

Şekil 4.3: Kenar belirleme örneği - 1

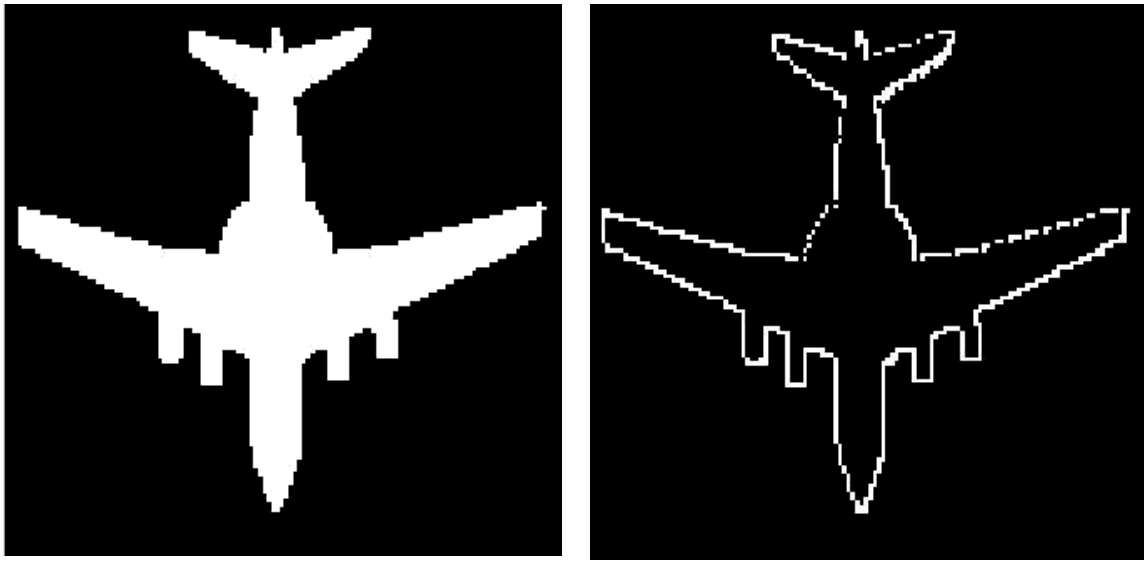
Şekil 4.3' te görüldüğü gibi, karmaşık şekil üstünde elde edilen sonuç yeterince başarılı olmamıştır. Bu nedenle üstteki giriş görüntüsü ve ilk denemeden elde edilen şablon değerleri ile yeni bir deneme yapılmıştır. Elde edilen şablon değerleri ve bu değerlerin değişik giriş görüntülerine uygulanmasıyla elde edilen çıkış görüntüleri aşağıda verilmiştir:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3.38 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} -0.56 & 1.78 & -2.51 \\ 1.09 & 4.5 & 0.54 \\ -1.76 & 0.95 & -1.98 \end{bmatrix} \quad I = 0.79 \tag{4.3}$$



Giriş görüntüsü Çıkış görüntüsü

Şekil 4.4: Kenar belirleme örneği - 2



Giriş görüntüsü Çıkış görüntüsü

Şekil 4.5: Kenar Belirleme örneği - 3

Yeni şablon değer seti ile elde edilen sonuçların ilk setten daha başarılı olduğu görülmektedir.

4.2.1. Gri Seviyede Kenar Belirleme (Grayscale Edge Detection)

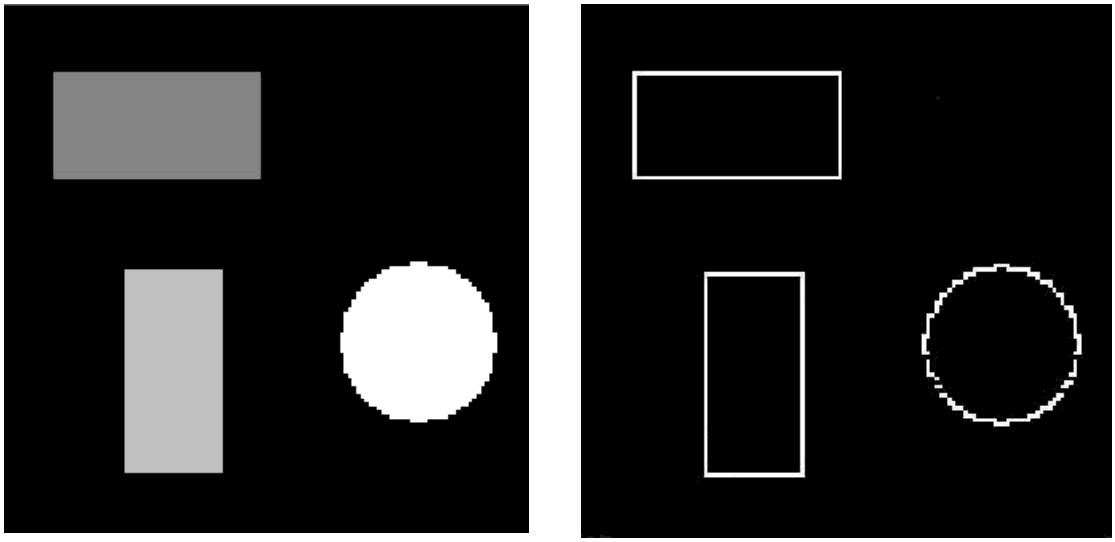
Kenar belirleme işlemi için elde edilen en son şablon değerleri başlangıç şablon değerleri olarak kullanılarak eğitim işlemine başlanmıştır:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3.38 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} -0.56 & 1.78 & -2.51 \\ 1.09 & 4.5 & 0.54 \\ -1.76 & 0.95 & -1.98 \end{bmatrix} \quad I = 0.79 \tag{4.4}$$

Elde edilen şablon değerleri şöyledir:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -0.52 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} -3 & 1.06 & -3 \\ 1.55 & 5.3 & 1.25 \\ -1.12 & -1.08 & -1.36 \end{bmatrix} \quad I = 1.53 \tag{4.5}$$

Bu şablon seti kullanılarak elde edilen değişik giriş ve çıkış görüntüleri aşağıda verilmiştir.



Giriş görüntüsü Çıkış görüntüsü

Şekil 4.6: Gri seviye kenar belirleme örneği -1

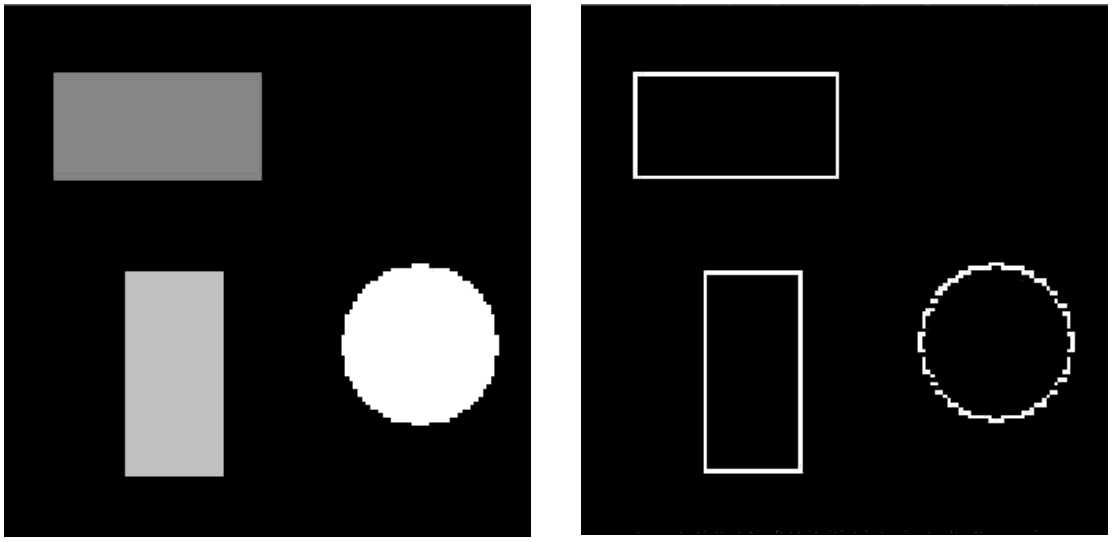
Giriş görüntüsü Çıkış görüntüsü

Şekil 4.7: Gri seviye kenar belirleme örneği -2

Yukarıdaki şablon setinin eğitimine devam edilerek elde edilen yeni set aşağıdaki şekildedir:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3.2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} -3 & 1.94 & -1.55 \\ 0.64 & 5.88 & -0.51 \\ -1.79 & -0.02 & -1.77 \end{bmatrix} \quad I = -0.8 \qquad (4.6)$$

Bu şablon seti kullanılarak elde edilen değişik giriş ve çıkış görüntüleri ise aşağıda verilmiştir.



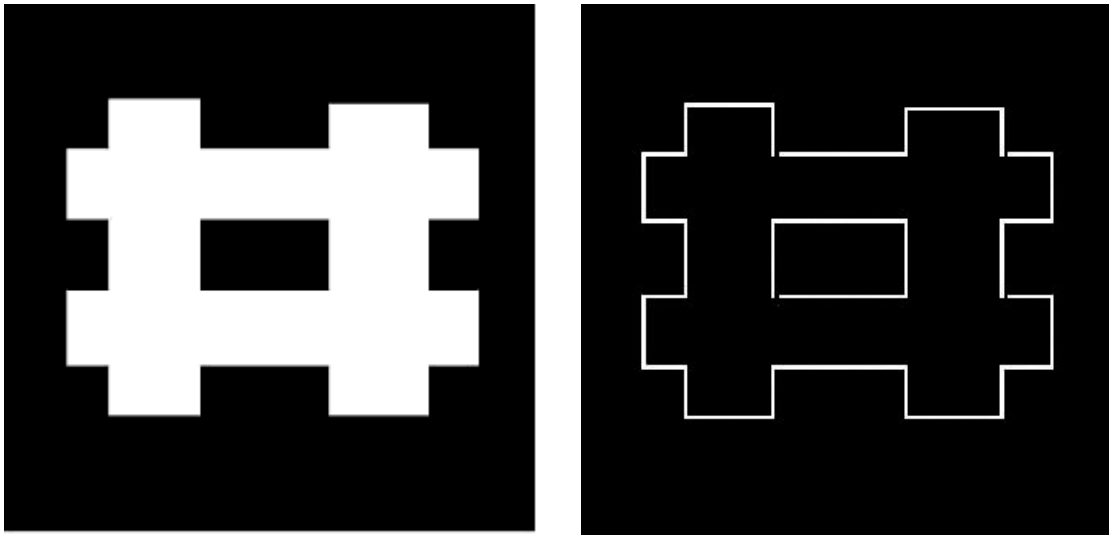
Giriş görüntüsü Çıkış görüntüsü

Şekil 4.8: Gri seviye kenar belirleme örneği - 3

Giriş görüntüsü Çıkış görüntüsü

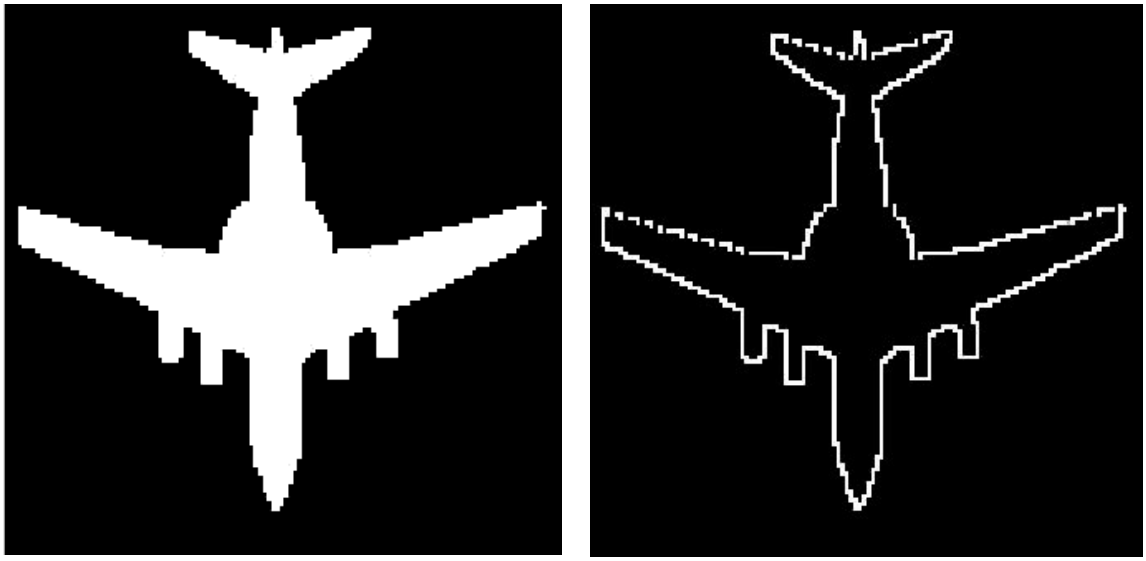
Şekil 4.9: Gri seviye kenar belirleme örneği - 4

Gri seviyede kenar belirleyen bu şablon seti aynı zamanda siyah/beyaz görüntülerin de kenarlarını belirleyebilmektedir:



Giriş görüntüsü Çıkış görüntüsü

Şekil 4.10: Gri seviye kenar belirleme örneği - 5



Giriş görüntüsü Çıkış görüntüsü

Şekil 4.11: Gri seviye kenar belirleme örneği - 6

Yukarıda verilmiş olan giriş ve çıkış görüntü çiftleri, (4.6) kenar belirleme şablon setinin hem gri seviyeli hem de siyah/beyaz görüntüler üzerinde işlem yapabildiğini, ayrıca en başarılı sonuçları verdiğini göstermektedir.

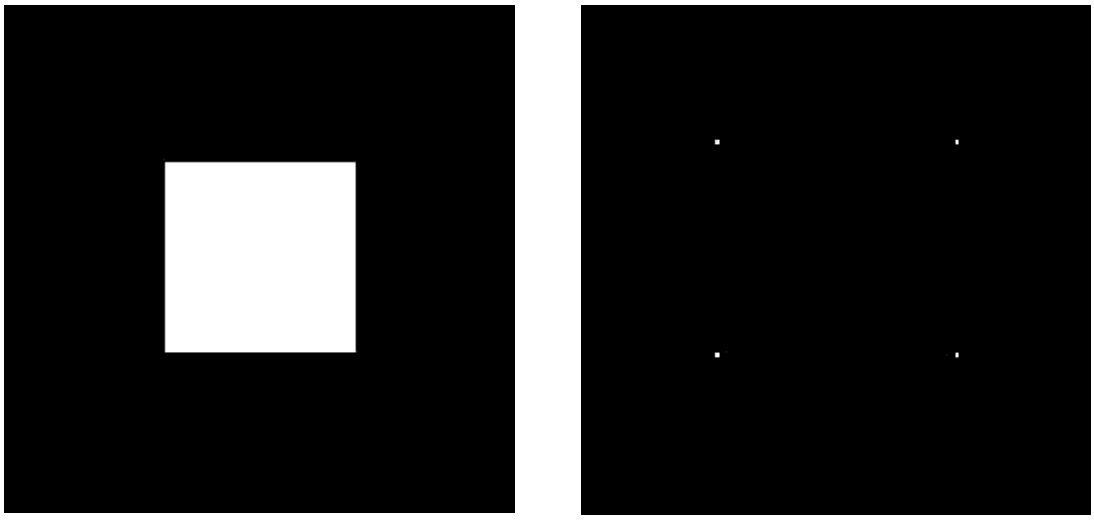
4.3. KÖŞE BELİRLEME (CORNER DETECTION)

Köşe belirleme işlemi için başlangıç şablon değerleri olarak şu değerler kullanılmıştır:

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 3 & 0\\ 0 & 0 & 0\end{bmatrix} \quad B=\begin{bmatrix}-1.3 & -1.3 & -1.3\\ -1.3 & 0 & -1.3\\ -1.3 & -1.3 & -1.3\end{bmatrix} \quad I=0 \tag{4.7}$$

İlk olarak kenar belirleme de olduğu gibi kare şekil kullanılarak eğitim yapılmıştır. Elde edilen şablon değerleri sonuç görüntüsü aşağıdadır:

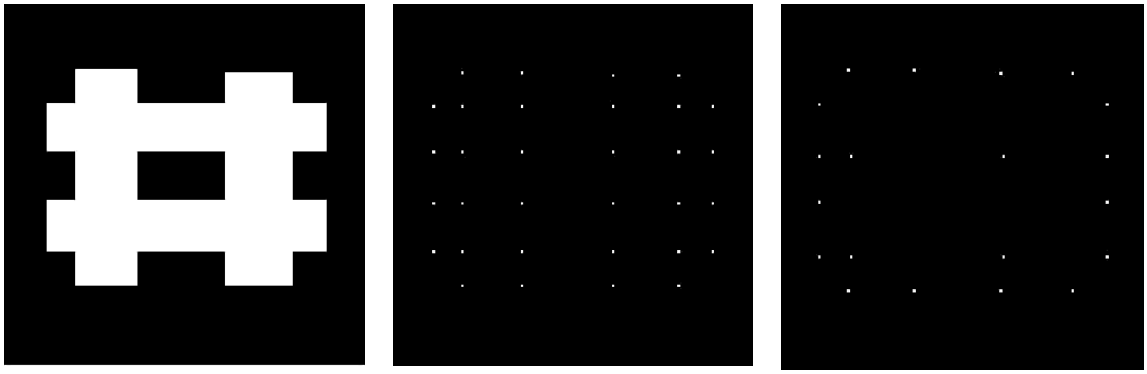
$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 5.84 & 0\\ 0 & 0 & 0\end{bmatrix} \quad B=\begin{bmatrix}2.11 & -2.4 & 1.72\\ -3 & 5.12 & -3\\ 1.88 & -3 & 2.39\end{bmatrix} \quad I=6 \tag{4.8}$$



Giriş görüntüsü Eğitim sonucu

Şekil 4.12: Kare şekli için köşe belirleme eğitimi

İyi görünen bu değerler ile karmaşık bir şeklin kenarları bulunmaya çalışıldığında aşağıdaki sonuçlara ulaşılmıştır.



Giriş görüntüsü | İstenen çıkış görüntüsü | Yonga çıkış görüntüsü

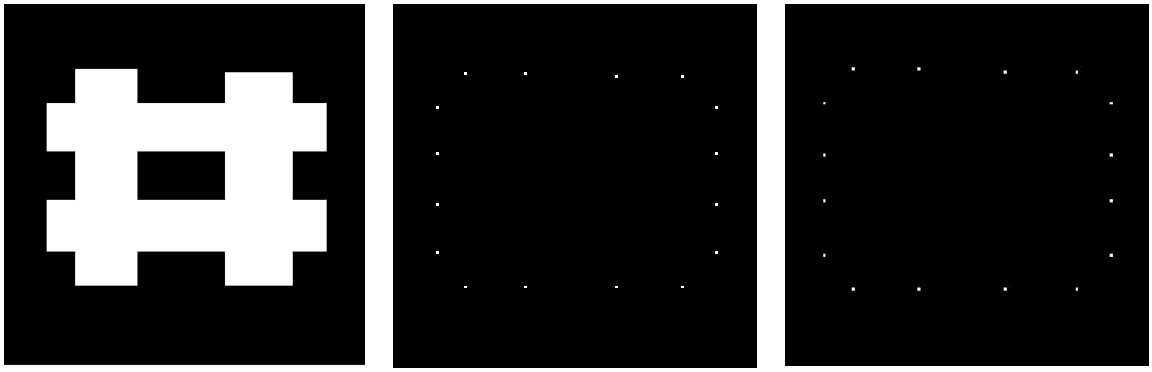
Şekil 4.13: Köşe belirleme örneği -1

Elde edilen sonuç tam doğru değildir.

Köşe belirleme işlemi yapılırken dikkat edilmesi gereken konulardan bir tanesi köşe tipidir. Köşedeki pikseli çevreleyen komşu piksellerin renk değerlerine göre içbükey (en az 5 beyaz komşu piksel) ve dışbükey (en az 5 siyah komşu piksel) olmak üzere iki tip köşe mevcuttur. Elde edilen şablon değer seti iki tip köşeyi bulma işlemini gerçekleştiremediğinden, bu iki tip köşeyi ayrı ayrı bulan şablon değer setleri için eğitimler yapılmıştır. Eğitimler sonucunda elde edilen şablon değerleri ve eğitim sonuç görüntüleri aşağıdaki şekildedir.

Dışbükey köşe bulma şablonu:

$$A = \begin{bmatrix} 0. & 0 & 0 \\ 0 & 5.07 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1.56 & -1.84 & 1.29 \\ -2.84 & 4.05 & -3 \\ 1.26 & -3 & 2.57 \end{bmatrix} \quad I = 6 \tag{4.9}$$

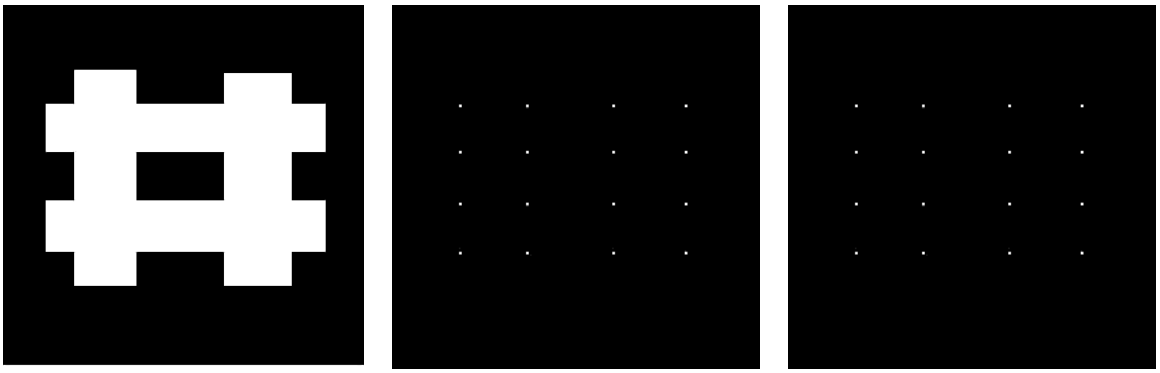


Giriş görüntüsü İstenen çıkış görüntüsü Eğitim sonucu

Şekil 4.14: Dışbükey köşe belirleme eğitimi

İçbükey köşe bulma şablonu:

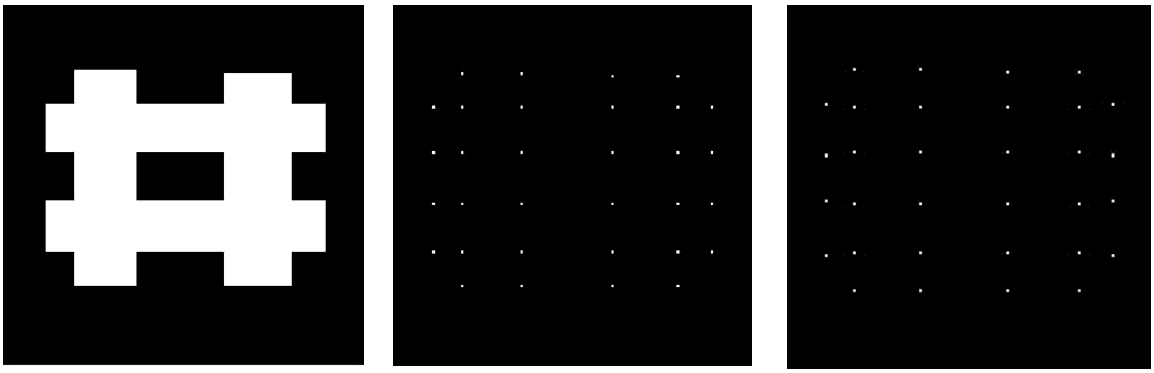
$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0.97 & -2.3 & 1.37 \\ -1.12 & 5.44 & -0.9 \\ 2.27 & -3 & 1.09 \end{bmatrix} \quad I = 4.49 \tag{4.10}$$



Giriş görüntüsü İstenen çıkış görüntüsü Eğitim sonucu

Şekil 4.15: İçbükey köşe belirleme eğitimi

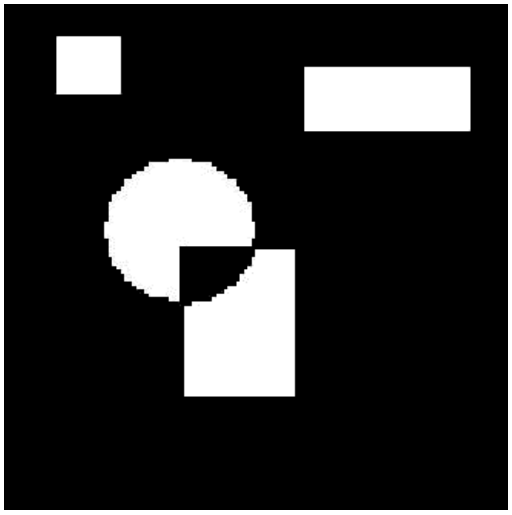
Eğitim sonuçları başarılı olmuş, içbükey ve dışbükey köşeler bulunmuştur. Bu iki şablon OR işlemine tabi tutularak yukarıdaki giriş görüntüsüne ait istenen çıkış bulunabilir.



Giriş görüntüsü İstenen çıkış görüntüsü Yonga çıkış görüntüsü

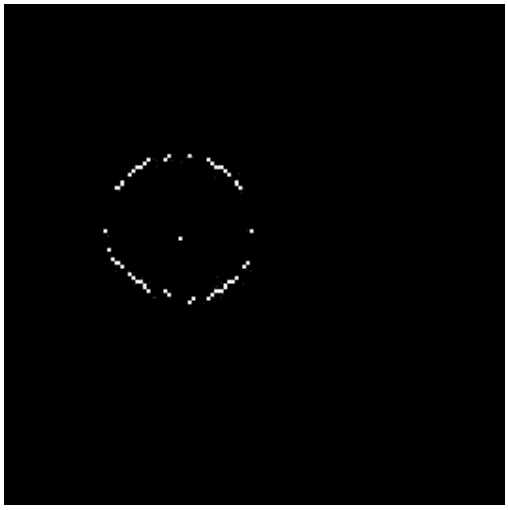
Şekil 4.16: OR işlemi sonucunda elde edilen köşe belirleme

Bu şablon değer setleri kullanılarak bazı şekillerin köşelerinin bulunması ile ilgili örnekler aşağıdadır.



Giriş görüntüsü

Dışbükey köşe belirleme şablon sonucu



İçbükey köşe belirleme şablon sonucu

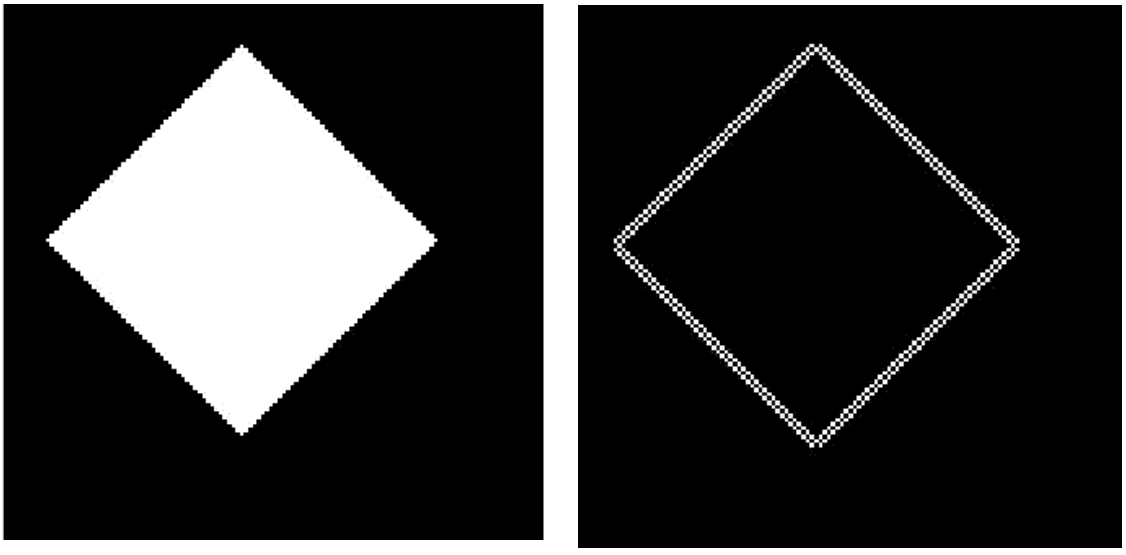
OR işlemi sonucu

Şekil 4.17: Köşe belirleme örneği -2

Giriş görüntüsü Dışbükey köşe belirleme şablon sonucu

Şekil 4.18: Köşe belirleme örneği - 3

En son elde edilmiş olan dışbükey köşe şablonu kullanarak aşağıdaki gibi bir deneme yapıldığında elde edilen sonuç aşağıda verilmiştir:



Giriş görüntüsü Dışbükey köşe belirleme şablon sonucu

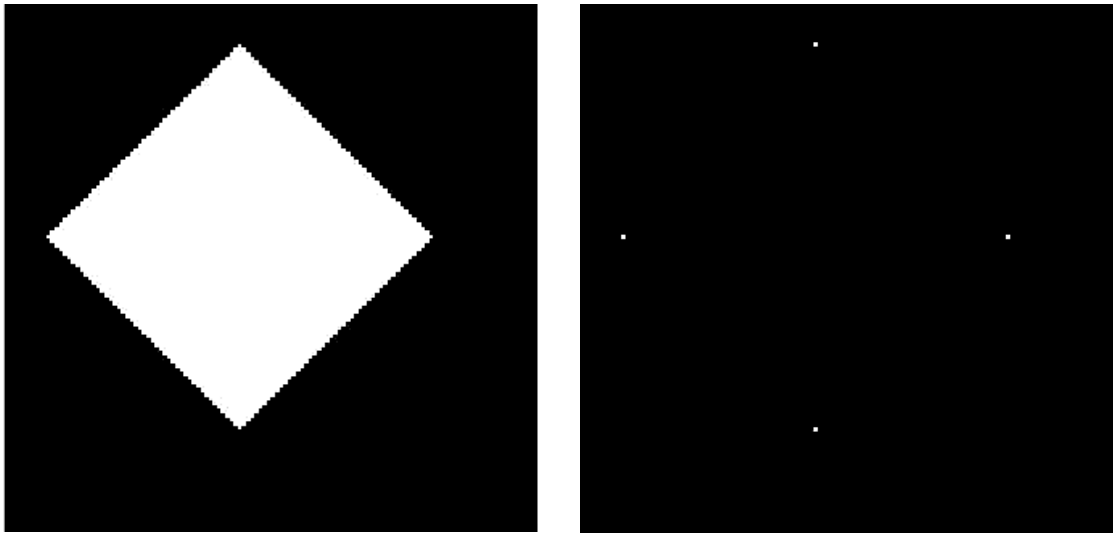
Şekil 4.19: Köşe belirleme örneği - 4

Şekil 4.19'de görüldüğü gibi bulunmuş olan dışbükey köşe şablonu döndürülmüş kenarları olan şekiller üzerinde etkisiz kalmaktadır. Bu nedenle eğitim işlemi yukarıdaki eşkenar dörtgen kullanılarak tekrarlanmıştır. Elde edilen yeni şablon değerleri şöyledir:

Dışbükey köşe bulma şablonu:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4.34 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} -1.48 & -0.1 & -1.08 \\ -0.64 & 6 & -1.36 \\ -1.79 & -0.72 & -1.37 \end{bmatrix} \quad I = 5.84 \tag{4.11}$$

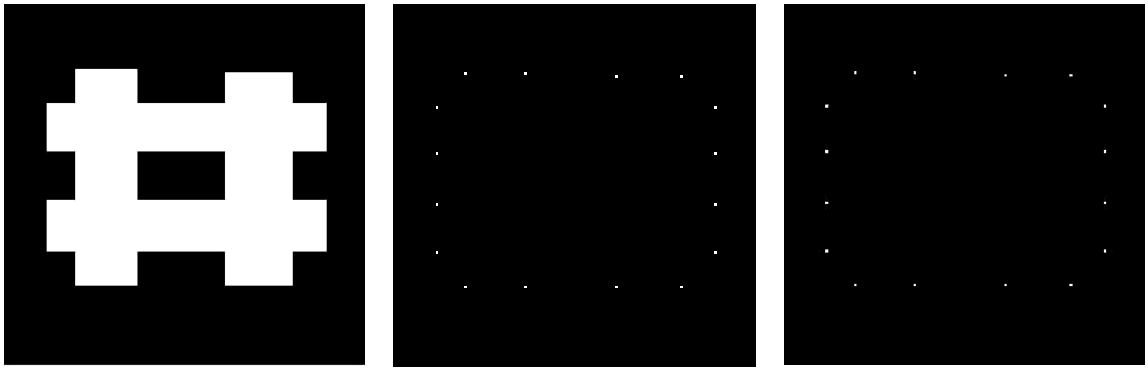
Eğitim sonucu aşağıdaki gibidir:



Giriş görüntüsü Dışbükey köşe belirleme şablon sonucu

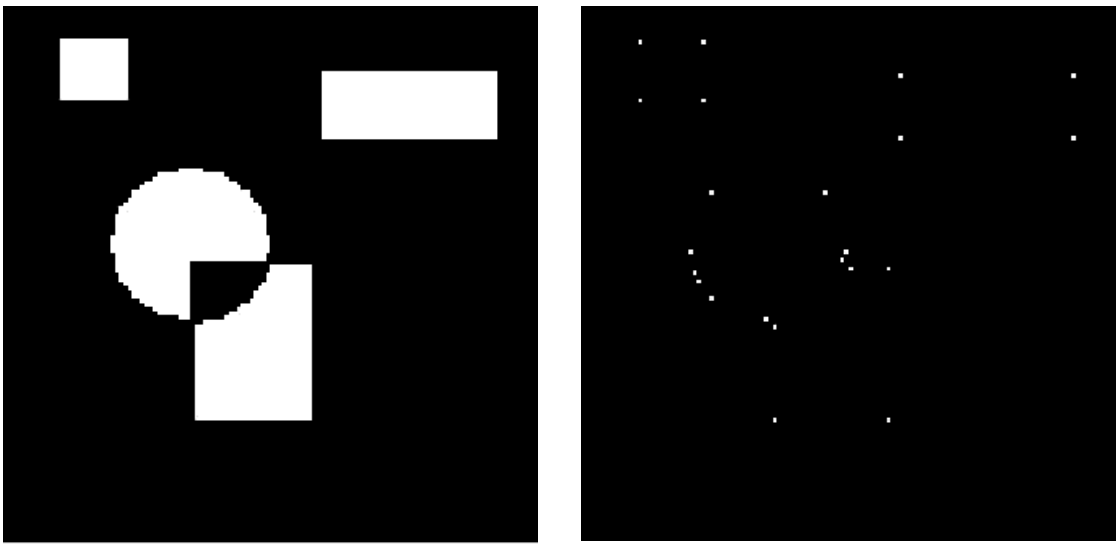
Şekil 4.20: Dışbükey köşe belirleme eğitimi

Bu şablon değer setleri kullanılarak bazı şekillerin köşelerinin bulunması ile ilgili örnekler aşağıdadır.



Giriş görüntüsü İstenen çıkış Eğitim sonucu

Şekil 4.21: Köşe belirleme örneği - 5



Giriş görüntüsü Dışbükey köşe belirleme şablon sonucu

Şekil 4.22: Köşe belirleme örneği - 6

Giriş görüntüsü Dışbükey köşe belirleme şablon sonucu

Şekil 4.23: Köşe belirleme örneği - 7

Şablonların uygulanması ile elde edilen çıkı görüntülerinden görülebileceği gibi, bulunan köşe belirleme şablonları oldukça başarılı sonuçlar üretmiştir. Hem içbükey hem de dışbükey şablonlar eğitilmiştir. Bu şablon değerlerinin sonuçlarının karşılaştırılması bir sonraki bölümde yapılacaktır.

4.4. KARŞILAŞTIRMA

Bu kısımda Iterative Annealing optimizasyon yöntemine dayanan IAŞT yazılımının performansı değerlendirilecektir. Karşılaştırma, köşe belirleme işlemi üzerinde yapılacaktır ve iki tür uygulamadan yararlanılacaktır. Bunlardan bir tanesi ACE16k yongası üzerinde çalışan başka bir bir şablon tasarım yazılımı olan CNNOPT [6] uygulamasıdır. Karşılaştırmada kullanılacak diğer yöntemler ise literatürde bulunan matematik hesaplamalar ile köşe belirleyen yöntemlerdir. İki adet yöntem kullanılacaktır. Biri Harris tarafından [48], diğeri He ve Yung tarafından geliştirilen yöntemdir [50]. Bu iki yöntemin programlanması MATLAB üzerinde yapılmıştır.

Geliştirilen IAŞT yazılımı ile diğer uygulamaların işlem zamanları ve elde edilen sonuçların doğruluğu karşılaştırılacaktır. Bu değerlendirmelerin yapıldığı ortamı, Bi-i sisteminin bağlı olduğu ve uygulamaların çalıştırıldığı Intel Core2Duo 2.0GHz işlemcili 1.5GB bellekli bir dizüstü bilgisayar oluşturmuştur.

4.4.1. CNNOPT ile Karşılaştırma

İlk olarak geliştirilen yazılım ile CNNOPT uygulamasının karşılaştırılması yapılmıştır. CNNOPT uygulaması da yinelemeli çalışan bir uygulamadır. ACE16k yongası üzerinde çalışmaktadır. Giriş, istenen çıkış görüntülerini alıp sonuca ulaşmaya çalışmaktadır. AMC programlama dili ve MATLAB kullanılarak gerçekleştirilmiştir. Bir çeşit SA yöntemi olan Coupled Simulated Annealing (CSA) [20] optimizasyon yöntemine dayanmaktadır.

İki uygulama çalışma zamanları ve elde edilen köşe belirleme şablonlarının sonuçları ile değerlendirilmiştir. İşlem zamanları, her iki yöntemin 250 adımlık kısmının zamanlarının 10 kez ölçülüp ortalamalarının alınması ile belirlenmiştir. Elde edilen zaman sonuçları aşağıdaki tabloda verilmiştir.

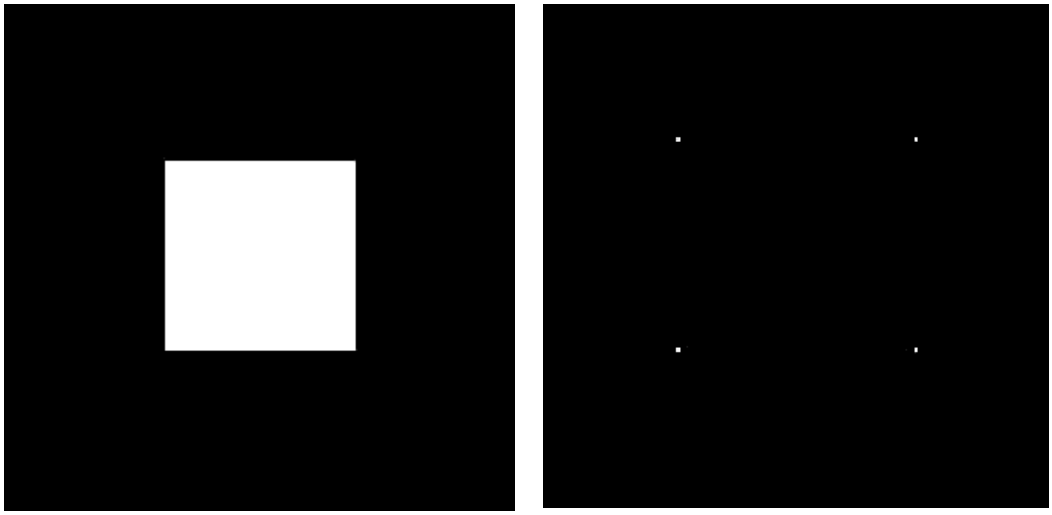
Tablo 4.1 : CNNOPT ile IAŞT yazılımının işlem zamanlarının karşılaştırılması

Yöntem	**250 adım**	**Her bir adım**	**5000 adım**
IAŞT	131 sn.	0,524 sn.	2620 sn. = 43,6 dk.
CNNOPT	129 sn.	0,516 sn.	2580 sn. = 43 dk.

CNNOPT uygulamasının kare ve eşkenar dörtgen giriş görüntülerine uygulanması ile edilen şablon sonuçları ve bu şablonların çeşitli örnek görüntüler üzerindeki sonuçları aşağıda verilmiştir.

Eğitim işlemine kare şekil ile başlanmıştır. Bulunan şablon seti şöyledir:

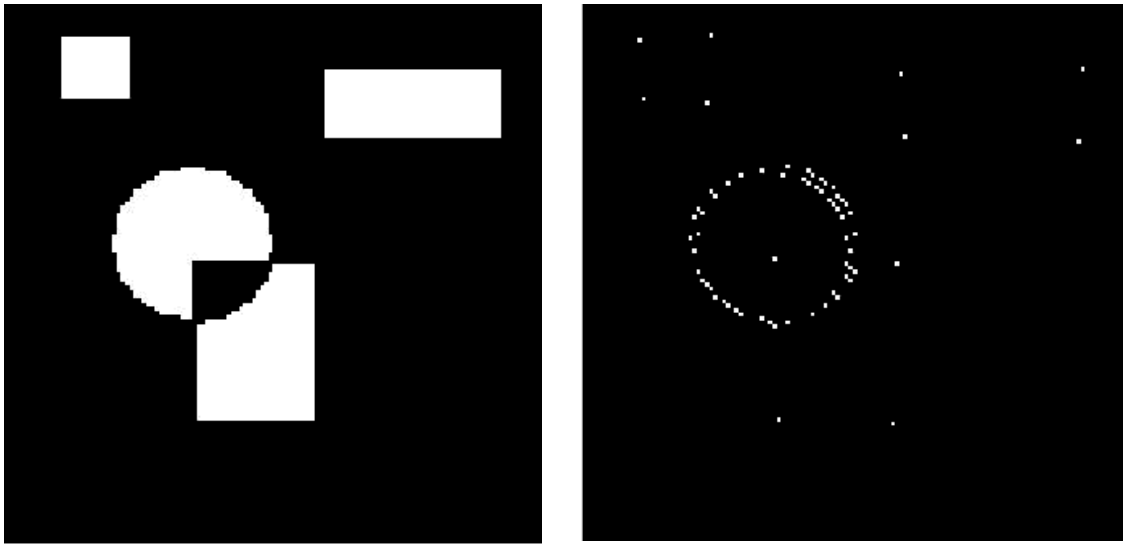
$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 5.67 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0.03 & 3 & -2.92 \\ -2.92 & 3.12 & 1.56 \\ 2.62 & -2.92 & 1 \end{bmatrix} \quad I = 6 \tag{4.12}$$



Giriş görüntüsü Eğitim sonucu

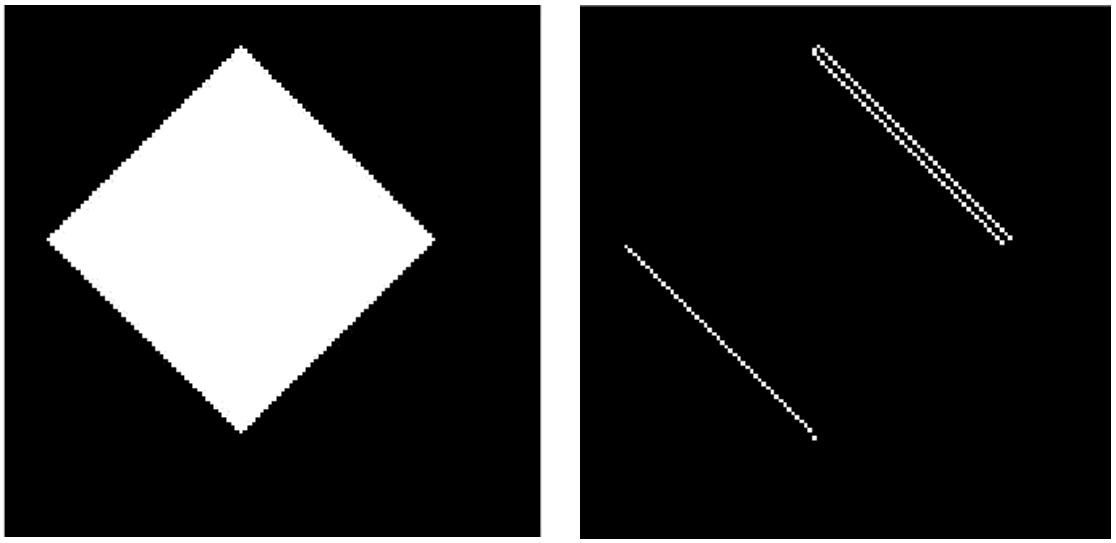
Şekil 4.24: CNNOPT ile kare köşe belirleme eğitimi

Bu şablon seti kullanılarak elde edilen diğer bazı sonuşlar aşağıda verilmiştir.



Giriş görüntüsü CNNOPT sonucu

Şekil 4.25: CNNOPT ile köşe belirleme örneği - 1



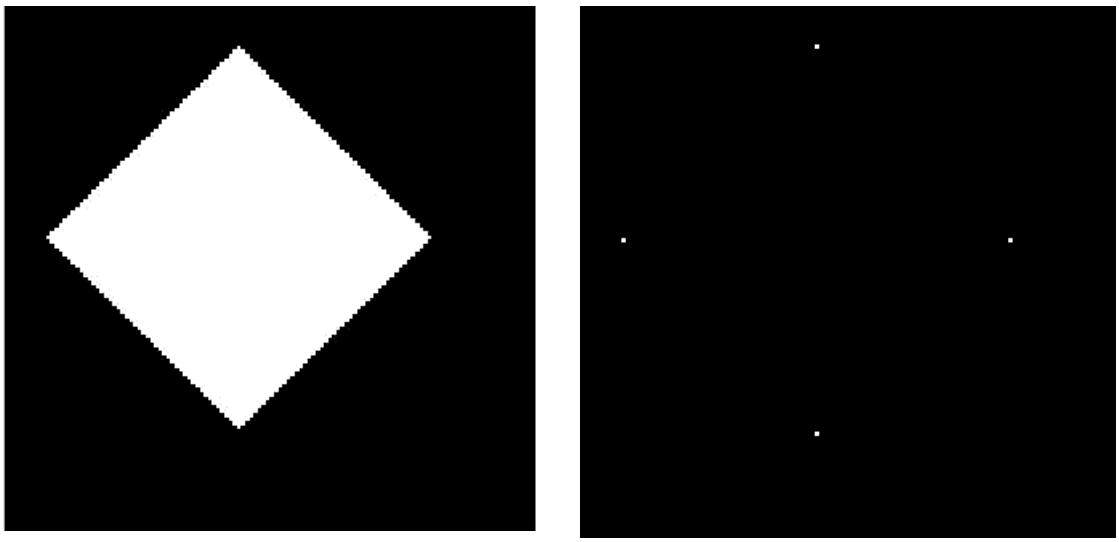
Giriş görüntüsü CNNOPT sonucu

Şekil 4.26: CNNOPT ile köşe belirleme örneği - 2

IAŞT yazılımında olduğu gibi, eşkenar dörtgen şekline uygulana şablon köşeleri verememiştir. Bu şekil kullanılarak eğitim işlemi devam edilmiştir. Elde edilen şablon seti aşağıdadır.

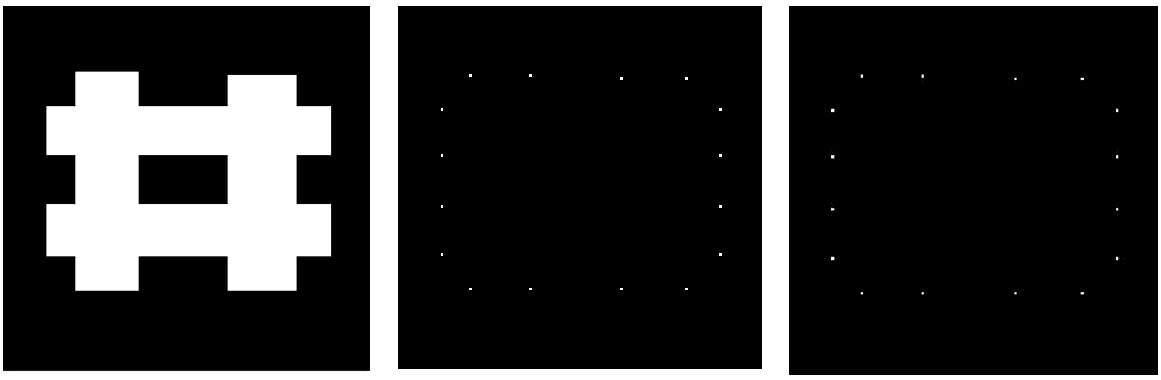
$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4.65 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} -1.87 & 0.53 & -2.33 \\ -0.8 & 5.95 & -0.74 \\ -1.2 & -1.35 & -0.82 \end{bmatrix} \quad I = 5.07 \tag{4.13}$$

Bu setin çeşitli örnek görüntülere uygulanmasıyla elde edilen sonuçlar aşağıda verilmiştir:



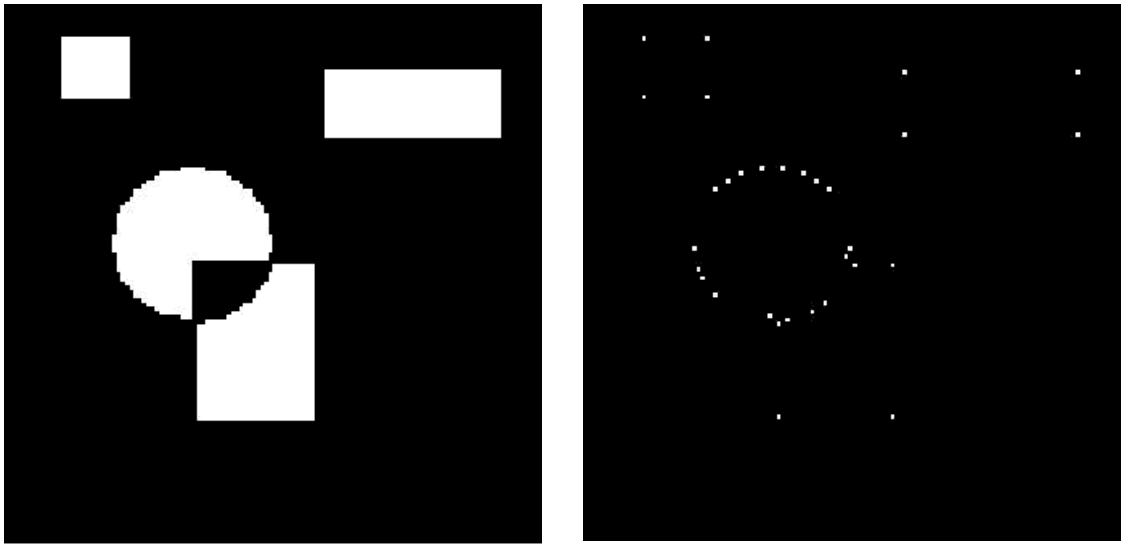
Giriş görüntüsü Dışbükey köşe belirleme şablon sonucu

Şekil 4.27: CNNOPT ile köşe belirleme örneği - 3



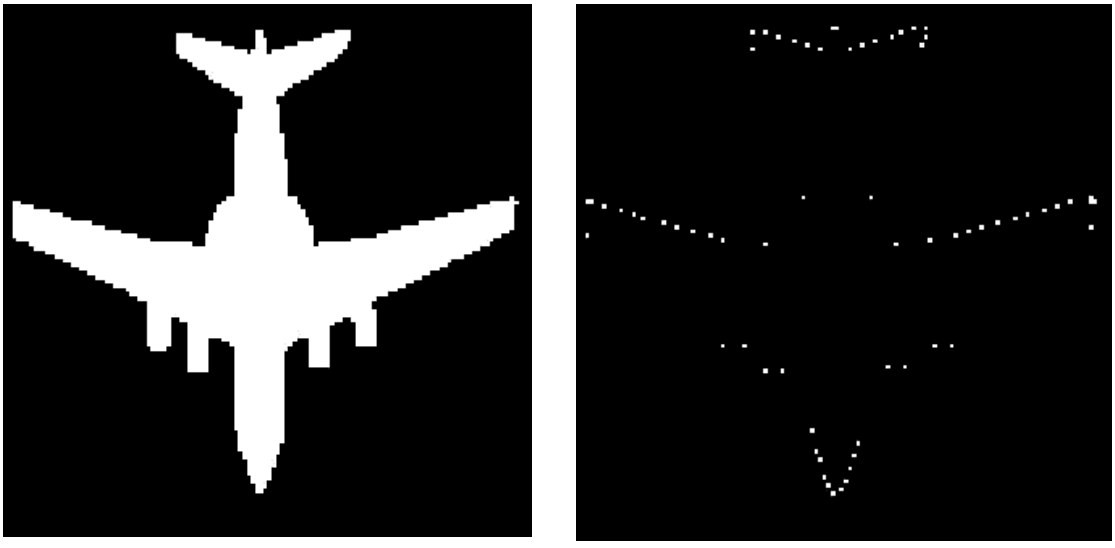
Giriş görüntüsü İstenen çıkış Eğitim sonucu

Şekil 4.28: CNNOPT ile köşe belirleme örneği - 4



Giriş görüntüsü Dışbükey köşe belirleme şablon sonucu

Şekil 4.29: CNNOPT ile köşe belirleme örneği - 5



Giriş görüntüsü Dışbükey köşe belirleme şablon sonucu

Şekil 4.30: CNNOPT ile köşe belirleme örneği - 6

Tablo 4.1'de verilen işlem zamanlarına bakıldığında, IAŞT yazılımı ile CNNOPT sisteminin neredeyse aynı adım zamanlarına sahip olduğu görülmektedir. IAŞT yazılımı algoritması gereği her bir adımda 2 adet şablon değer seti üretmektedir. Bu setler ACE16k yongası ile giriş görüntüsüne uygulanmaktadır. Bu değer setlerinin ACE16k değer setine gönderilmesi işlemi C++ programlama diliyle programlama yapılmasından dolayı sabit disk üstünde bulunan "dosyaadi.tem" şeklindeki metin dosyalarına algoritmanın her bir adımında iki kez yazma işlemi gerçekleştirilmesi ile mümkün olmaktadır. Bu yazma işlemi algoritmanın en uzun süreli işlemleridir. CNNOPT sisteminde ise AMC programlama dili ve MATLAB kullanılarak programlama yapılmıştır. AMC programlama dilinde şablon değerlerinin ACE16k yongasına gönderilmesi için sabit diske yazım zorunluluğu yoktur. Bu nedenle elde edilen sürelere bakıldığında, IAŞT yazılımının oldukça hızlı çalışabildiği görülmektedir.

CNNOPT ile yapılan köşe belirleme eğitimleri sonuçları ve bu sonuçların çeşitli örnek görüntülere uygulanması ile elde edilen sonuçlar her iki sisteminde bulduğu şablon değerlerinin benzer şekilde köşeleri belirlediğini göstermektedir.

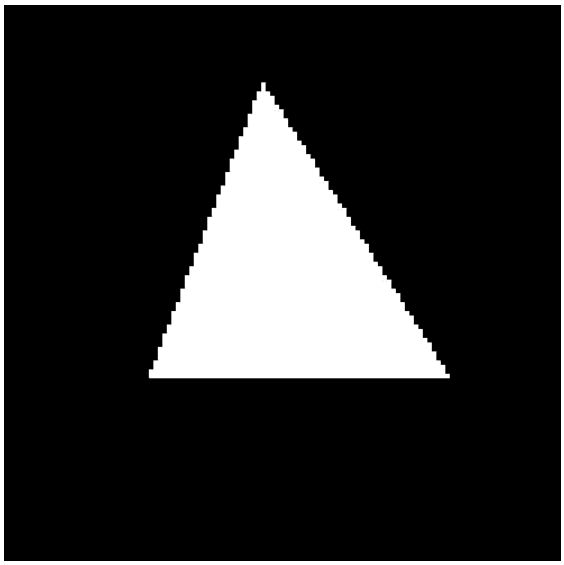
4.4.2. HSA tabanlı olmayan yöntemlerle karşılaştırma

Bu kısımda, HSA tabanlı olmayan yöntemler (Harris, He ve Yung) kullanılarak karşılaştırma yapılmıştır. Bu yöntemlerden Harris yöntemi ile bilgiler Bölüm 3.1.4'te verilmiştir. He ve Yung yöntemi ise eğim derece uzayı (CSS: Curvature Scale-Space) [60] isimli bir tekniğe dayanmaktadır ve Canny kenar belirleyicisi [55] ile kenarların bulunması gerekmektedir. Bulunan kenarlar üzerinde her bir köşe için eğimler hesaplanır ve bunların yerel maksimumlarının köşe olabileceği düşünülür.

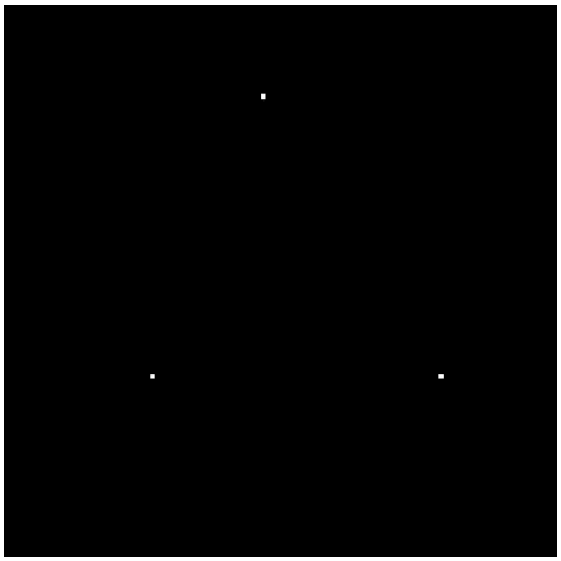
IAŞT yazılımı ile diğer yöntemlerin örnek görüntüler üzerine uygulanması ile bulunan sonuçlar karşılaştırılmıştır. Ayrıca Bi-i sistemi üzerinde şablonların çalıştırılmasıyla köşe belirleme süreleri ile MATLAB üzerinde çalışan yöntemlerin köşeleri belirleme zamanları ölçülüp karşılaştırılmıştır. Zaman ölçümünde her bir uygulama 10 kez çalıştırılmış ve elde edilen zamanların ortalaması alınmıştır. Bulunan zaman sonuçları ve çeşitli örnek görüntüler üzerinde elde edilen sonuçlar aşağıda verilmiştir:

Tablo 4.2: HSA tabanlı olmayan yöntemlerle IAŞT yazılımının işlem zamanlarının karşılaştırılması

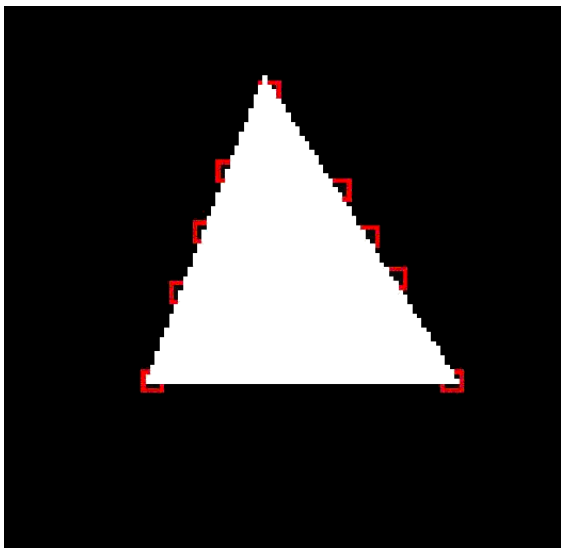
Yöntem	**Üçgen**	**#**	**Uçak**
IAŞT	0,000545 sn.	0,000545 sn.	0,000545 sn.
Harris	0.03460 sn.	0,05264 sn.	0,0416 sn.
He ve Yung	0.08258 sn.	0,09614 sn.	0,10469 sn.



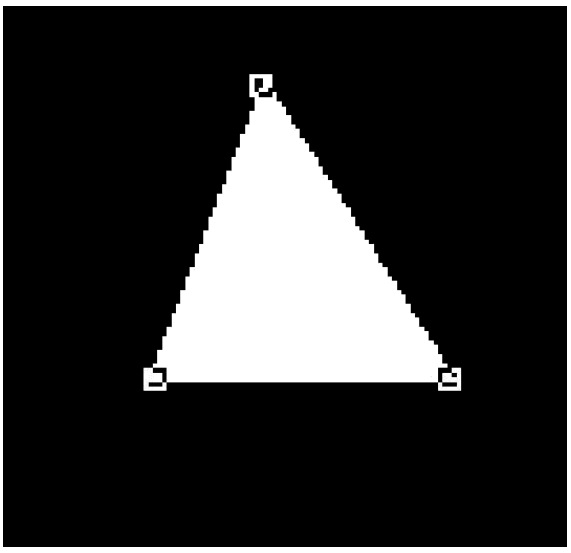
Giriş görüntüsü



Köşe belirleme şablon sonucu

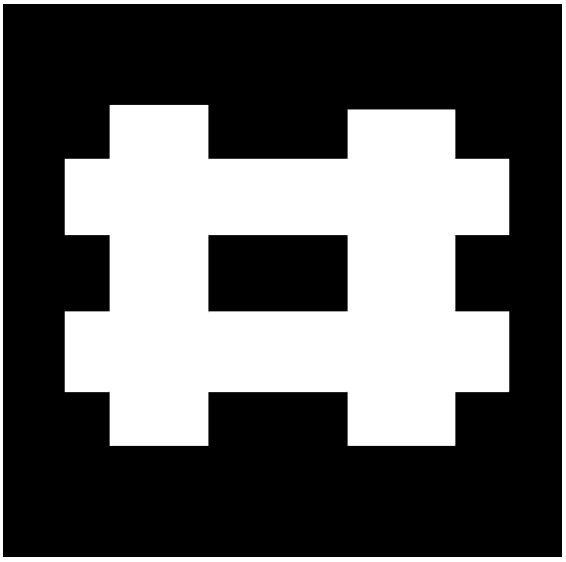


Harris Yöntemi

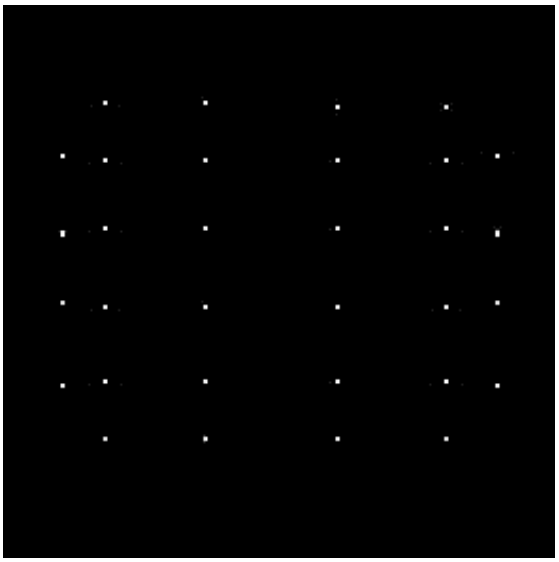


He ve Yung Yöntemi

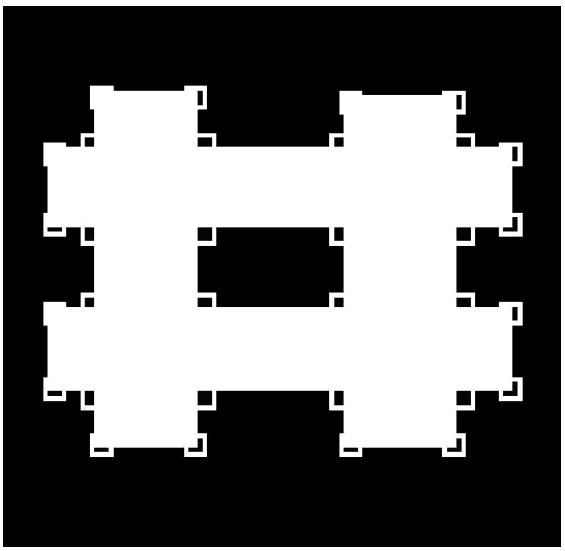
Şekil 4.31: Karşılaştırma 1



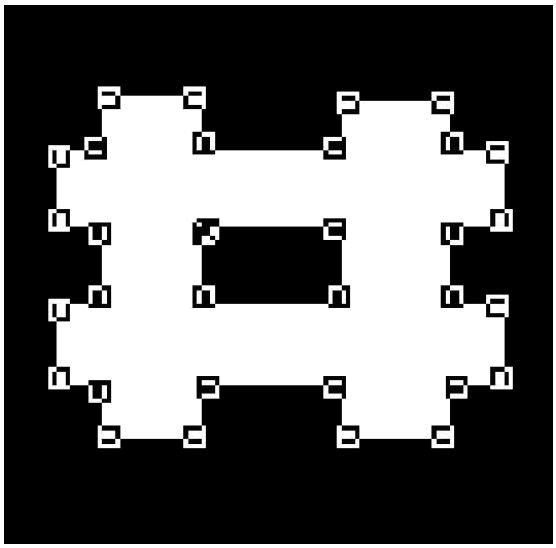
Giriş görüntüsü



Köşe belirleme şablon sonucu



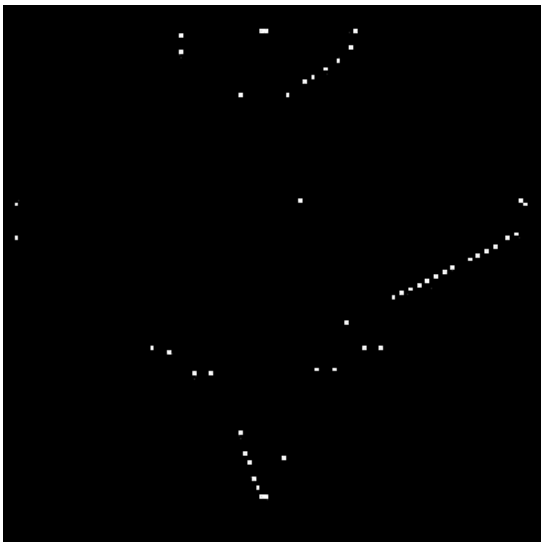
Harris Yöntemi



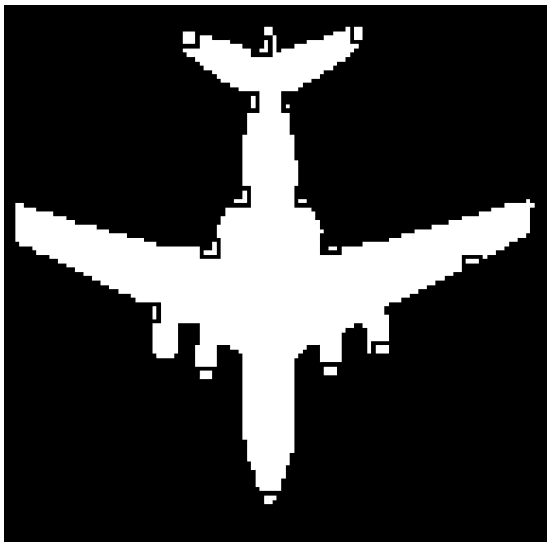
He ve Yung Yöntemi

Şekil 4.32: Karşılaştırma 2

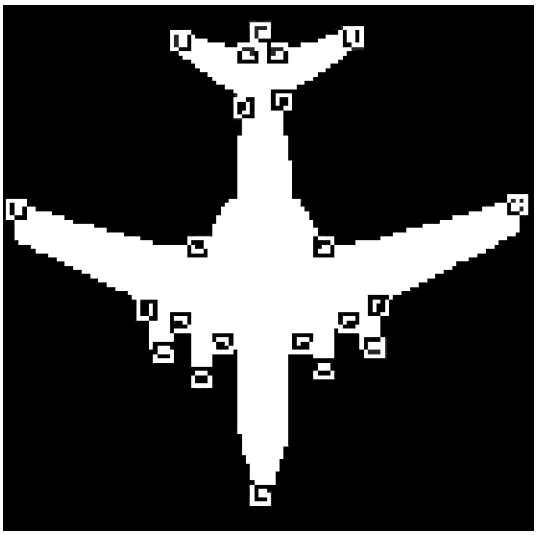
Giriş görüntüsü



Köşe belirleme şablon sonucu



Harris Yöntemi



He ve Yung Yöntemi

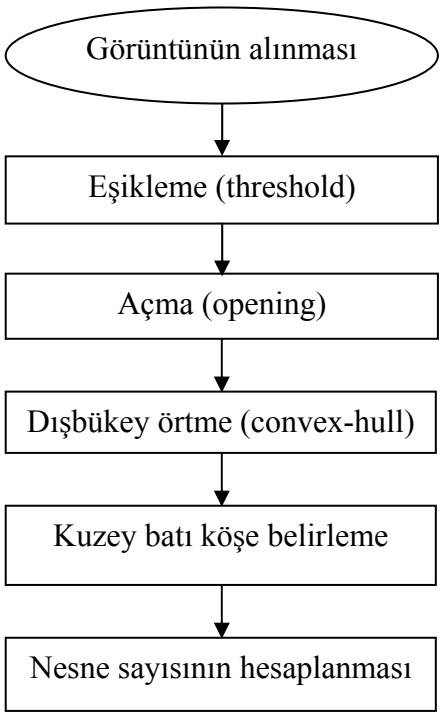
Şekil 4.33: Karşılaştırma 3

Tablo 4.2 de bulunan işlem zamanları Bi-i sisteminin işlem gücünü çok açık bir şekilde gözler önüne sermektedir. Şablonların uygulanarak ACE16k yongası ile köşe belirleme işleminin HSA tabanlı olmayan yöntemlere göre yaklaşık 100 kat daha hızlı bir şekilde çalıştığı görülmektedir. Ayrıca, farklı görüntüler üzerinde farklı sayıda köşelerin belirlenmesi işlemlerini HSA tabanlı olmayan yöntemler farklı sürelerde gerçekleştirirken, ACE16k yongası üzerinde görüntüden bağımsız olarak aynı sürelerde köşeler belirlenmiştir.

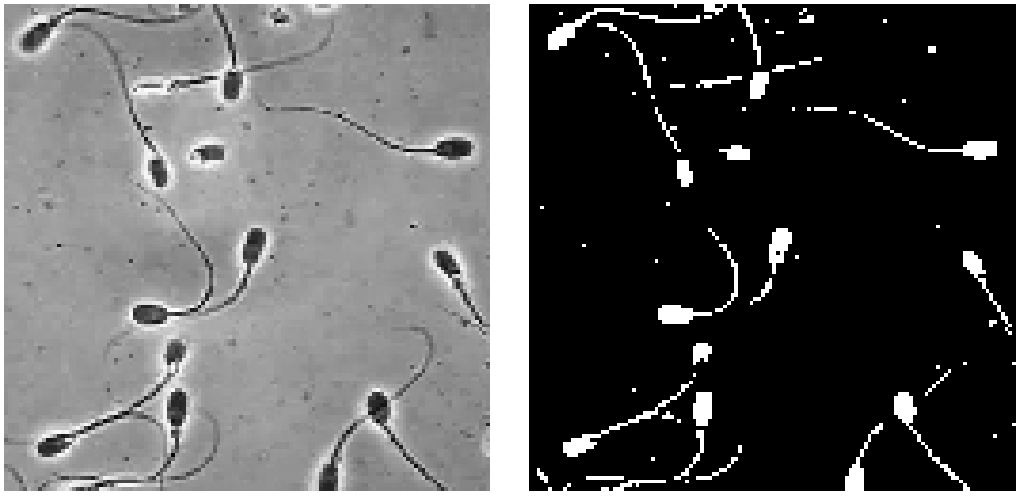
Çeşitli görüntüler üzerinde elde edilen şablonların ve HSA tabanlı olmayan yöntemlerin belirlediği köşeleri gösteren karşılaştırma görüntüleri ile Bölüm 4.3'te bulunan sonuçlara bakıldığında, IAŞT yazılımının bulduğu şablonların yüksek bir oranda köşeleri belirlediği görülmektedir. Doğru köşeleri belirlemede Harris yönteminden daha iyi, He ve Yung yöntemine ise yakın sonuçlar alınmıştır.

4.5. HSA TABANLI BİR NESNE SAYMA ALGORİTMASININ ACE16K YONGASI KULLANILARAK GERÇEKLENMESİ

Tez çalışmasının bu kısmında Bi-i sistemi üzerinde çalışan bir nesne sayma uygulaması geliştirilmiştir. Kullanılan algoritma [47]'e dayanmaktadır. Benzetim olarak gerçekleştirilmiş algoritma ACE16k yongası üzerine uygulanmış ve geliştirilmiştir. Uygulama gri seviyeli bir görüntüde bulunan nesnelerin sayısını vermektedir. Bu işlem her bir nesnenin bir nokta ile temsil edilmesine dayanmaktadır. Nesnelerin noktalara dönüştürülmesi IAŞT yazılımı kullanılarak eğitilmiş olan kuzey-batı (K-B) köşe belirleme şablonu ile gerçekleştirilir. Verilen giriş görüntüsü üzerinde eşikleme, açma ve dışbükey örtme fonksiyonları uygulanarak, görüntü K-B köşe belirleme şablonunun uygulanmasına hazır hale getirilir. Gerçekleştirilmiş olan algoritmanın blok diyagramı Şekil 4.33'te verilmiştir. Kullanılan fonksiyonlar Bi-i SDK kütüphanesinde bulunan ve ACE16k yongası üzerinde çalışabilen fonksiyonlardır. Algoritmanın nasıl çalıştığı örnek bir sperm görüntüsü üzerinde açıklanmıştır. Ayrıca aynı algoritma Bi-i sistemi içerisinde bulunan DSP işlemcisi ve MATLAB üzerinde çalıştırılarak elde edilen yürütme zamanları tablo halinde sunulmuş ve değerlendirmesi yapılmıştır.



Şekil 4.34: Nesne sayma algoritması



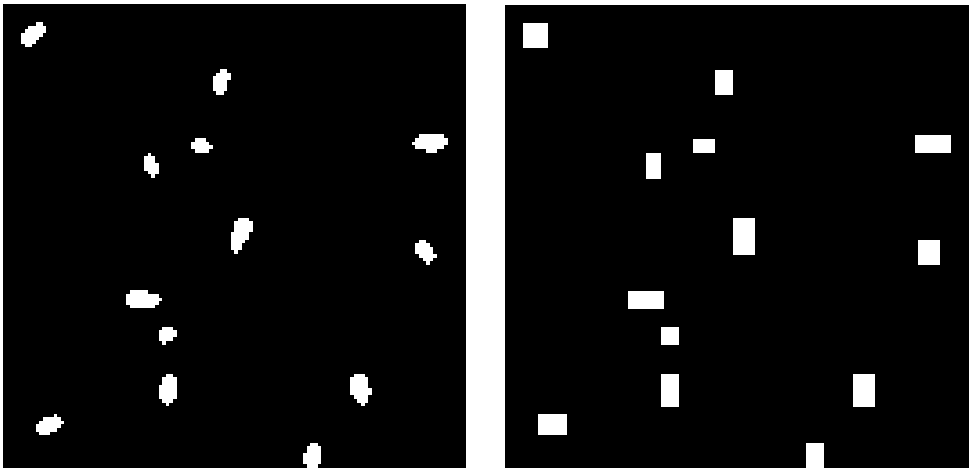
a) **b)**

Şekil 4.35: a) Gri seviyedeki giriş görüntüsü, b) Eşikleme işlemi sonucu

Yukarıdaki algoritmadan da görüldüğü gibi öncelikle işlem gri seviyedeki giriş görüntüsünün alınmasıyla başlar (Şekil 4.34a). Alınan görüntü eşikleme işlemi ile ACE16k yongası kullanılarak ikili görüntü haline getirilir. Bu işlem için *ConvLAMtoLLM* isimli fonksiyon kullanılmıştır. Elde edilen sonuç görüntüsü Şekil 4.34b' de görülmektedir.

Algoritmanın 2. adımı olan Açma işlemi için, elde edilen bu görüntü üzerine *Opening4* fonksiyonu uygulanır. Bu fonksiyon önce *Aşınma* (Erosion) sonrasında *Genişleme* (Dilation) işlemlerini gerçekleştirerek görüntü içerisindeki gürültü olarak adlandırılabilecek olan küçük nesneleri ortadan kaldırır. Aynı zamanda eğer varsa birleşik nesnelerinde ayrılmasını sağlar. Bu fonksiyonun uygulanmasıyla elde edilen sonuç Şekil 4.35a'da gösterilmiştir.

Dışbükey Örtme ile görüntü üzerinde kalan nesneler kuzey-batı köşe belirleme şablonunun uygulanabilmesi için dörtgen haline tamamlanır. Bu işlem için *Convex-Hull* fonksiyonu *CONVEX_90_DISCON* parametresi ile birlikte kullanılmıştır. Nesnelerin dörtgen hallerini içeren görüntü Şekil 4.35b'de verilmiştir.

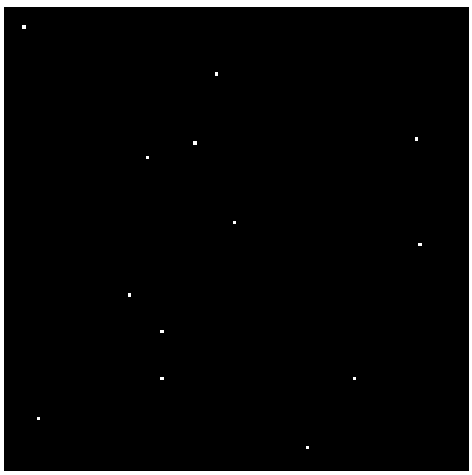


a) **b)**

Şekil 4.36: a) Açma işlemi sonucu, b) Dışbükey örtme işlemi sonucu

Dörtgen haline dönüştürülmüş nesnelere (4.11) ifadesi ile gösterilen sol-üst köşe bulma şablonu uygulanarak her bir nesnenin tek bir nokta ile gösterilmesi sağlanır. Bu işlemde kullanılan kuzey-batı köşe belirleme şablonu IAŞT yazılımı ile eğitilmiştir. Elde edilen sonuç Şekil 4.36'da görülmektedir. Bu noktaların her biri bir nesneyi temsil etmektedir ve bu noktaların sayılmasıyla görüntü içindeki mevcut nesne sayısı hesaplanmış olur. Bu işlem için C++ programlama dili içerisinde *for* döngüsü kullanılmıştır.

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 4.76 & 0\\ 0 & 0 & 0\end{bmatrix} \quad B=\begin{bmatrix}3 & -2.93 & 1.63\\ -1.87 & 3.51 & -0.19\\ -0.49 & -0.12 & 2.3\end{bmatrix} \quad I=5.49 \tag{4.11}$$



Şekil 4.37: Köşelerin belirlenmesi

4.5.1. Yürütme Zamanları Değerlendirmesi

Algoritma ACE16k yongasının yanı sıra DSP ve Matlab üzerinde de gerçeklenerek elde edilen yürütme zamanları ölçülmüştür. Bu zamanlar aşağıdaki tablodan görülebilir.

Tablo 4.3: Nesne sayma algoritmasının değişik ortamlardaki yürütme süreleri

	ACE16k	**DSP**	**Matlab**
Eşikleme	641 µs	841 µs	670 µs
Açma	425 µs	1673 µs	36000 µs
Dışbükey Örtme	1362 µs	96154 µs	9200 µs
K-B Köşe Belirleme	423 µs	-	390000 µs
Nesne Sayma	4169 µs	4169 µs	760 µs
Toplam Süre	7020 µs = 0.007020 s	102837 µs = 0.102 s	436630 µs = 0.436 s

Tablodan görüldüğü gibi, algoritmanın ilk 3 adımı tüm ortamlarda gerçeklenmiştir. Eşikleme işleminin süresi 3 ortam içinde yaklaşık aynıdır. Açma ve dışbükey örtme işlemlerinde en kısa zamanlar ACE16k yongası üzerinde elde edilmiştir. Şablon işlemi DSP üzerinde gerçeklenememektedir. Matlab üzerinde ise şablon işlem süresi MATCNN [56] isimli bir araç kullanılarak ölçülmüştür. Verilen süre K-B şablonunun süresi değildir. Çünkü bu şablon sadece ACE16k yongası üzerinde çalışabilecek şekilde eğitilmiştir.

Toplam yürütme sürelerine bakıldığında en hızlı ortamın ACE16k olduğu açıkça görülmektedir. ACE16k üzerinde algoritmanın çalışma süresi Matlab üzerinde gereken süreden yaklaşık 60 kat daha kısadır. Bunun yanında ACE16k, DSP den de yaklaşık 15 kat daha hızlıdır. Bu sonuçlar HSA tabanlı ACE16k yongasının geleneksel mimariler karşısındaki hız avantajını bir kez daha göstermektedir.

5. TARTIŞMA VE SONUÇ

Bu tez çalışmasında gerçek zamanlı görüntü işleyebilen analog bir HSA çok fonksiyonlu makine olan Bi-i Hücresel görü sistemi üzerinde Iterative Anealing optimizasyon yöntemi kullanılarak oluşturulan şablonlar ile kenar ve köşe belirleme eğitimleri gerçekleştirilmiştir. Bunun için gerçekleştirilen işlemler sırasıyla aşağıda verilmiştir.

İlk olarak genel HSA çok fonksiyonlu makine yapısı ve bu tezde kullanılan gerçek zamanlı görüntü işleyebilen analog bir HSA çok fonksiyonlu makine içeren Bi-i Hücresel görü sistemi ayrıntılı olarak incelenmiştir.

Şablon ve şablon tasarımı ile ilgili bilgiler verilmiştir. Genel şablon tasarım sürecinden ve bu şablon tasarım sürecini etkileyen sebeplerden bahsedilmiştir.

Iterative Annealing optimizasyon yöntemi ve bu yöntemin dayandığı Simulated Annealing optimizasyon yöntemleri hakkında bilgiler verilmiştir.

Bi-i sistemi üzerinde Iterative Anealing optimizasyon yöntemi kullanılarak şablon eğitiminin gerçekleştirilebileceği bir yazılım geliştirmek amacıyla çalışma yapılmıştır. Bu çalışma esnasında IA yönteminin ACE16k yongası üzerinde çalışabilmesi için ne tür değişikliklerin yapılması gerektiği anlatılmıştır. Ardından değiştirilen bu yöntem kullanılarak C++ programlama diliyle Code Composer Studio geliştirme arabirimi üzerinde IAŞT yazılmış ve bu yazılımın nasıl çalıştığı anlatılmıştır.

IAŞT yazılımı kullanılarak kenar ve köşe belirleme eğitimleri gerçekleştirilmiş daha sonra da çeşitli kenar belirleme işlemleri ile bu yazılımın çalışması test edilmiştir. Rasgele değerler ile kare şekli kullanılarak başlayan eğitim işlemleri sonucunda elde edilen şablon değerlerinin daha karmaşık görüntülere uygulanması ile bulunan sonuçların ne ölçüde doğru olduğu kontrol edilmiş ve farklı şekiller ile yapılan eğitim

işlemleri ile daha iyi kenar belirleme şablonu elde edilmiştir. Elde edilen şablonun doğruluğunu gösteren uygulamalar yapılmıştır. Bu uygulamalardan görüldüğü üzere bulunan şablon kenarları gayet iyi bir şekilde belirleyebilmektedir.

IAŞT yazılımının ikili görüntüler üzerinde kenar belirleyecek şekilde şablon üretimini gerçekleştirmesinden sonra, bu aracın gri seviyedeki görüntüler üzerinde de çalışabildiği gösterilmiştir. İkili görüntüler üzerinde üzerinde kenar belirleyen şablon değer seti başlangıç olarak verilerek gri seviyeli görüntüler üzerinde eğitim işlemleri yapılmıştır. Bulunan şablon değerleri yine karmaşık şekillere uygulanarak doğruluğu kontrol edilmiş ve karmaşık şekiller üzerinde yapılan eğitimlerle hem gri seviyeli görüntüler hem de siyah/beyaz görüntüler üzerinde kenar belirleyebilen şablon değerleri elde edilmiştir.

İkili ve gri seviyeli görüntüler üzerinde kenar belirleme şablonlarının eğitimlerinden sonra ACE16k yongası üzerinde çalışabilen köşe belirleme şablonun eğitilebilmesine çalışılmıştır. Kare şekli ile başlayan eğitim işlemleri elde edilen şablon değerlerinin farklı görüntülere uygulanması ile devam etmiştir. Elde edilen sonuçlar tek bir şablon ile köşelerin belirlenemeyeceğini göstermiştir. Köşedeki pikseli çevreleyen komşu piksellerin renk değerlerine göre içbükey (en az 5 beyaz komşu piksel) ve dışbükey (en az 5 siyah komşu piksel) olmak üzere iki tip köşe mevcut olması nedeniyle bu iki tip köşeyi ayrı ayrı bulan şablonların eğitimi yapılmıştır. Bulunan köşe belirleme şablonlarının doğrulukları kontrol edilmiştir. Bu esnada dışbükey köşe belirleme şablonunun eşkenar dörtgen gibi kenarları düz olmayan şekillerde köşeleri doğru belirleyemediği görülmüştür. Bu şekil üzerinde gerçekleştirilen yeni eğitimlerle bulunan şablon çok daha iyi sonuçlar vermiştir.

IAŞT yazılımının etkinliği başka bir tasarım aracı olan CNNOPT ve HSA tabanlı olmayan yöntemlerle karşılaştırılarak ölçülmüştür. Bu karşılaştırmalar sonucunda, işlem zamanlarına bakıldığında, geliştirilen yazılım ile CNNOPT sisteminin neredeyse aynı adım zamanlarına sahip olduğu görülmüştür. Ancak iki sistem arasında sabit diske yazma kısmında bir farklılık söz konusudur. IAŞT yazılımı, algoritması gereği her bir adımda iki adet şablon değer seti üretip, ürettiği bu değer setlerini ACE16k yongasına göndermekte ve bu gönderim işlemi C++ programlama diliyle programlama

yapılmasından dolayı iki kez sabit diske yazım işlemi gerektirmekte olduğundan sistemde oldukça uzun süren aşamaları oluşmaktadır. Buna karşın CNNOPT sisteminde ise kullanılan AMC programlama dilinde şablon değerlerinin ACE16k yongasına gönderilmesi için sabit diske yazım zorunluluğu yoktur. Bu nedenle elde edilen sürelere bakıldığında, geliştirilen IAŞT yazılımının oldukça hızlı çalışabildiği görülmüştür.

CNNOPT ile yapılan köşe belirleme eğitimleri sonuçları ve bu sonuçların çeşitli örnek görüntülere uygulanması ile elde edilen sonuçlar her iki sisteminde bulduğu şablon değerlerinin benzer şekilde köşeleri belirlediğini göstermektedir.

HSA tabanlı olmayan yöntemlerle yapılan işlem zamanları değerlendirmesi Bi-i sisteminin işlem gücünü çok açık bir şekilde gözler önüne sermektedir. Şablonların uygulanarak ACE16k yongası ile köşe belirleme işleminin HSA tabanlı olmayan yöntemlere göre yaklaşık 100 kat daha hızlı bir şekilde çalıştığı görülmektedir. Ayrıca, farklı görüntüler üzerinde farklı sayıda köşelerin belirlenmesi işlemlerini HSA tabanlı olmayan yöntemler farklı sürelerde gerçekleştirirken, ACE16k yongası üzerinde görüntüden bağımsız olarak aynı sürelerde köşeler belirlenmiştir.

Çeşitli görüntüler üzerinde elde edilen şablonların ve HSA tabanlı olmayan yöntemlerin belirlediği köşeleri gösteren karşılaştırma görüntüleri ile Bölüm 4.3'te bulunan sonuçlara bakıldığında, geliştirilen IAŞT yazılımının bulduğu şablonların yüksek bir oranda köşeleri belirlediği görülmektedir. Elde edilen sonuçlarda yanlış köşe belirleme oranı Harris yöntemine göre daha düşük olduğu görülmüştür. Doğru köşeleri belirlemede ise yine Harris yönteminden daha iyi, He ve Yung yöntemine ise yakın sonuçlar alınmıştır.

Tezin son kısmında Bi-i sistemi üzerinde çalışan bir nesne sayma uygulaması gerçekleştirilmiştir. Bu uygulama gri seviyeli görüntüleri giriş olarak alarak içerisinde bulunan nesneleri saymaktadır. Bu işlemi gerçekleştirirken öncelikle gri seviyedeki görüntü ikili görüntüye çevrilir, sonra görüntü üzerindeki gürültülerin temizlenmesi ve eğer varsa birleşik şekillerin ayrılması için açma işlemi kullanılır, görüntüdeki nesneler dörtgene tamamlanır. Nesnelerin dörtgene tamamlanma sebebi her bir nesnenin sadece bir köşe ile temsil edilebilmesini sağlamaktır. Bunun için sol-üst köşe belirleme şablonu

eğitilmiştir. Bu şablon kullanılarak her bir nesne tek bir noktaya indirgenerek bu noktaların sayılması ile görüntü içerisindeki nesne sayısı bulunabilmektedir. Bu algoritmanın çalışması örnek bir görüntü üzerinde açıklanmıştır. Ayrıca DSP ve Matlab ortamlarında da algoritmanın gerçeklenmesiyle elde edilen yürütme zamanları tablo halinde sunulmuştur.

Elde edilen sonuçlar ve yapılan uygulamalar gerçek zamanlı işlem yapabilen analog HSA çok fonksiyonlu makinenin etkinliğini göstermektedir. Böyle bir sistem olan Bi-i Hücresel görü sistemi görüntü işleme konusunda oldukça yüksek performans göstermektedir. Şablon kullanarak köşe belirleme işlemini HSA tabanlı olmayan yöntemlere göre 100 kat daha hızlı yapabilmesi ve nesne sayma algoritmasının Matlab ortamına göre 60 kat daha hızlı olması, ACE16k sisteminin görüntü işleme konusunda çok uygun bir platform olduğunu göstermektedir. Ancak, tüm bu üstün yönlerine karşılık, HSA çok fonksiyonlu makineler mükemmel değildir. Şablon parametrelerinin belli aralıklarda değerler alabilmesi, giriş görüntüsünün boyutunun sabit ve küçük olması (ACE16k için 128x128 piksel boyutu) gibi donanımsal sınırlamalar, elde edilen sonuçların dış etkenler ve üretim süreçlerinden kaynaklanan sorunlar nedeniyle kesin ve kararlı olmaması, aynı giriş ve şablon için farklı sonuçlar üretilebilmesi, programlama sürecindeki zorluklar ve istenilenlerin uygulamaya tam olarak dökülememesi gibi çeşitli eksik yönleri de bulunmaktadır. Bu eksikliklerin HSA çok fonksiyonlu makinelerin gelecek sürümlerinde düzeltilebilmesi gerekmektedir. Yine de, bu sistemlerin kullanılması özellikle hız konusunda önemli avantajlar sağlamaktadır. HSA çok fonksiyonlu makinelerin başka görüntü işleme sistemleri, bilgisayarlar vb. ile birlikte kullanılmaları ile çok daha karmaşık görüntü işleme uygulamaları gerçekleştirilebilir.

KAYNAKLAR

[1] Neumann v. J., 1987, *Papers of John von Neumann on Computing and Computer Theory.* MIT Press and Tomash Publ., Los Angeles/San Francisco.

[2] Bobda C., 2007, *Introduction to Reconfigurable Computing: Architectures, Algorithms, and Applications.* Springer, e-ISBN 978-1-4020-6100-4.

[3] Kahle J. A., (Jul./Sep. 2005), "Introduction to the Cell multiprocessor", *IBM J. RES. & DEV, [online]*, 49, (4/5).

[4] Rodríguez-Vázquez A., 2004, "ACE16k: The Third Generation of Mixed-Signal SIMD-CNN ACE Chips Toward VSoCs", *IEEE Transactions on Circuits and Systems Part I: Regular Papers*, 51, (5), 851–863.

[5] AnaFocus. http://www.anafocus.com/.

[6] Hillier D., de Souza S.X., Suykens J.A.K., 2006, "CNNOPT: Learning dynamics and CNN chip-specific robustness", *10th International Workshop on Cellular Neural Networks and Their Applications*.

[7] Chua L. O. ve Yang L., 1988, "Cellular neural networks :Theory," *IEEE Trans. Circuits* Syst., 35, 1257–1272.

[8] Chua, L.O ve Yang, L., 1988, Cellular Neural Networks: Applications, IEEE Transaction on Circuits and Systems, 35, (10), 1273-1290.

[9] Roska T. ve Chua L. O., 1993, "The CNN Universal Machine," *IEEE Transactions on Circuits and Systems*, 40, 163–173.

[10] Cruz J. M. ve Chua L. O., 1991, "A cnn chip for connected component detection," *IEEE transactions on Circuits and Systems*, 38, 812–817.

[11] Dom´ınguez-Castro R., Espejo S., Rodr´ıguez-V´azquez A. ve Carmona R., 1994, "A cnn universal chip in cmos technology", *IEEE International Workshop CNNA*, 91–96.

[12] Li˜n´an G., Espejo S., Dom´ınguez-Castro R., ve Rodr´ıguez-V´azquez A., 2002, "Ace4k: An analog i/o 64*64 visual microprocessor chip with 7-bit analog accuracy," *International Journal of Circuit Theory and Applications*, 30, (2-3), 89–116.

[13] Chua L. O. ve Roska T., 2002, *Cellular neural networks and visual computing, Foundations and applications*, Cambridge University Press.

[14] Chua L. O. ve Roska T., 1993, "The CNN paradigm," *IEEE Transactions on Circuits and Systems*, 40, 147–156.

[15] Zar´andy A. ve Rekeczky C., 2005, "Bi-i: a standalone ultra high speed cellular vision system," *IEEE Circuits and Systems Magazine*, 5, (2), 36–45.

[16] http://www.analogic-computers.com/

[17] Bi-i Vision System: User Manual.

[18] Kirkpatrick S., Gelatt C.D., Vecchi M.P.,1983, "Optimization by Simulated Annealing", *Science*, 220, (4598), 671-680.

[19] Feiden D., Tetzlaff R., 2001, "Iterative annealing a new efficient optimization method for cellular neural networks", *Image Processing,* 1, 549-552.

[20] De Souza S. X., 2007, "Optimisation and Robustness of Cellular Neural Networks", Phd Thesis, Katholieke Universitiet.

[21] De Souza S. X.., Yalcin M. E., Suykens J. A. K. ve Vandewalle j., 2004, "Toward CNN Chip-specific Robustness," *IEEE Trans. Circ. And Systems-I:Fundamental Theory and Appl.*, 51, (5), 892–902.

[22] Földesy P., Kek L., Zarandy A., 1999, "Fault-tolerant design of analogic CNN templates and algorithms-Part I: The binary output case", *Circuits and Systems I: Fundamental Theory and Applications*, 46, (2), 312-322.

[23] Tetzlaff R., Kunz R. ve Geis G., 1997, "Analysis of cellular neural Networks with parameter deviations", *In Proc. IEEE ECCTD 97*, 650–654.

[24] Kunz R., Tefzlaff R. ve Wolf D., 1996, "SCNN: a Universal Simulator for Cellular Neural Networks". *In Proceedings of The Fourth IEEE International Workshop on Cellular Neural Networks and Their Applications*, 255–260.

[25] Roska T., K´ek L., Nemes L., Zar´andy A., Brendel M. ve Szolgay P., 1998, "CADETWin." *Computer and Automation Institute of the Hungarian Academy of Sciences*.

[26] Zaim A.H., Ersoy C., "Reliable Network Topology Design Using Simulated Annealing".

[27] Rich E., Knight K., Artificial Intelligence.

[28] Zara´ndy A., 1999, "The art of the CNN template design," *Int. J. Circuit Theory Applications,* 27, (1),5-23.

[29] Schönmeyer R., Feiden D., Tetzlaff R., 2002, "Multi-template training for image processing with cellular neural networks". *CNNA 2002*, 523-531.

[30] Schönmeyer R., Feiden D., Tetzlaff R., 2003, "On-chip template training for pattern matching by cellular neural network universal machines (CNN-UM)", *ISCAS 2003*, 3, 14-517 .

[31] Feiden D., Tetzlaff R., 2003, "On-Chip-Training for Cellular Neural Networks using Iterative Annealing", *VLSI circuits and systems*., 5119, 470-477.

[32] Nossek J. A., 1996, "Design and Learning with Cellular Neural Networks.", *Int. J. Circuit Theory and Applications*, 24, (1), 15-24.

[33] Kozek T., Roska T. ve Chua L. O., 1993, "Genetic Algorithm for CNN Template Learning." *IEEE Trans. on Circuits and Systems – I*, 40, (6), 392-402.

[34] Chandler B., Rekeczky C., Nishio Y. ve Ushida A., 1999, "Adaptive Simulated Annealing in CNN Template Learning", *IEICE Trans. Fundamentals*, 82, (2), 398–402.

[35] Hangi M. ve Moschytz G. S., 1998, "Stochastic and Hybrid Approaches Toward Robust Templates", *In Proceedings of IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'98)*, 366–371.

[36] Hangi M. ve Moschytz G. S., 1999, "An Exact and Direct Analytical Method for the Design of Optimally Robust CNN Templates", *IEEE Trans. Circuits and Systems*, 46, (2), 304–311.

[37] Kinget P. ve Steyaert M., 1996, "Evaluation of CNN Template Robustness Toward VLSI Implementation", *International Journal of Circuit Theory and Applications*, 24, (1), 93–110.

[38] Mirzai B., Lim D. ve Moschytz G. S., 1996, "Robust CNN Templates: Theory and Simulations", *In Proceedings of IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'96)*, 393–398.

[39] Nossek J. A., 1996, "Design and Learning with Cellular Neural Networks", *International Journal of Circuit Theory and Applications*, 24, 15–24.

[40] Schuler A. J., Brabec M., Schubel D. ve Nossek J. A., 1994, "Hardware-Oriented Learning for Cellular Neural Networks", *In Proceedings of IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'94)*, 183–188.

[41] Tetzlaff R., Kunz R., Geis G. ve Wolf D., 1998, "Minimizing the effects of tolerance faults on hardware realizations of cellular neural Networks", *In Proceedings*

of The Fifth IEEE International Workshop on Cellular Neural Networks and Their Applications, 385–390.

[42] Tetzlaff R., Kunz R. ve Wolf D., 1999, "Minimizing the effects of parameter deviations on cellular neural Networks", *Int. J. of Circuit Theory and Applications*, 27, (1), 77–86.

[43] Tetzlaff R., Kunz R., Geis G., 1997, "Analysis of cellular neural networks with parameter deviations", In Proc. IEEE ECCTD 97, 650– 654.

[44] Merkwirth C., 2004, "Finite Iteration DT-CNN – New Design and Operating Principle," *in Proc. ISCAS*, 504-507.

[45] Wichard J., Ogorzalek M. ve Merkwirth C., 2005, "Peformance of Finite Iteration DTCNN with Truncated Stationary Templates," *in Proceedings ISCAS*, 4657-4660.

[46] Hängi M. ve Moschytz G. S., 1999, "Analytic and VLSI Specific Design of Robust CNN Templates." *J. VLSI Signal Processing*, 23, 415-417.

[47] Fasih A., Chedjou J. ve Kyamakya K., 2008, "Ultra Fast Object Counting Based-on Cellular Neural Network", *First International Workshop on Nonlinear Dynamcis and Synchronization (INDS'08)*, 181-183.

[48] Harris C. ve Stephens M., 1998, "A Combined Corner and Edge Detector*", Alvey Vision Conference*, s.147-151.

[49] Rosten E. ve Drummond T., 2006, "Machine Learning for High-speed Corner Detection", *In European Conference on Computer Vision*, 430-443.

[50] He X.C. ve Yung N.H.C., 2004, "Curvature Scale Space Corner Detector with Adaptive Threshold and Dynamic Region of Support", *Proceedings of the 17th International Conference on Pattern Recognition,* 791-794.

[51] Zhang X., Lei M., Yang D., Wang Y. ve Ma L., 2007, "Multi-scale Curvature Product for Robust Image Corner Detection in Curvature Scale Space", *Pattern Recognition Letters,* 28, (5), 545-554.

[52] Nain N., Laxmi V., Jain A. K. ve Agarwal R., 2006, "Morphological Edge Detection And Corner Detection Algorithm Using Chain-Encoding", *IPCV'06,* 520-525.

[53] Chang X., Gao L. ve Li Y., 2007, "Corner Detection Based on Morphological Disk Element", *Proceedings of the 2007 American Control Conference,* 1994-1999.

[54] Moravec, H., 1980, *Obstacle avoidance and navigation in the real world by a seeing robot rover. In: tech. report CMU-RI-TR-80-03*, Robotics Institute, Carnegie Mellon University & doctoral dissertation, Stanford University. Carnegie Mellon University.

[55] Canny J.F., 1986, “A Computational Approach to Edge Detection,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8, (6), 679-698.

[56] www.mathworks.com

[57] Arena P., Argentino S., Basile A., Fortuna L. ve Frasca M., 2001, “MARGE: A 3D CNN simulator,” *in Proc. Eur. Conf. Circuit Theory and Design ECCTD’01*, 28–31.

[58] Szolgay P., Làszlò K., Kèk L., Kozek T., Nemes L., Petràs I., Rekeczky C., Szatmàri I., Zaràndi A., Zöld S. ve Roska T., 1999, “The CADETWin application software design system—A tutorial,” *in Proc. Eur. Conf. Circuit Theory and Design CCTD’99*, 58–68.

[59] “CANDY - Multilayer CNN Simulator”, *Analogic and Neural Computing Laboratory, Computer and Automation Research Institute, Hungarian Academy of Sciences, Budapest*, http://lab.analogic.sztaki.hu/Candy/candy.html.

[60] Mokhtarian F. ve Suomela R., 1998, “Curvature Scale Space for Robust Image Corner Detection,” *Int’l Conf. Pattern Recognition*.

ÖZGEÇMİŞ

05/11/1978 tarihinde İstanbul'da doğdu. İstanbul Şehremini Lisesi'nden 1995 yılında mezun oldu. Aynı yıl İ. Ü. Bilgisayar Bilimleri Mühendisliği'ne girdi. 1999 yılında lisans eğitimini tamamlayıp, İ.Ü. Bilgisayar Mühendisliği'nde Yüksek Lisans eğitimine başladı. 2000 yılında İ.Ü. Bilgisayar Mühendisliği'nde Araştırma Görevlisi olarak göreve başladı. 2003 yılında Yüksek lisans eğitimini bitirip, İ.Ü. Bilgisayar Mühendisliği'nde Doktora eğitimine başladı. Halen Araştırma Görevlisi olarak görevine devam etmektedir.