T.C.

AKDENİZ ÜNİVERSİTESİ

FPGA PLATFORMU İÇİN VIVADO HLS TABANLI SURF ALGORİTMASININ GERÇEKLENMESİ

Hüseyin ÖZDEMİR

FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

OCAK 2018

ANTALYA

T.C.

AKDENİZ ÜNİVERSİTESİ

FPGA PLATFORMU İÇİN VIVADO HLS TABANLI SURF ALGORİTMASININ GERÇEKLENMESİ

Hüseyin ÖZDEMİR

FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

OCAK 2018

ANTALYA

T.C.
AKDENİZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FPGA PLATFORMU İÇİN VIVADO HLS TABANLI SURF ALGORİTMASININ GERÇEKLENMESİ

Hüseyin ÖZDEMİR

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

Bu tez 10/01/2018 tarihinde jüri tarafından oybirliği ile kabul edilmiştir.

Yrd. Doç. Dr. Övünç POLAT (Danışman) ……..
Yrd. Doç. Dr. H. Feza CARLAK ……..
Yrd. Doç. Dr. Özge ÖZTİMUR KARADAĞ ……..

ÖZET

FPGA PLATFORMU İÇİN VIVADO HLS TABANLI SURF ALGORİTMASININ GERÇEKLENMESİ

Hüseyin ÖZDEMİR

Yüksek Lisans Tezi, Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Övünç POLAT

Ocak 2018; 68 sayfa

SURF algoritması, görüntü işlemede kullanılan, görüntünün boyut, renk, kontrast gibi özellik değişimlerinden etkilenmeyen bir yöntemdir**.** Bu çalışmada, işlem yoğunluğu çok olan SURF algoritması, Vivado HLS aracı ile oluşturulmuştur. HLS ile donanım tanımlama dilleri (verilog, VHDL, vb.) kullanılmadan C, C++,vb. diller kullanılarak algoritmalar gerçeklenebilmektedir. SURF algoritması C dili ile gerçekleştirilmiştir. HLS'de bulunan direktifler ile sistemin donanımsal optimizasyonu gerçekleştirilmiştir. Bu kapsamda, sistemin daha hızlı olması ve saklayıcıların daha az yer kaplaması için veri tipi olarak kayan nokta (floating-point) yerine sabit nokta (fixed-point) seçilmiştir. Ayrıca uygun kod blokları paralel çalıştırılarak çalışma zamanı azaltılmıştır. SURF algoritmasının başarısına etki eden parametrelerin belirlenebilmesi için "Genetik Algoritma" kullanılmıştır. Tespit edilen parametreler kullanıldığında başarı oranının önemli ölçüde arttığı gözlemlenmiştir.

Önerilen metot karakter boyutu ve dönmeden bağımsız sabit fontlu rakam tanıma uygulaması için denenmiştir. Yapılan çalışmada rakamlardan oluşturulan resimler ile testler gerçekleştirilmiştir. Her rakam için referans resmi ve test resimleri oluşturulmuştur. Test resimleri için rakamların farklı büyüklük ve rotasyonda (±10) olduğu durumlar seçilmiştir. Böylece referans resimleri ile test resimlerinin aynı olmaması amaçlanmıştır. Uygun parametre değerleri matlab ile tespit edildikten sonra HLS'de kullanılmıştır. Önerilen optimize edilmiş SURF yapısı kullanılarak yüksek başarı oranları elde edilmiştir.

ANAHTAR KELİMELER: Alan Programlanabilir Kapı Dizisi (FPGA), Hızlandırılmış Gürbüz Özellikler (SURF), OPENSURF, Yüksek-Seviye Sentez (HLS).

JÜRİ: Yrd. Doç. Dr. Övünç POLAT

Yrd. Doç. Dr. H. Feza CARLAK

Yrd. Doç. Dr. Özge ÖZTİMUR KARADAĞ

ABSTRACT

REALIZATION OF SURF ALGORITHM BASED ON VIVADO HLS FOR FPGA PLATFORM

Hüseyin ÖZDEMİR

M.Sc. Thesis in Electrical-Electronics Engineering

Supervisor: Assist. Prof. Övünç POLAT

January 2018, 68 pages.

The SURF algorithm is a method which is used in image processing that is not affected by feature changes such as size, color and contrast. In this study, the SURF algorithm which has a lot of processes, was created with Vivado HLS tool. HLS can be implemented using languages as C, C++, etc. without using hardware description languages such as verilog, VHDL, etc. The SURF algorithm was implemented with a C-language. Hardware optimization of system has been implemented with directives in HLS. In this context, the data type has been chosen as "fixed-point" instead of "floating-point" for the faster system and less storage space. In addition, the runtime was reduced by executing the appropriate code blocks by the parallel operation. "Genetic Algorithm" has been used to determine the parameters which are affected the success of the SURF algorithm. It has been observed that the success rate significantly increased when the determined parameters were used.

The proposed method has been tested for the character recognition application with the fixed font which is independent from the "rotation and character size". In this study, the tests were carried out with the images formed by the numbers. Reference picture and test pictures for all figures were created. Test pictures with different sizes and rotations (± 10) were selected. Thus, it was aimed that the reference and test pictures was not the same. The appropriate parameter values were used in HLS after being determined with matlab. The rate of high success were obtained using the proposed optimized SURF structure.

KEYWORDS: Field Programmable Gate Array (FPGA), High-Level Synthesis (HLS), OPENSURF, Speeded-Up Robust Features (SURF)**.**

COMMITTEE: Assist. Prof. Övünç POLAT

Assist. Prof. H. Feza CARLAK

Assist. Prof. Özge ÖZTİMUR KARADAĞ

ÖNSÖZ

Yüksek Lisans dönemim boyunca yardımlarını ve desteklerini benden esirgemeyen danışman hocam Yrd.Doç.Dr. Övünç POLAT ve eski danışman hocam Dr. Refik SEVER'e teşekkürü bir borç bilirim.

Maddi manevi her zaman yanımda olan aileme sevgilerimi sunarım.

İÇİNDEKİLER

AKADEMİK BEYAN

Yüksek Lisans Tezi olarak sunduğum " FPGA PLATFORMU İÇİN VIVADO HLS TABANLI SURF ALGORİTMASININ GERÇEKLENMESİ" adlı bu çalışmanın, akademik kurallar ve etik değerlere uygun olarak yazıldığını belirtir, bu tez çalışmasında bana ait olmayan tüm bilgilerin kaynağını gösterdiğimi beyan ederim.

10/01/2018

Hüseyin ÖZDEMİR

.....

SİMGELER VE KISALTMALAR

Simgeler

σ : Standart sapma

Kısaltmalar

HLS : High-Level Synthesis

ROI : Region Of Interest

FPGA : Field Programmable Gate Array

LUT : Look-Up Table

DSP : Digital Signal Processing

FF : Flip-Flop

SURF : Speeded-Up Robust Features

SIFT : Scale-Invariant Feature Transform

NMS : Non-Maximal Suppression

PCA : Principal Components Analysis

GLOH : Gradient Location And Orientation Histogram

VHDL : Very high speed integrated circuit Hardware Description Language

GA : Genetik Algoritma

ŞEKİLLER

ÇİZELGELER DİZİNİ

1. GİRİŞ

SURF (Speeded Up Robust Features) algoritması, görüntü üzerinde tespit edilmesi istenen bir nesnenin sahip olduğu en iyi özellikleri tespit etmeye çalışır ve işlem sonucunu hızlı bir şekilde sonuçlandırabilir. Algoritmanın diğer birçok özellik çıkartma algoritmalarına kıyasla daha iyi performans göstermesi, işlemleri sırasında integral görüntü kullanmasından dolayıdır. Görüntü eşleştirmeleri için arama üç ana başlık ile değerlendirilebilir.

İlk olarak ilgi noktaları (resimdeki köşeler gibi) seçilir. Bir ilgi noktası dedektörünün en değerli özelliği tekrarlanabilirliği, yani farklı görüş koşulları altında aynı ilgi noktalarını güvenilir bir şekilde bulup bulamayacağının ölçüsüdür.

Daha sonra, her ilgi noktasının komşusu bir özellik vektörü ile temsil edilir. Bu tanımlayıcı, ayırt edici olmalı ve aynı zamanda gürültüye, algılama hatalarına, geometrik ve fotometrik deformasyonlara karşı dayanıklı olmalıdır.

Son olarak, tanımlayıcı vektörler farklı görüntüler arasında eşleştirilir. Eşleştirme çoğu zaman vektörler arasındaki mesafeye dayanır (Mahalanobis veya Öklid mesafesi). Tanımlayıcının boyutu, bunun ne kadar zaman aldığı üzerinde doğrudan bir etkiye sahiptir ve daha düşük boyutlar olması arzu edilir.

Bu çalışmada SURF algoritmasının matlab ile analizlerinin yapılarak algoritma başarısının artırılması ve HLS ile FPGA üzerinde gerçeklenmesi amaçlanmıştır. Bu kapsamda algoritma, varsayılan parametreler ile test edilerek başarı oranları tespit edilmiştir. Başarı oranlarını arttırmak için genetik algoritması ile optimizasyon uygulanarak parametreler için en uygun değerlerin tespiti amaçlanmıştır. Ayrıca kullanılan hafızanın ve çalışma süresinin azaltılması için tanımlanan saklayıcı tiplerinin sabit-nokta (fixed-point) olarak seçilmesi amaçlanmıştır.

OPENSURF algoritması HLS'ye entegre edilebilecek şekilde bölümlere ayrılarak yeniden düzenlenmiştir (Şekil 4.21 (dönme kontrolü ile ilgili kısımlar iptal edilmiştir), Şekil 4.20). "HLS_0" modülü integral görüntünün oluşturulması için kullanılmaktadır. "HLS_1" modülü ile belirli bir ölçek-uzayı seviyesindeki determinant tepki haritası (response map) belirlenir. "HLS_2" modülünde aday ilgi nokta setini bulmak için En Büyük Olmayanı Bastırma (non-maximal suppression) yöntemi gerçekleştirilir. "HLS_3" modülünde tespit edilen aday ilgi noktasının, gerçek bir ilgi noktası olarak kabul edilip edilmeyeceğine karar verilir. Belirlenen ilgi noktaları için tanımlayıcı vektörleri "HLS_4" modülünde oluşturulur. "HLS_5" modülünde ise referans görüntü için ve arama yapılacak görüntü için ayrı ayrı SURF algoritması kullanılarak ilgi noktaları tespit edilir (ilk beş adım her iki resim için de uygulanır). İki resimde tespit edilen ilgi noktalarının benzerlikleri tanımlayıcı vektörler kullanılarak belirlenir. Her ilgi noktası 64 boyutlu bir vektör ile temsil edildiğinden dolayı, iki resimde bulunan her bir vektör arasındaki mesafe bilgisi hesaplanır ve en yakın olan vektörden başlanarak ilgi noktaları tekrar sıralanır. Bu sıralama sonucuna göre iki resimde bulunan ilgi noktaları eşleşme potansiyeline göre sıralanır.

Referans ve test görüntüleri için 45x80 ebatlarında rakam resimleri oluşturulmuştur. Referans olarak 10 tane resim oluşturulmuştur. Test veri setinde

kullanılmak üzere referans görüntü ile aynı olmayacak şekilde farklı büyüklükte ve rotasyonda rakamlar kullanılarak resimler oluşturulmuştur. Sonuç olarak referans için 10 tane resim, test veri setinde kullanılmak üzere toplam 80 tane resim oluşturulmuştur.

Referans resmi olarak 45x80 ebatlarında resim kullanılırken, test resmi olarak 45x80 ebatlarında on adet resmin birleştirilmesiyle oluşturulan 45x800 ebatlarında resim kullanılmıştır. Buradaki amaç, test veri setinden seçilerek oluşturulan resimde her bir rakamın bulunmasıdır. Böylece SURF algoritması kullanılarak test veri setinden seçilerek oluşturulan her bir resimden ilgili rakamın (tanınması istenen rakamın) belirlenmesi amaçlanır.

Referans rakamının hangisi olduğu ve ilgili rakamın rastgele oluşturulan test resminin hangi bölgesinde olacağı bilindiği için başarı oranı tespiti yapılabilmektedir. Yani, referans resminde kullanılan rakam için ilgi noktaları tespit edilir. Aynı şekilde test resmi için ilgi noktaları tespit edilir. Bulunan ilgi noktaları birbirleri ile karşılaştırılarak sıralanır ve en çok benzeyenler tespit edilir. Test görüntüsünde referans ilgi noktalarına en çok benzeyen ilgi noktalarının dağılımı test başarısını ifade eder. İlgi noktalarının dağılımının tamamı aranan rakamın bölgesinde ise başarı oranı %100 olarak ifade edilir.

Varsayılan parametre değerleri ile elde edilen başarı oranını arttırmak için GA optimizasyon algoritması kullanılmıştır. Optimizasyon algoritması kullanılarak tespit edilen ilgi noktalarının sayısının %50 üzeri ve %70'den üzeri olmak üzere farklı iki durum için optimizasyon algoritması çalıştırılmıştır. Bu iki farklı durum için optimizasyon sonucunda da varsayılan parametrelere göre önemli ölçüde başarı oranının arttığı gözlemlenmiştir.

Optimizasyon algoritması ile belirlenen parametre değerleri HLS'de kullanılmıştır. HLS'de yapılan testler için 45x80 ebatlarında referans ve test resimleri kullanılmıştır.

2. KAYNAK TARAMASI

Bay ve arkadaşları, performanstan ödün vermeyen, hızlı hesaplama yapabilen, yeterince ayırt edici özellikleri bulabilen, tanımlayıcı boyutunu ve karmaşıklığını azaltan bir dedektör geliştirmeyi amaçlamışlardır (Bay vd. 2006). Bu kapsamda SURF dedektörünü geliştirmek için daha önceden yapılan çalışmalardan faydalanmışlardır (Lindeberg 1998, Lowe 2004, Mikolajczyk vd. 2002, Ke vd. 2004, Tuytelaars vd. 2000, Matas vd. 2002). Literatürde çok çeşitli dedektörler ve tanımlayıcılar önerilmiştir.

En çok kullanılan ilgi noktası dedektörlerinden biri 1988 de önerilen Harris köşe dedektörüdür (Harris vd. 1988). İkinci moment matrisinin özdeğerlerine göre hesaplanır. Ancak ölçekten bağımsız değildir.

Lindeberg, otomatik ölçek seçimi kavramını getirmiştir (Lindeberg 1998). Bu, bir görüntüdeki her biri kendi karakteristik ölçeğine sahip ilgi noktalarını tespit etmeyi sağlar. Blob benzeri yapıları tespit etmek amacıyla hem Hessian matrisinin determinantını hem de Laplacian (Hessian matrisinin işaretine tekabül eder)'ı denemiştir.

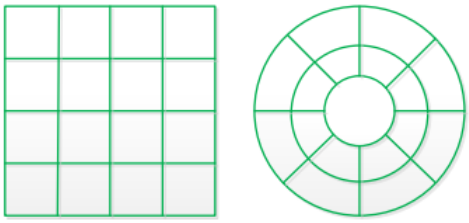
Mikolajczyk ve Schmid, Lindeberg'in yöntemini geliştirerek, yüksek tekrarlanabilirlik ve sağlam özelliklere sahip ölçekten etkilenmeyen Harris-Laplace ile Hessian-Laplace'ı oluşturmuşlardır (Mikolajczyk vd. 2002). Konum seçimi için Harris ölçümü veya Hessian matrisinin determinantını, ölçek seçimi için ise Laplacian'ı kullanmışlardır. Lowe , hız üzerine odaklanarak Gaussların Laplace'ini (LoG) ve Gaussların Farkı (DoG) filtrelerini kullanmıştır (Lowe 1999). LoG algoritması ile Gauss filtresine Laplace alınarak işlem yapılır. DoG ise ölçek uzayı için kullanılır. Çünkü, Gauss'un Laplace normalizasyonu için stabil ve verimlidir.

Ölçekten etkilenmeyen başka ilgi noktası tanımlayıcıları da mevcuttur. Kadir ve Brady'nin önerdiği belirgin bölge dedektörü bölgedeki entropiyi en üst düzeye çıkarır (Kadir vd 2001). Jurie ve arkadaşları tarafından önerilen diğer bir dedektör ise kenara dayalı bir algoritmadır (Jurie vd. 2004). Fakat bu algoritma hızlanma için çok elverişli değildir.

Bay ve arkadaşları mevcut dedektörleri incelemişler ve Hessian temelli dedektörlerin Harris temelli benzerlerinden daha istikrarlı ve tekrarlanabilir olduklarına karar vermişlerdir (Bay vd. 2006). İşaretinden (Laplacian) ziyade Hessian matrisinin determinantını kullanmanın daha avantajlı olduğu önerilmiştir.

SIFT (Scale Invariant Feature Transform) olarak adlandırılan tanımlayıcı küçük deformasyon ve lokalizasyon hatalarına karşı dayanıklıdır (Lowe 2004). Tanımlayıcıların hesaplanmasında anahtar nokta (key-point) etrafında 16x16 boyutunda bölgeler oluşturulur. Bu bölge 4x4 lük parçalara ayrılır. Bu şekilde 16 adet parça oluşturulur. Her parça için, gradyan yönelimleri 45 derecelik 8 farklı açı aralığına etiketlenmiş histogramlar oluşturulur. Bu vektörün 128 elemanı bulunmaktadır. GLOH betimleyicileri radyal yönde logaritmik kutupsal 3 bölmeye ve açısal yönde 8 bölgeye ayrılır. Bu şekilde 17 adet parça elde edilir (Şekil 2.1). Merkez bölge açısal yönde bölgeye ayrılmaz. Gradyan yönelimleri de 16 farklı açı aralığına etiketlenmiş histogram oluşturulur. 16 açı aralığına etiketlenmiş 17 histogramlı vektörden 272 elemanlı

tanımlayıcı vektör elde edilir. Elde edilen tanımlayıcı vektör Temel bileşen analizi kullanılarak boyutu indirgenir. Temel bileşen analizi yöntemi için kovaryans elde edilir. En büyük 128 özvektör tanımlayıcı vektör olarak kullanılır (Alpaslan 2013). GLOH tanımlayıcısı yapılandırılmış sahnelerdeki nokta eşleme uygulamalarında daha iyi performans gösterirken, SIFT tanımlayıcıları doku tabanlı imgelerde daha iyi performans göstermektedir.



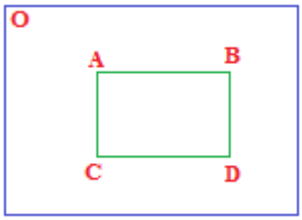
Şekil 2.1. Aday ilgi noktalarının tespiti için kullanılan En Büyük Olmayanı Bastırma Yöntemi (Non-Maximal Suppression) (Evans 2009)

Bu temel düzen üzerinde çeşitli iyileştirmeler önerilmiştir. Ke ve Sukthankar gradyent görüntü üzerinde PCA (Principal Component Analysis) uygulamıştır (Ke vd. 2004). PCA-SIFT olarak adlandırılan bu algoritma, eşleştirme için hızlı fakat SIFT algoritmasından daha az ayırıcı olduğu ispatlanan 36 boyutlu bir tanımlayıcı üretmektedir (Mikolajczyk vd. 2005). Yavaş öznitelik hesaplaması, hızlı eşleme etkisini azaltmaktadır. Ayrıca aynı çalışmada GLOH (Gradient location and orientation histogram) adı verilen SIFT'ın bir değişik versiyonu önerilmiştir. Ancak, bu algoritma daha karmaşıktır. GLOH öznitelik tanımlayıcısı, SIFT algoritmasından 2 temel fark ile ayrılır. İlk olarak, kullanılan ızgara konumunun değiştirilerek öznitelik vektör boyutunun artırılmasıdır. Diğer fark ise oluşturulan yüksek boyutlu öznitelik tanımlayıcılarının PCA (Temel Bileşen Analizi) algoritması ile boyutunun azaltılmasıdır.

SURF (Speeded Up Robust Features) algoritması Bay ve arkadaşları tarafından oluşturulan bir dedektör tanımlayıcıdır. SURF dedektörü, Hessian matrisini temel alır fakat aynı DoG algoritmasının basit bir Laplacian temelli bir dedektör olduğu gibi burada da basit bir yaklaşım kullanılır. Hesaplama süresinin azaltılması için integral görüntüler kullanılmıştır. Bu nedenle Fast-Hessian dedektörü olarak ifade edilmiştir. Ayrıca, tanımlayıcı ilgi noktası komşuluğundaki Haar dalgacık yanıtlarının dağılımını tanımlar. Yine burada da integral görüntüler kullanılarak işlemler yapılmaktadır ve sadece 64 boyutlu vektör kullanılarak özellik tanımlama ve eşleme zamanı azaltılmıştır. Tanımlayıcının hızını artırmanın yanı sıra sağlamlığını arttırmak için de Laplace'in işaretini referans alan bir indeksleme adımı tanımlanmıştır (Bay vd. 2006).

İntegral görüntü kullanılarak kutu tipi konvolüsyon filtresi daha hızlı bir şekilde uygulanmıştır (Viola vd. 2001). Tüm pikseller için integral görüntü hesaplandıktan sonra, herhangi bir dikdörtgensel bölgenin alanı içerisindeki integral görüntü

değerlerinin toplam yoğunluğunu bulmak için, dikdörtgen boyutu önemli olmaksızın aynı işlem yapılarak tespit edilebilmektedir. Bu durum Şekil 2.2'de özetlenmiştir.



Şekil 2.2. Görüntü üzerindeki herhangi bir dikdörtgensel alanın integral görüntü ile temsil edilmesi

SURF algoritması HLS ile C kodu kullanılarak gerçekleştirilebilmektedir (Φαλιάγκας, K. 2013). HLS ile donanım tanıma dilleri (verilog, VHDL gibi) kullanılmadan algoritmalar gerçekleştirimi mümkündür. Κωνσταντίνος'in yapmış olduğu çalışmada HLS kullanılarak SURF algoritması C kodu ile gerçekleştirilmiştir. Burada, saklayıcıları tanımlarken kullanılan veri tipi olarak "float" ve "integer" seçilmiştir. HLS'de var olan direktifler kullanılarak sistemin optimize edilmesi sağlanmıştır. Sistem yazılımsal bir işlemci çekirdeği olan Microblaze ile çalışacak şekilde tasarlanmıştır.

Bu çalışmada Κωνσταντίνος'in aksine saklayıcıların tanımlanması için veri tipi olarak sabit-nokta (fixed-point) seçilmiştir. Böylece hafıza kullanımının azaltılması amaçlanmıştır. SURF algoritması varsayılan olarak ± 15 dereceye kadar rotasyon değişmezliğini desteklediği için, algoritma içerisindeki dönme ile ilgili yapı çıkarılarak sistemin daha hızlı çalışabilmesi amaçlanmıştır. Ayrıca, SURF algoritmasında başarı oranının daha çok olması için, SURF algoritmasında kullanılan eşik değeri ve oktav – 1 değerleri genetik algoritması ile optimize edilerek başarının en yüksek olduğu değerlerle kullanılmıştır. Burada elde edilen parametreler HLS'de kullanılarak en iyi başarı oranının oluşturulması amaçlanmıştır.

Bu çalışmada uygun parametrelerin belirlenmesi ve sistemin test edilmesi için, rakamlardan oluşturulan resimler kullanılmıştır. Farklı büyüklükte ve farklı rotasyondaki rakamların tespit edilmesi amaçlanmıştır. Font tanıma ve karakter tanıma ile ilgili literatürde çeşitli kaynaklar mevcuttur. Zahedi, M. ve Eslami, S. tarafından önerilen yöntem SIFT temeline dayanan otomatik Farsça/Arapça fontlarını tanıyabilmektedir. SIFT algoritması ölçek ve rotasyon değişmezliğini desteklediği için bu gibi etmenlerden etkilenmez. Bir veri tabanındaki 1400 metin görüntüsü üzerinde yapılan testlerde %100 tanıma oranı elde edilmiştir. Ayrıca, büyük veri tabanlarında sistemin yavaş çalışabileceği, bunun için SIFT yerine benzer başarı oranına sahip ve daha hızlı çalışabilen SURF algoritması önerilmiştir. Benzer şekilde Jamjuntr, P. ve Dejdumrong, N. tarafından Tayland yazı tipini tanımlamak için SIFT algoritması kullanılmıştır ve yapılan testlerde %97.37 başarı oranı sağlanmıştır.

3. MATERYAL VE METOT

3.1. SURF Algoritması

3.1.1. Giriş

SURF, özellik eşleştirme için kullanılan bir algoritmadır. SIFT algoritması temel alınarak geliştirilmiştir. SIFT bir resimde var olan anahtar noktaları tespit edip, bu noktalar üzerinden tanımlayıcılar yardımıyla öznitelik hesaplamaktadır. Bu algoritmanın en büyük avantajı, anahtar noktaları bulurken resmin farklı yönlerde çevrilmesi, boyutunun değişik olması ve resimdeki ışık yoğunluğunun değişkenlik göstermesi gibi durumlardan çok fazla etkilenmemesidir (Lowe 2004). SURF algoritması SIFT'e göre daha hızlı ve etkili çalışabilmektedir.

SIFT algoritmasıyla bir görüntüden çok sayıda öznitelik çıkarılabilir. Çıkarılan öznitelik vektörlerinin her biri 128 boyutludur. Öznitelik çıkarımı için görüntü farklı ölçeklerde incelenerek önemli noktalar bulunur. Görüntünün farklı ölçeklerde incelenmesi için Gauss'ların Farkı yöntemi kullanılır. Bu yöntemde görüntüye farklı varyanslı Gauss filtreleri uygulanır. Bu filtrelerin uygulanmasıyla görüntü farklı miktarlarda bulanıklaştırılmış olur. Bu görüntülerin farkı alınarak görüntüdeki kenarlar ve köşeler elde edilir. Bu farkların bazı yöntemlerle elenmesiyle önemli noktalar bulunur. Önemli noktalarda ve komşularında gradyanlar hesaplanır. Her önemli nokta için bir genlik ve yön bilgisi hesaplanır. Bir Gauss penceresi kullanılarak önemli noktaya yakın olan noktaların etkisi arttırılırken, uzak olanlarınki azaltılır. Hesaplanan yön histogramları kullanılarak öznitelik vektörleri elde edilir (Kutluk 2012).

SIFT algoritması görüntü üzerindeki ayırt edici noktaları algılar ve her ayırt edici nokta için tanımlayıcılar çıkarır. Çıkarılan bu ayırt edici noktalar rotasyon, ölçek gibi özelliklerden bağımsızdır. SIFT algoritmasının genel adımları şu şekildedir:

- Gauss ölçek uzayı hesaplanır.
- Gauss farkı bulunur.
- Aday olabilecek anahtar noktalar tespit edilir.
- Kararsız olan noktalar filtrelenir.
- Anahtar noktalara birer yön atanır.
- Anahtar noktaların her biri için ayırt edici tanımlayıcılar oluşturulur.

SURF algoritması görüntü üzerindeki ilgi noktalarını yerelleştirmeye çalışır. İlgi noktalarının genellikle sahip oldukları temel özellikler şu şekildedir (Bouris vd. 2010):

- Matematiksel olarak iyi tanımlanmış olmalı.
- Görüntü uzayında iyi tanımlanmış bir konuma sahip olmalı.
- Çevresindeki yerel görüntü yapısı bilgi bakımından zengin olmalı.
- Görüntü değişimi, döndürme, aydınlatma farklılıkları, perspektif değişikliği gibi etkilerden çok fazla etkilenmemelidir.

Bu özelliklere sahip olma potansiyeli yüksek olan noktalar ilgi noktası olarak seçilir ve bu noktalara göre işlemler devam ettirilir.

SURF temel olarak aşağıdaki adımlardan oluşur (Bouris vd. 2010):

- Görüntünün her pikseli için integral görüntü değeri hesaplanır.
- Belirli bir ölçek-uzayı seviyesinde determinant tepki haritası (response map) belirlenir.
- Determinant tepki haritasının ölçek normalizasyonu yapılır.
- Ölçek uzayındaki ilgi noktalarını yerelleştirmek için En Büyük Olmayanı Bastırma yöntemi gerçekleştirilir (non-maximal suppression).
- En Büyük Olmayanı Bastırma yönteminin sonuçları daha önceden belirlenen bir eşik değeri ile karşılaştırılır.
- Belirli bir eşitlik temel alınarak ölçek uzayına ilgi noktası yerelleştirme işlemi gerçekleştirilir.
- Her ilgi noktasının yönü hesaplanır.

SURF algoritmasının yüksek performans ve doğruluğu integral görüntü kullanımına bağlanabilir (Viola vd. 2001). Ayrıca SURF dedektörünün verimli olması ve doğru sonuçlar vermesi Hessian matrisi kullanımından kaynaklanmaktadır. Örnek olarak sürekli fonksiyon f'nin Hessian matrisi, (3.1)'de gösterildiği gibi f fonksiyonunun kısmi türevlerinin matrisi iken, Hessian matrisinin determinantı (3.2)'de gösterildiği gibi hesaplanmaktadır (Bouris vd. 2010).

$$H(f(x,y)) = \begin{bmatrix} \frac{\theta^2 f}{\theta x^2} & \frac{\theta^2 f}{\theta x \theta y} \\ \frac{\theta^2 f}{\theta x \theta y} & \frac{\theta^2 f}{\theta y^2} \end{bmatrix} \tag{3.1}$$

$$\det(H) = \frac{\theta^2 f}{\theta x^2}\frac{\theta^2 f}{\theta y^2} - \left(\frac{\theta^2 f}{\theta x \theta y}\right)^2 \tag{3.2}$$

Nesneleri farklı ölçeklerde tespit etme gerekliliği ölçek-uzayı kavramının ortaya çıkmasına sebep olmuştur (Lindeberg 1996). Bir ölçek-uzayı, olası tüm ölçekler arasındaki maksimum ve minimum değerleri bulmak için kullanılan sürekli bir işlevdir (Witkin 1983). SURF algoritması ölçek-uzayını, orijinal resme artan boyuttaki kutu filtrelerini uygulayarak oluşturmaktadır. Bu durumun performans artırma sebepleri:

- Ölçek-uzay piramidinin çoklu katmanlarının aynı anda işlenmesine izin vermesi
- Görüntünün alt örneklerinin oluşturulmadan işlem yapılmasıdır. Yani farklı ölçek durumları için görüntünün ölçeği değiştirilmeden filtrelerin ölçeği değiştirilerek işlem yapılmasıdır.

Ölçek uzayı, bir oktavın belirli filtrelerle hesaplanan tepki haritaları dizisine atıfta bulunduğu oktav sayısına bölünür. Ölçek uzayının oluşturulması, en küçük ölçek için görüntünün determinant cevabını hesaplayan 9x9 bir filtreden başlar (bu filtre, σ = 1.2 ile gerçek değerli bir Gauss dağılımına karşılık gelir, burada "σ" filtrenin ölçeğidir).

Her oktav dört aralığa (interval) bölünmüştür ve filtre boyutu denklem (3.3) ile gösterilmektedir. Sonraki katmanlar, aynı filtre-yerleşim oranını koruyarak filtrelerin yukarı ölçeklendirilmesiyle elde edilir. Filtre boyutu arttıkça, ilişkili Gauss ölçeğinin değeri de artar ve formül (3.4) ile hesaplanır:

$$Filtre\ Boyutu = 3(2^{oktav} xaralik + 1) \tag{3.3}$$

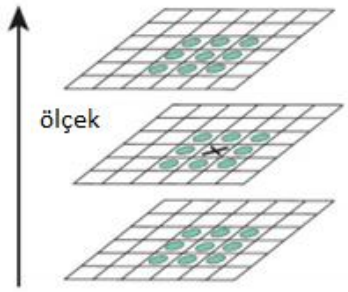
$$\begin{aligned} \sigma_{yaklasik} &= Mevcut\ Filtre\ Boyutu\ .\frac{Temel\ Filtre\ Olcegi}{Temel\ Filtre\ Boyutu} \\ &= Mevcut\ Filtre\ Boyutu\ \frac{1.2}{9} \end{aligned} \tag{3.4}$$

Tanımlayıcılar, görüntünün belirli pikselleri için hesaplanır. Farklı yapı ve boyuttaki filtreler hangi satır ve sütun aralığındaki piksellerin işlenmesi gerektiğini belirlemektedir.

Hessian matrisinin dört girişi hesaplandıktan sonra, her bir ölçek-uzay konumu için normalize işlemi uygulanır. Determinant yanıt haritasının bu ölçekle normalizasyonu, bu girdileri filtre alanına bölerek elde edilir.

İlgi noktası yerelleştirme işlemi algılama işinin son kısmını oluşturmaktadır. Bu görev üç kısımdan oluşmaktadır (Bouris vd. 2010):

- İlk olarak determinantlar daha önceden belirlenmiş bir eşik değeri ile karşılaştırılır ve eşik değerinin altındaki değerler için yerelleştirme işlemi gerçekleştirilmez. Eşik değeri arttıkça tespit edilen ilgi noktalarının sayısı azalırken, daha düşük bir eşik değerinin seçilmesiyle daha fazla sayıda ilgi noktasının oluşması sağlanır.
- İkinci olarak, daha uygun bir aday ilgi noktası kümesini tespit etmek için En Büyük Olmayanı Bastırma (Non-Maximal Suppression) yöntemi gerçekleştirilir. Bu yöntem ile kenar bilgilerini doğru ve hassas şekilde belirleyebilmek amacıyla her piksel için hesaplanan kenar olma eğilim değerleri, hesaplanan yöndeki komşuları ile karşılaştırılır. Komşu piksel değerlerinden büyük olmayan piksellerin eğilim değerleri sıfır yapılır. Yani, ölçek uzayındaki her piksel yerel ölçekteki 8 nokta ile üstündeki ve altındaki ölçeklerin her birinde 9 nokta içeren 26 (8+9+9) komşusuyla karşılaştırılarak işlemler yapılır (Şekil 3.1).



Şekil 3.1. Aday ilgi noktalarının tespiti için kullanılan En Büyük Olmayanı Bastırma Yöntemi (Non-Maximal Suppression) (Evans 2009)

- İlgi noktası yerelleştirme görevinin son aşaması, hem gerçek uzayda hem de ölçekli uzayda ilgi noktalarının konumunu alt-piksel doğruluğuyla bulmak için yapılan interpolasyon işlemidir. Bu işlem Brown'un önerdiği gibi (Brown vd. 2002) bir 3D kuadratik şema kullanılarak yapılır. Sonuç olarak, Taylor açılımı kullanılarak Hessian fonksiyonunun determinantı $H(x, y, \sigma)$ olarak ifade edilir (Bouris vd. 2010).

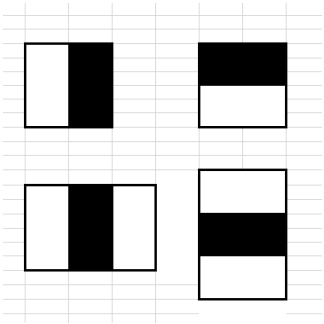
$$H(x) = H + \frac{\theta H^T}{\theta_x} x + \frac{1}{2} x^T \frac{\theta^2 H}{\theta_x{}^2} x \tag{3.5}$$

$$x = -\frac{\theta^2 H^{-1}}{\theta_x{}^2} \frac{\theta H}{\theta_x} \tag{3.6}$$

En Büyük Olmayanı Bastırma yöntemi ile ortaya çıkarılan aday ilgi noktası , $x(x, y, \sigma)$, başlangıçta bu fonksiyonun türevini alarak ve bunu sıfıra ayarlayarak hesaplanır (3.6). Türevler, aday ilgi noktasının komşu determinant değerlerinin sonlu farkları ile yaklaşık olarak hesaplanır. (3.6) numaralı denklem üç değerli bir vektördür. Bu üç değer ile ilgi noktasının tam olarak konumu belirlenir. Yani x, y ve σ (ölçek) değeri. Her üç değerin de 0.5'den az olması durumunda, x ve y örnekleme aralığı ile çarpılır, çarpma sonuçları sırasıyla x ve y ye eklenir, sonuçlar en yakın tam sayıya yuvarlanır (Bouris vd. 2010). Yaklaşık ölçek değeri (3.4)'e dayanılarak hesaplanır.

SURF tanımlayıcısı, piksel yoğunluğunun, algılanan her bir ilgi noktasından oluşan ölçeğe bağlı bir komşulukta nasıl dağıldığını rapor eder. Tanımlayıcının çıkarılması iki ayrı alt göreve ayrılmıştır. Öncelikle, her ilgi noktasına bir yön verilir ve buna dayalı olarak ölçeğe bağlı bir pencere oluşturulmaktadır. Algoritma ile ölçeklendirilmiş bu pencereden 64 boyutlu bir vektör oluşturulur. Görüntünün dönme durumuna karşı etkilenme olmaması için her tespit edilen ilgi noktasına tekrarlanabilir bir yön verilir. Tüm tanımlayıcı bileşenlerin çıkartılması bu yön bilgisi ile

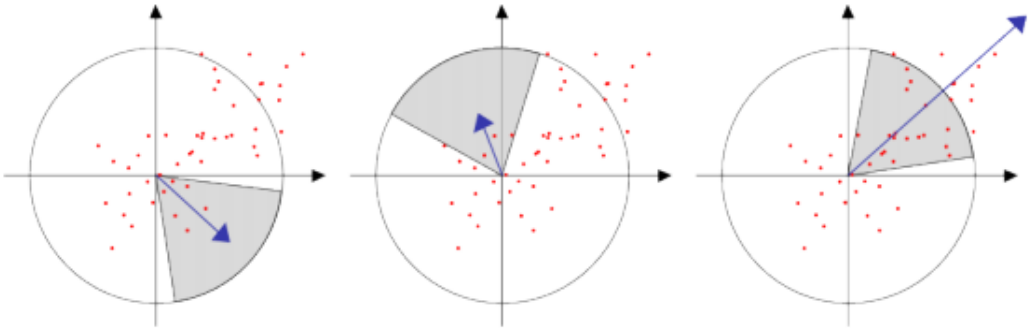
gerçekleştirildiği için bu yönün değişen koşullar altında tekrarlanabilir olması önemlidir (Bouris vd. 2010). Yönü belirlemek için Haar Dalgacıkları kullanılır (Şekil 3.2).



Şekil 3.2. Baskın yönelimi tespit etmek için kullanılan Haar dalgacıkları

Haar Dalgacıkları, sonuçların sağlamlığını arttırmak ve hesaplama süresini azaltmak için integral görüntülerle birlikte kullanılan filtrelerdir. Şekil 3.2'de gösterilen bu basit filtreler, x ve y yönlerinde gradyan değerlerini bulmak için kullanılır. Sol taraftaki yanıtları x yönünde, sağ taraftaki y yönünde hesaplar. Ağırlık değerleri siyah bölgeler için 1, beyaz bölgeler için -1'dir.

4 σ boyutundaki tüm Haar dalgacık tepkileri ilgi noktasından 6 σ'lık bir yarıçap içinde bulunan bir dizi piksel için hesaplanır. Bu piksellerin her biri için hesaplanan tepkiler, bir vektörün yönlendirme açısını oluşturur. Daha sonra bir sınıflandırma işlemi (Şekil 3.3'de gösterilen şekilde kayan bir pencere kullanılarak) bu açıların tümüne uygulanır ve böylece ilgi noktasının baskın yönlendirmesi oluşturulur.



Şekil 3.3. Baskın yönlendirmenin bulunması (Evans 2009)

3.1.2. İntegral görüntü

SURF algoritmasının performanslı olma sebepleri arasında integral görüntü kullanması gösterilebilir. İntegral görüntü bir görüntü için hızlı bir şekilde hesaplanır ve

görüntü üzerindeki dikdörtgen bir alanın hesaplanmasını hızlandırmak için kullanılır. Örneğin, I görüntüsü üzerinde (x , y) noktası verilsin. I_{Σ} integral görüntüsü nokta ile orijin arasındaki değerlerin toplamına eşittir (3.7). İntegral görüntü hesaplandıktan sonra görüntü üzerindeki herhangi bir dikdörtgen bölgenin alanını hesaplama görevi üç işleme azaltılmış oluyor (3.8) (Şekil 2.2). Burada önemli olan nokta, dikdörtgen bölgenin alanı ne kadar büyük olursa olsun yine üç işlem ile bölgenin alanı hesaplanabilmektedir. Bundan dolayı hesaplama süresi azalmaktadır.

$$I_{\Sigma}(x,y) = \sum_{i=0}^{i\le x}\sum_{j=0}^{j\le y} I(x,y) \tag{3.7}$$

$$\Sigma = A + D - (C + B) \tag{3.8}$$

3.1.3. Fast-Hessian dedektörü

3.1.3.1. Hessian matrisi

SURF dedektörü Hessian matrisinin determinantına dayanmaktadır. Konum ve ölçeği tespit etmek için farklı bir ölçüm yapmak yerine Hessian matrisinin determinantı kullanılır. Hessian matrisinin kullanımını iyileştirmek için iki değerin sürekli bir fonksiyonu dikkate alınır. Şöyle ki, (x , y) noktasındaki fonksiyonun değeri f(x , y) ile verilir (Evans 2009). Hessian matrisi, H, f fonksiyonunun kısmi türevlerinin matrisidir (3.1). Diskriminant olarak bilinen bu matrisin determinantı (3.2)'deki gibi hesaplanmaktadır.

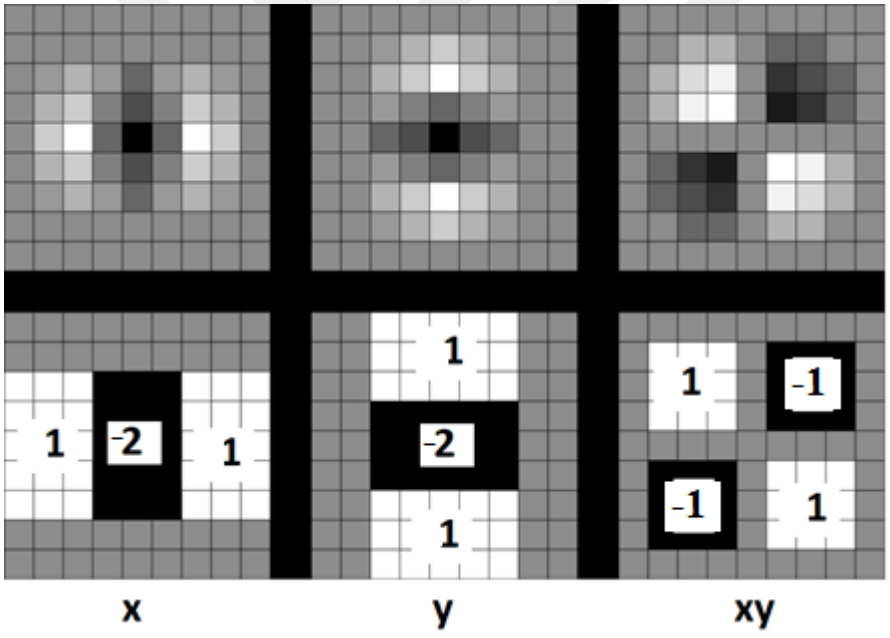
Diskriminantın değeri, ikinci mertebeden türev testi ile fonksiyonun maksimum ve minimum değerlerini sınıflandırmak için kullanılır. Determinant, Hessian matrisinin özdeğerlerinin çarpımı olduğu için sonuçların işaretine dayanarak noktalar sınıflandırılabilir. Determinant negatif ise, özdeğerlerin işaretleri farklıdır ve bu nedenle ilgili nokta yerel bir ekstremum değildir. Eğer pozitif ise, ya her iki özdeğer pozitif ya da ikisi de negatif demektir ve her iki durumda da ilgili nokta bir ekstremum olarak sınıflandırılır (Evans 2009).

İlgili yöntem görüntü üzerinde çalıştırabilmek için f(x , y) fonksiyon değerleri, I(x , y) görüntü piksel yoğunlukları ile değiştirilir. Ardından, görüntünün ikinci mertebeden kısmi türevlerini hesaplamak gerekir. Türevler uygun bir çekirdek konvolüsyonu ile hesaplanabilir. SURF algoritmasında seçilen filtre olarak ikinci dereceden normalize edilen Gauss ölçek-uzay da analiz yapılmasına olanak sağlar. Gauss türevleri için çekirdekler x, y ve bileşik xy yönünde yapılandırılır, böylece Hessian matrisinin dört girişi hesaplanmış olur. Gauss kullanımı ile konvolüsyon aşamasında görüntü üzerinde uygulanacak olan yumuşatma oranı değiştirilebilir ve böylece determinant farklı ölçeklerde hesaplanabilir. Ayrıca, Gauss izotropik (dairesel olarak simetrik) bir fonksiyon olduğundan çekirdek konvolüsyonu rotasyon değişmezliği sağlar. Bu veriler ışığında Hessian matrisi hesaplanabilir.

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \tag{3.9}$$

Burada $L_{xx}(x,\sigma)$ ile ikinci dereceden Gauss türevinin konvolüsyonu kastedilir (**x** = (x,y) noktasındaki). L_{yy} ve L_{xy} de benzer şekilde ifade edilir. Bu türevler LoG (Laplacian of Gaussian) olarak bilinir.

Görüntüdeki her piksel için Hessian matrisinin determinantı hesaplanabilir ve bu değerler ilgi noktalarının bulunmasında kullanılabilir. Bay, ilgili çekirdeklerin kutu filtreler kullanılarak hesaplandığı bir yaklaşımı önermiştir (Bay vd. 2006). Şekil 3.4'de ayrık ve kırpılmış çekirdekler ile onların kutu filtre muadilleri arasındaki benzerliği göstermektedir (Evans 2009). Bu filtreler integral görüntü ile birlikte kullanıldıklarında önemli performans artışları sağlamaktadır.



Şekil 3.4. Determinant tepki haritasını hesaplamak için kullanılan ikinci mertebeden gauss türevi ve kutu filtre muadilleri (9x9 filtre) (Evans 2009)

Performans artışını değerlendirmek için konvolüsyon işlemi sırasında gereken dizi erişimi ve işlem sayısı referans alınabilir. Örnek olarak boyutu 9x9 olan bir filtre için, 81 elemana erişim ve işlem gerekirken kutu filtre kullanıldığında bu sayı sadece 8 olmaktadır. Ayrıca kutu filtre boyuttan bağımsız olduğu için filtre boyutunun artması ek bir işlem yükü getirmez. Sonuç olarak, D_{xx} için 2, D_{yy} için 2 ve D_{xy} için 4 olmak üzere toplam 8 hesaplama yapılır.

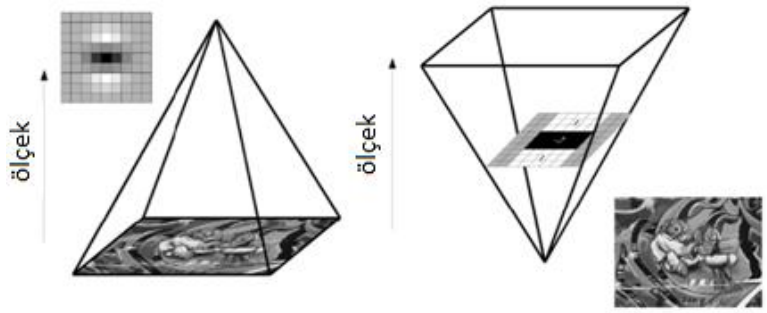
D_{xy} filtresi için siyah bölgeler 1 değeriyle, beyaz bölgeler -1 değeriyle filtrelenir ve geri kalan bölgeler hiç işleme alınmaz. D_{xx} ve D_{yy} filtreleri de benzer şekilde filtrelenir, ancak beyaz bölgelerin değeri -1, siyah bölgelerin değeri 2 olarak kullanılır. Bu basit katsayı kullanımı alanların hızlı bir şekilde hesaplanmasına olanak sağlar. Şekil 3.4'de üst satırda ayrık ve kırpılmış Gauss türevleri (sırasıyla x, y, xy yönlerinde ve L_{xx} , L_{yy}, L_{xy} ile ifade edilir) gösterilmiştir ve alt satırda ise kutu filtre yaklaşımları (sırasıyla x, y, xy yönlerinde ve D_{xx}, D_{yy}, D_{xy} ile ifade edilir) gösterilmiştir. Buna göre Bay tarafından tahmini Gausslar kullanılarak Hessian matrisinin determinantı (3.10) numaralı denklemdeki gibi ifade edilmiştir (Bay vd. 2006).

$$\det(H_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{3.10}$$

Buradaki determinant $\mathbf{x}=(x, y, \sigma)$ konumundaki görüntü bölgesi olarak ifade edilmektedir (Bay vd. 2006). w değeri Gausslar arasındaki enerji dönüşümü ile hesaplanmaktadır ve bu değer Bay tarafından 0.9 olarak önerilmektedir.

3.1.3.2. Ölçek-uzay yapısı

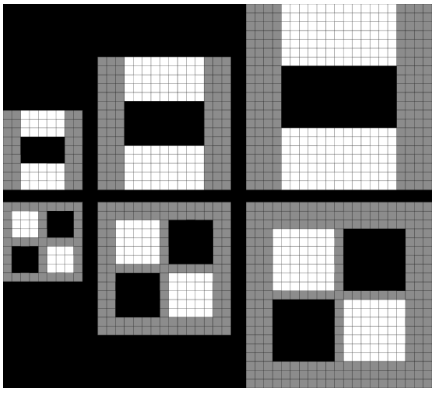
Hessian matrisinin determinantı kullanılarak ilgi noktalarının saptanması için bir ölçek-uzay kullanılmalıdır. Ölçek-uzay, olası tüm ölçekler arasında maksimum ve minimum noktaları bulmak için kullanılabilen sürekli bir işlevdir (Witkin 1983). Ölçek-uzayı tipik olarak, görüntü piramidi olarak uygulanmaktadır. Burada, görüntünün ölçeği değiştirilerek birçok sefer benzer işlemlere tabi tutulması söz konusudur. SURF algoritmasında bu durum farklılık göstermektedir. Değişen görüntü ölçeği yerine, değişen boyutta filtreler kullanılarak benzer bir etki oluşturulur. Böylece, ölçek-uzay piramidinin çok katmanlı elemanlarının eş zamanlı olarak işlenmesine, görüntünün alt örneklerinin oluşturulmasına gerek kalınmamasına ve dolayısıyla performans artışına sebep olmaktadır.



Şekil 3.5. Genel olarak kullanılan ölçek-uzay kavramı (solda) ve SURF algoritmasında kullanılan ölçek-uzay kavramı (sağda)'nı temsil eden filtre piramitleri (Evans 2009)

Şekil 3.5'de genel olarak kullanılan ölçek-uzay kavramı (solda) ve SURF algoritmasında kullanılan ölçek-uzay kavramı (sağda) gösterilmiştir. İlk yöntemde görüntü boyutu değişkendir ve Gauss filtresi değişen görüntü üzerinde tekrar tekrar uygulanır. SURF algoritmasında ise görüntü değişikliğine gidilmeden yalnızca filtre boyutu değiştirilerek daha hızlı sonuçlar elde edilir.

Şekil 3.4'de gösterilen boyutu 9x9 olan filtrelerin çıktısı, s=1.2 (ölçek), atılacak olan başlangıç ölçek katmanı olarak kabul edilir (Bay vd. 2006). Filtre boyutları 9x9, 15x15, 21x21, 27x27, vb. şeklinde olmalıdır. Daha büyük ölçeklerde ardışık filtre boyutları arasındaki adım da bu şekilde olmalıdır (artış miktarı 6 olmalı). Bir merkez pikselin varlığına ihtiyaç duyulduğu için, filtre boyutları bu konum etrafında eşit olarak artırılmalıdır ve bu nedenle lop boyutunun artışı minimum iki piksel olmalıdır. Her bir filtrede aynı boyutta olması gereken üç lop bulunduğundan, ardışık filtreler arasındaki en küçük artış miktarı altı olur. Filtre oranları, ölçeklendirme sonrasında sabit kaldığı için, yaklaşık Gauss türevleri buna göre ölçeklendirilir. Örneğin, filtre boyutu 27x27 için s=3x1.2 (3.6) olarak hesaplanır. (3.3) ve (3.4) numaralı denklemler ile karşılık gelen yaklaşık ölçek değeri hesaplanır. Şekil 3.6'da filtre boyutlarının değişimi gösterilmiştir.



Şekil 3.6. Filtre boyutlarının değişimi (Evans 2009)

3.1.3.3. İlgi noktası yerelleştirmesi

İlgi noktalarının yerelleştirilmesi görevi üç aşamaya ayrılır. İlk olarak, hesaplanan değerler daha önceden belirlenen bir eşik değer ile karşılaştırılır ve eşiğin altında kalan değerler işleme alınmaz. Eşiğin artırılması, algılanan ilgi noktalarının sayısını düşürürken, sadece en güçlü noktaların kalmasını sağlar. Eşiğin gereğinden fazla azaltılması durumunda ise çok fazla ilgi noktasının işlenmesi durumu ortaya çıkabilir.

Eşik işleminden sonra, aday ilgi nokta setini bulmak için En Büyük Olmayanı Bastırma (non-maximal suppression) yöntemi gerçekleştirilir. Bunu yapmak için, ölçek alanındaki her piksel yerel ölçekteki 8 nokta, üst ve alt ölçeklerin her birinde 9 nokta olmak üzere toplam 26 komşu nokta ile karşılaştırılır. Şekil 3.1'de bu durum detaylı bir şekilde gösterilmiştir. Bu aşamada, belirlenen eşik değeriyle ve aynı zamanda ölçek-uzayındaki yerel maksimum-minimum değerine göre bir dizi ilgi noktası bulunur.

Noktaları yerelleştirmenin son aşaması, ölçek ve uzayın alt piksel doğruluğu ile yerini bulmak için yakındaki verilerin interpolasyonunu içerir. Bu, Brown (Brown vd. 2002) tarafından önerilen 3D kuadratik bir yöntemle yapılır. Bunun için Hessian fonksiyonunun determinantı, $H(x, y, \sigma)$, tespit edilen konumda merkezli ikinci dereceden terimlere kadar bir Taylor genişlemesi olarak ifade edilir (Evans 2009). (3.11) numaralı denklemde ifade edilmiştir.

$$H(x) = H + \frac{\partial H^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 H}{\partial x^2} x \tag{3.11}$$

Ekstremumun interpolasyonlu konumu, $\hat{x} = (x, y, \sigma)$, bu fonksiyonun türevi alınıp sıfıra eşitlenerek bulunur (3.12).

$$\hat{x} = -\frac{\partial^2 H^{-1}}{\partial x^2} \frac{\partial H}{\partial x} \tag{3.12}$$

Buradaki türevler komşu piksellerin sonlu farkları ile yaklaştırılır. Eğer $\hat{x}$; x, y veya σ yönlerinde 0.5'den daha büyük ise konum ayarlanır ve interpolasyon işlemi tekrarlanır. Bu işlemler $\hat{x}$ değeri tüm yönlerde 0.5'den daha küçük olana kadar veya daha önceden belirlenen interpolasyon adım sayısına ulaşılana kadar devam eder. Sadece istikrarlı ve tekrarlanabilir noktalar seçilerek, yakınsanamayan noktalar ilgi noktası kümesinden çıkarılır.

3.1.4. İlgi noktası tanımlayıcısı

SURF tanımlayıcısı, Fast-Hessian dedektörü tarafından tespit edilen her ilgi noktasının ölçeğe bağlı bir komşusundaki piksel yoğunluğunun nasıl dağıldığını açıklar. Bu yaklaşım SIFT (Lowe 2004) algoritmasına benzer, fakat farklı olarak Haar dalgacıkları olarak bilinen filtrelerle birlikte kullanılan integral görüntüler sağlamlığı arttırmak ve hesaplama süresini azaltmak için kullanılırlar. Haar dalgacıkları, x ve y yönlerinde gradyanları (gradients) bulmak için kullanılabilen basit filtrelerdir. Şekil 3.2'de örnek Haar dalgacık modelleri gösterilmiştir. Burada, soldaki filtre ile x yönünde, sağdaki ile y yönünde yanıtlar hesaplanır. Siyah bölgeler için ağırlıklar 1, beyaz bölgeler için -1'dir. İntegral görüntülerle birlikte kullanıldıklarında her hesaplama için sadece altı işlem gereklidir.

Tanımlayıcı çıkarma işlemi iki ana göreve ayrılır. İlk olarak her ilgi noktasına tekrarlanabilir bir yön verilmektedir. Daha sonra 64 boyutlu bir vektörün çıkarıldığı ölçeğe bağlı bir pencere oluşturulur. Tanımlayıcı için yapılan tüm hesaplamaların, ölçekten etkilenmemesi için tespit edilen ölçeğe göre yapılması gereklidir.

3.1.4.1. Yön ataması

Görüntünün döndüğü durumlarda ilgi noktalarının dönmeden etkilenmemesi için tespit edilen her ilgi noktasına tekrarlanabilir bir yön verilmektedir. Yönü belirleyebilmek için, 4σ boyutundaki Haar dalgacık tepkileri, algılanan noktanın 6σ 'lık bir yarıçap içerisindeki belirlenmiş pikselleri için hesaplanır. Burada σ, noktanın tespit edildiği ölçeği belirtir. Yanıtlar, ilgi noktasında merkezlenmiş bir Gauss ile ağırlıklandırılır. Kalana bağlı olarak Gauss, noktanın ölçeğine bağlıdır ve standart sapma 2.5σ olarak seçilmiştir. Yanıtlar ağırlıklandırıldıktan sonra, vektör uzayındaki noktalar olarak temsil edilir (x yanıtları apsis boyunca, y yanıtları ordinat boyunca). Baskın yönlendirme, orijin etrafında pi/3 açısını kapsayan bir daire parçası döndürülerek seçilir. Her pozisyonda, parça içerisindeki x ve y cevapları toplanır ve yeni bir vektör oluşturmak için kullanılır. En büyük vektör ilgi noktasının yönelimini belirtir. Şekil 3.3'de gösterilmiştir.

Bazı uygulamalarda dönme değişikliği söz konusu değildir. Bu nedenle bu aşama uygulanmayabilir ve daha fazla performans sağlanabilir. Tanımlayıcının bu versiyonu Upright SURF (U-SURF) olarak bahsedilir ve ± 15 dereceye kadar görüntü rotasyonları için dayanıklılığı koruması sağlanmıştır (Bay vd. 2006).

3.1.4.2. Tanımlayıcı bileşenleri

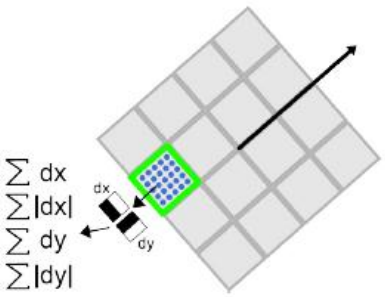
SURF tanımlayıcısını oluşturmak için ilk olarak ilgi noktasının etrafında kare bir pencere oluşturulur. Bu pencere, tanımlayıcı vektöre girdi oluşturacak ve boyutu 20σ olan pikselleri içerir. Buradaki σ, tespit edilen ölçeği ifade etmektedir. Ayrıca,

oluşturulan pencere bölüm 3.4.1'de bulunan yön boyunca yönlendirilir ve hesaplamalar bu yönde yapılır.

Tanımlayıcı penceresi 4x4 olacak şekilde alt bölgelere ayrılmıştır. Bu alt bölgelerin her birinde düzenli olarak dağıtılan 25 örnek noktası için, 2σ büyüklüğünde Haar dalgacıkları hesaplanır. x ve y dalgacık yanıtlarını sırasıyla dx ve dy olarak ifade edersek, topladığımız bu 25 örnek noktası için (her alt bölge için) 4 yeni değer elde edilir (3.13).

$$\upsilon_{altbolge} = \left[\sum dx, \sum dy, \sum |dx|, \sum |dy|\right] \tag{3.13}$$

Böylece tanımlayıcı vektör 4x4 alt bölgeden oluşur ve 4x4x4 uzunluğunda bir büyüklüğe sahip olur. Elde edilen SURF tanımlayıcı vektörünün boyutu 64 olur. Bu tanımlayıcı rotasyon, ölçek, parlaklık gibi değişkenlik gösteren durumlardan etkilenmez. Şekil 3.7'de tanımlayıcı vektörün yapısı gösterilmiştir (Evans 2009). Burada, yeşil kare 16 alt bölgeden birini sınırlar, mavi daireler dalgacık tepkilerini hesapladığımız örnek noktaları temsil eder.



Şekil 3.7. İlgi noktasının tanımlayıcı bileşenler ile ifade edilmesi (Evans 2009)

3.2. Alanda Programlanabilir Kapı Dizileri (FPGA)

3.2.1. Giriş

Alanda Programlanabilen Kapı Dizileri (FPGA), herhangi bir sayısal devreyi veya sistemi oluşturmak için elektriksel olarak programlanabilen ve çok sayıda mantık hücresinden meydana gelen silikon teknolojilerdir. FPGA'lar, sabit fonksiyonlu Uygulamaya Özgü Tümleşik Devre (ASIC) teknolojisinden üstün olarak birçok avantaj sağlamaktadır. ASIC'ler, tipik olarak istenilen bir devreyi elde etmek için uzunca sürebilen üretim aşamaları ve yüksek maliyetlere sahiptir. FPGA'ler ise daha kısa

sürede oluşturma ve herhangi bir hata durumunda yeniden programlanabilirlik avantajına sahiptirler (Yıldırım vd. 2012).

Tasarım sırasında büyük esneklik sağlaması ve paralel işlem yapabilme kabiliyeti sebebiyle FPGA kullanımı günümüzde oldukça yaygınlaşmıştır. FPGA büyük ölçekli paralel işlem özelliğinden dolayı görüntü işleme gibi uygulamaların gerçekleştirilmesine oldukça uygundur.

3.2.2. FPGA mimarisi

FPGA'ler birbirini tekrarlayan küçük lojik bloklardan oluşan sahada programlanabilen lojik (FPL) cihazlar ailesinin bir üyesidir. FPGA, programlanabilir mantık blokları ve bu bloklar arasındaki programlanabilir ara bağlantılardan oluşan ve geniş uygulama alanlarına sahip olan sayısal tümleşik devrelerdir. İhtiyaç duyulan mantık fonksiyonlarını gerçekleştirme amacına yönelik olarak üretilmiştir. Bunun için her bir mantık bloğunun fonksiyonu kullanıcı tarafından düzenlenebilmektedir. FPGA ile temel mantık kapılarının ve yapısı daha karmaşık olan devre elemanlarının işlevselliği artırılmaktadır. Alanda programlanabilir ismi verilmesinin nedeni, mantık bloklarının ve ara bağlantıların imalat sürecinden sonra programlanabilmesidir (Güneren 2010).

Bir FPGA'i oluşturan temel yapılar, biçimlendirilebilir lojik bloklar (Configurable Logic Blocks, CLB), giriş/çıkış blokları (Input/Output Blocks, IOB) ve ara bağlantılardır (Güneren 2010).

Biçimlendirilebilir lojik bloklar genel olarak doğruluk tabloları (Look-up Table, LUT) ve flip-flop'lardan oluşur. CLB'ler ara bağlantılarla birbirlerine bağlanabilir ve karmaşık fonksiyonlar gerçeklenebilir. IOB'ler FPGA içindeki sinyallerin dış ortamla olan bağlantısını sağlar. Ara bağlantılar ise uygun şekilde programlanarak CLB'ler ve IOB'leri birbirine bağlar.

FPGA'leri programlamak için en çok kullanılan donanım tasarlama dilleri Verilog ve VHDL'dir. Ayrıca bu tezde de kullanılan HLS aracı ile C, C++ dilleri kullanılarak fpga için kod yazılabilmekte ve direktifler ile sentezlenebilmektedir.

3.2.3. FPGA'in programlanması

FPGA programlamada ilk adım verilog, VHDL gibi donanım tanımlama dilleri ile donanımın davranışını tanımlamaktır. İkinci olarak devrenin bağlantı listesi oluşturulur. Burada, oluşturulan donanım bilgisayar ile simüle edilerek doğrulama işlemi gerçekleştirilebilir. Oluşan hatalar veya yanlış tasarlanan donanım yapıları varsa değiştirilir. Daha sonra lojik sentezleme adımına geçilir. Burada devre fonksiyonları FPGA içerisindeki CLB'ler ile eşleştirilir ve kapı seviyesinde bir bağlantı listesi oluşturulur. Daha sonra devre fonksiyonları ve eşleştirilmiş CLB'ler, FPGA içerisinde uygun bölgelere yerleştirilir ve aralarındaki bağlantılar oluşturulur. Oluşturulan bu yapı gerçeğe çok yakındır ve benzetim yoluyla sınandıktan sonra FPGA'i programlamak için

gerekli kod oluşturulur. Son aşamada ise oluşturulan kodlar ile FPGA programlanır (Güneren 2010).

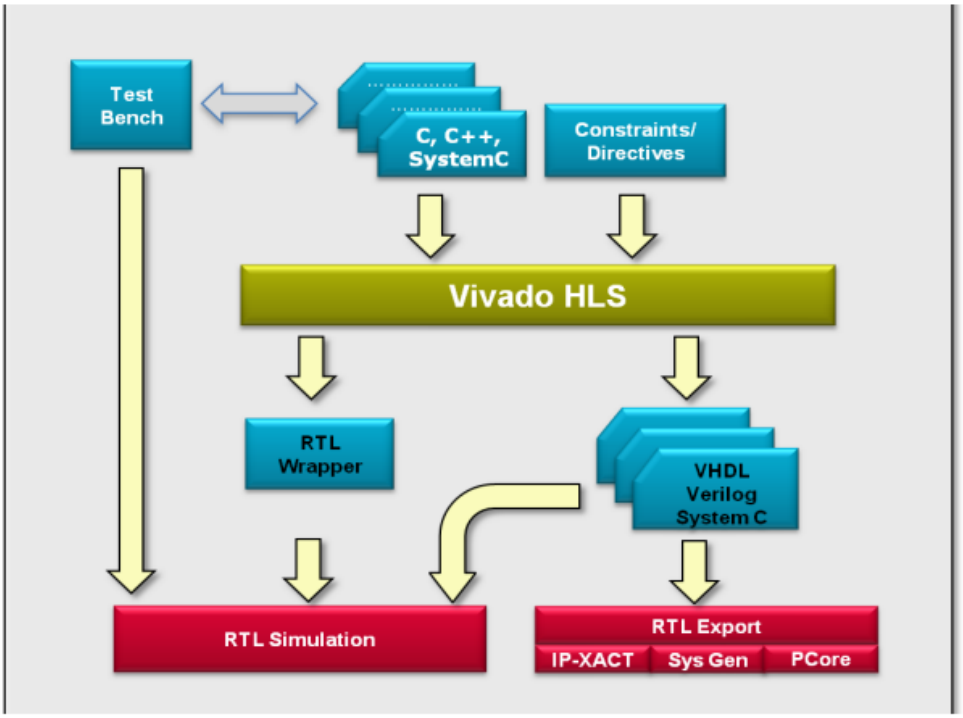
3.3. Vivado High Level Synthesis (HLS)

3.3.1. Giriş

Xilinx'in geliştirdiği Vivado HLS ile C, C++ veya SystemC ile oluşturulan kodlar RTL kaynak kodlarına veya IP (Intellectual Property) Core olarak adlandırılan paketlere dönüştürülebilir. Şekil 3.8'de HLS ile tasarım süreçleri gösterilmiştir.

Vivado HLS ile iki farklı sentezleme desteklenmektedir:

- Algoritma sentezi fonksiyonların içeriğini alır ve fonksiyonel ifadeleri RTL ifadelerine sentezler. Bu sentezleme algoritmayı gerçekleştirir fakat arabirim sentezinden önemli ölçüde etkilenir.
- Arabirim sentezi, tasarımın sistemdeki diğer tasarımlarla iletişim kurmasına izin veren belirli zamanlama protokolleri ile fonksiyon argümanlarını (veya parametreleri) RTL bağlantı noktalarına dönüştürür.



Şekil 3.8. Vivado HLS kullanarak oluşturulan FPGA tasarımı için akış örneği (Xilinx 2013)

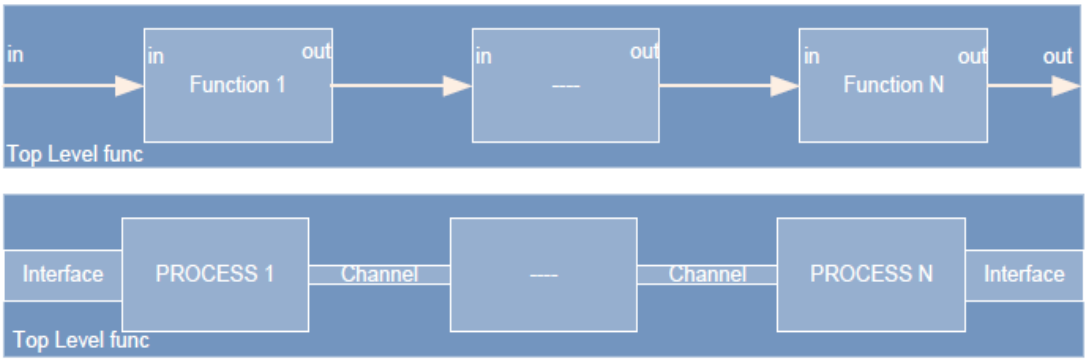
Akış diyagramına göre uygulamanın ilk adımı, desteklenen bir proglama dili (C, C++, SystemC) ile algoritmanın oluşturulmasıdır. Oluşturulan kod "HLS Tools" a eklenir. Ayrıca, performansı arttırmak veya kullanılan donanım miktarını azaltmak için sistemde bulunan direktifler kullanılabilir.

3.3.2. Vivado HLS'nin optimize edilmesi

Optimize edilmiş RTL kodunu üretmek için HLS'de bulunan direktifler kullanılır (Xilinx 2013). Bu direktifler doğrudan kod içerisine yazılabildiği gibi '.tcl' dosyalarında da tanımlanarak kullanılabilir.

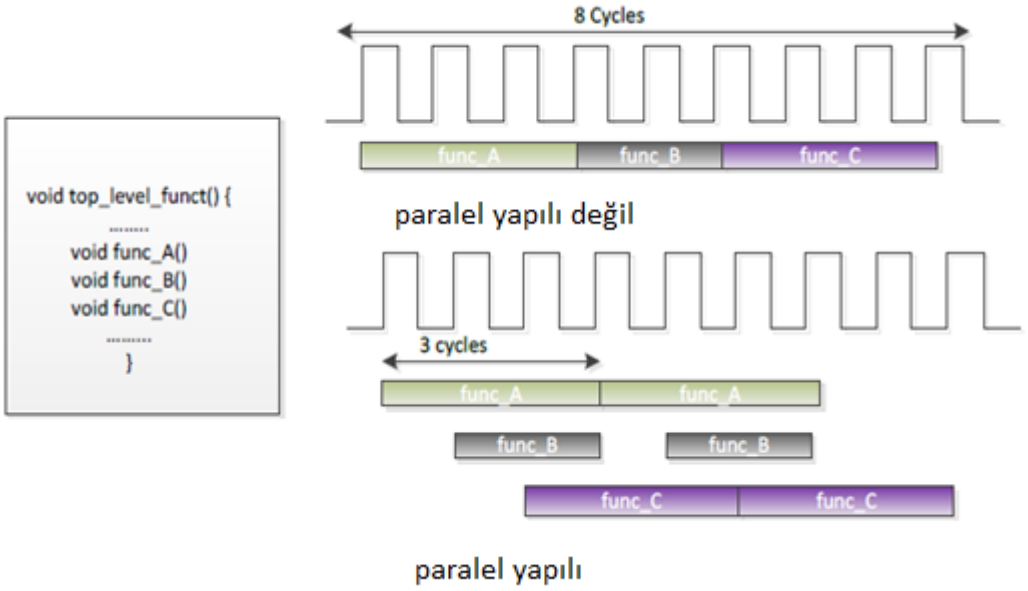
3.3.2.1. HLS'de boru hatları

Kullanılmak istenen boru hattı, istenen fonksiyonlar arasında bir optimizasyon olarak veya bir fonksiyonun içerisindeki işlemlere uygulanabilir. Ayrıca bir döngü içerisindeki veya döngüler arasındaki işlemlere de uygulanabilir. Boru hattı kullanılarak fonksiyonları veya döngüleri sıralı olarak çalıştırmak yerine, bir önceki fonksiyon ya da döngüdeki tüm işlemlerin bitmesini beklemeden sıradaki işlemleri başlatarak sonuçların daha hızlı oluşması sağlanabilir. Boru hattı fonksiyonlar arasına uygulanırsa veri akışlı boru hattı olarak adlandırılır. Aşağıdaki şekilde sıralı olarak çalışan bir fonksiyonlar zincirinin paralel olarak nasıl çalıştırıldığı gösterilmiştir.



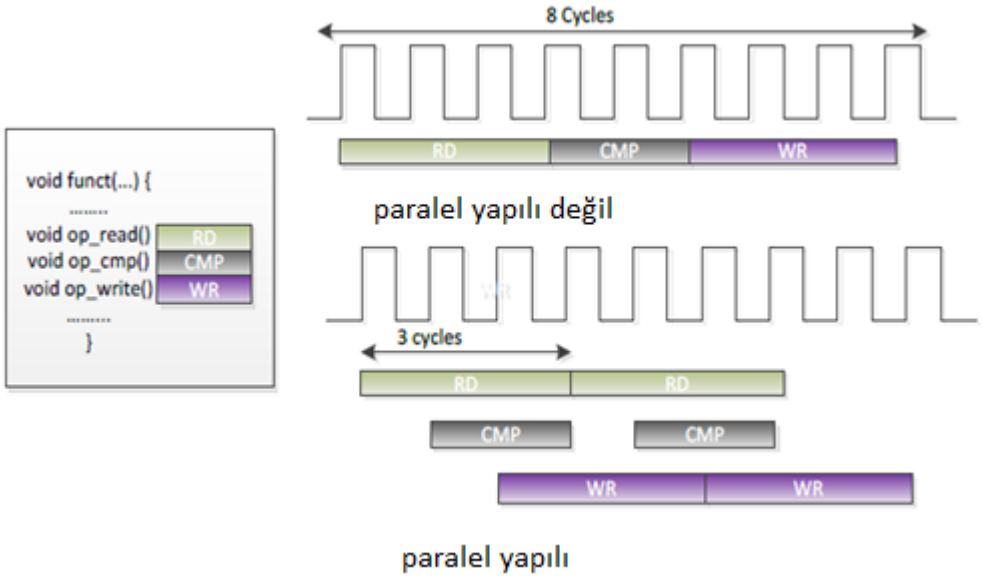
Şekil 3.9. Sıralı ve paralel çalışma yapısı (Baguma 2014)

Şekil 3.9'da bahsedilen yapının iç yapısı Şekil 3.10'da daha detaylı olarak gösterilmiştir.



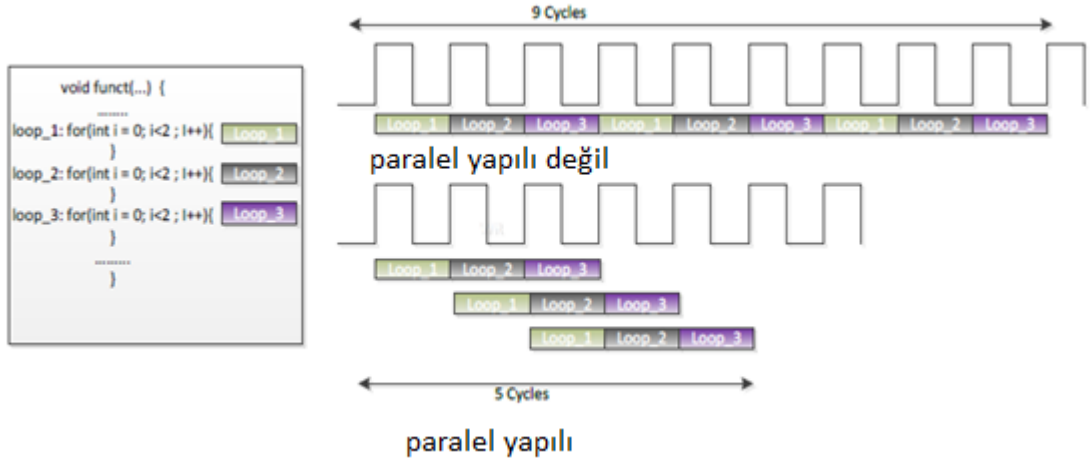
Şekil 3.10. Paralel veri akış örneğinin yapısı (Baguma 2014)

Şekil 3.11'de gösterildiği gibi fonksiyonlar kaynak çakışması ve veri bağımlılığı olmayacak şekilde eş zamanlı olarak yürütülerek sonuçların daha hızlı oluşması sağlanabilir.



Şekil 3.11. Fonksiyonlardaki boru hattı yapısı (Baguma 2014)

Şekil 3.12'de ise döngülerin eş zamanlı olarak nasıl çalıştığı bir örnek ile ifade edilmiştir.



Şekil 3.12. Döngülerdeki boru hattı yapısı (Baguma 2014)

3.3.2.2. HLS'de dizilerin tanımlanması

Diziler HLS'de varsayılan olarak RAM'lere kaydedilirler. Diziler, direktifler kullanılarak yapılandırılabilirler.

Çizelge 3.1. Dizi tanımlama direktifleri (Baguma 2014)

Direktif	Açıklama
Resource	Bir dizinin hangi donanıma (RAM bileşeni) kaydedileceğini belirtir.
Array_Map	RAM kaynaklarını optimize etmek için ve kullanılan alanı azaltmak için birden çok küçük diziyi tek bir büyük diziye birleştirerek dizi boyutlarını yeniden yapılandırır.
Array_Partition	RAM erişimlerinde oluşabilecek darboğazı azaltmak için büyük dizileri daha küçük dizilere parçalar. Ayrıca saklayıcılar olarak da saklanmasını sağlar.
Array_Reshape	Çok RAM kullanmadan RAM erişimlerini iyileştirmek için kullanılır.
Stream	Bir dizinin RAM yerine FIFO olarak tanımlanmasını sağlar.

Şekil 3.13 ve Şekil 3.14'de "Array_Map" direktifinin kullanımı bir örnek ile gösterilmiştir. "array1" ve "array2" dizileri direktif kullanılarak hafızada tek parça

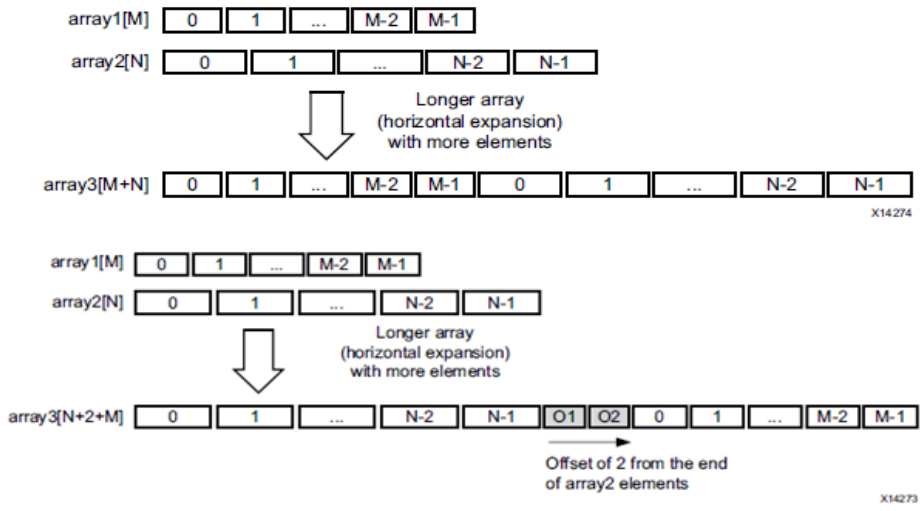
olacak şekilde “array3” halinde depolanmıştır. Böylece küçük dizilerin daha büyük tek bir dizide toplanması amaçlanmıştır.

```
void foo (...) {
  int8  array1[M];
  int12 array2[N];
  ...
loop_1: for(i=0;i<M;i++) {
    array1[i] = ...;
    array2[i] = ...;
    ...
}
...
}
```

++++++++++++++++++++++

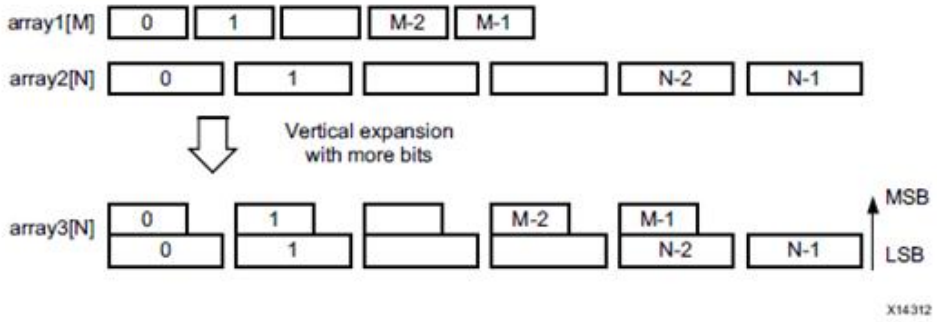
```
void foo (...) {
  int8  array1[M];
  int12 array2[N];
#pragma HLS ARRAY_MAP variable=array1 instance=array3 horizontal
#pragma HLS ARRAY_MAP variable=array2 instance=array3 horizontal
  ...
loop_1: for(i=0;i<M;i++) {
    array1[i] = ...;
    array2[i] = ...;
    ...
}
...
}
```

Şekil 3.13. Yatay dizi tanımlama örnek kodları (Xilinx 2013)



Şekil 3.14. Yatay dizi tanımlama gösterimi (Xilinx 2013)

```
void foo (...) {
  int8  array1[M];
  int12 array2[N];
#pragma HLS ARRAY_MAP variable=array2 instance=array3 vertical
#pragma HLS ARRAY_MAP variable=array1 instance=array3 vertical
  ...
loop_1: for(i=0;i<M;i++) {
    array1[i] = ...;
    array2[i] = ...;
    ...
}
...
}
```



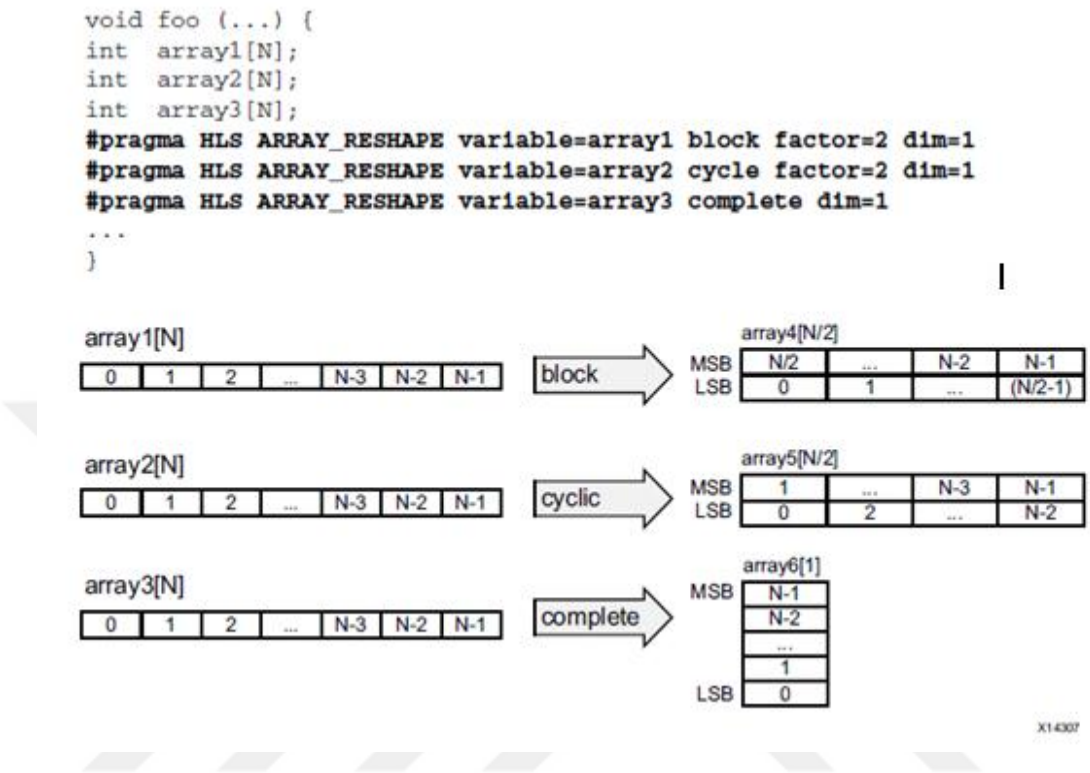
Şekil 3.15. Dikey dizi tanımlama örneği (Xilinx 2013)

Şekil 3.15'de "array1" ve "array2" dizilerinin hafıza da dikey olarak tanımlanması bir örnek ile gösterilmiştir. "Array_Partition" direktifi ile daha önceden parçalanmış bir diziyi "Array_Map" ile birleştirmek için, direktifin aşağıda belirtildiği gibi tüm parçalara uygulanması gerekir.

```
#pragma HLS array_partition variable=m_accum cyclic factor=2 dim=1
#pragma HLS array_partition variable=v_accum cyclic factor=2 dim=1
#pragma HLS array_map variable=m_accum[0] instance=_accum horizontal
#pragma HLS array_map variable=v_accum[0] instance=mv_accum horizontal
#pragma HLS array_map variable=m_accum[1] instance=mv_accum_1 horizontal
#pragma HLS array_map variable=v_accum[1] instance=mv_accum_1 horizontal
```

"Array_Reshape" direktifi ile diziler yeniden şekillendirilebilir. Şekil 3.16'da detaylı olarak gösterilmiştir. Örneğe göre, "block" modu seçildiğinde "factor = 2" değeri için, dizi orta noktasından ayrılacak şekilde iki parça olacak şekilde tanımlanır. Yani "factor" değerine bölünerek yeni diziler oluşturulur. "cycle" modu seçildiğinde ise "factor = 2" değerine göre yine 2 yeni dizi oluşturulur. Fakat, dizinin sıfırıncı elemanı birinci diziye, birinci elamanı ikinci diziye, ikinci elemanı birinci diziye kopyalanacak şekilde diğer elemanlarda kopyalanır. Bu şekilde yeni dizi oluşturulur. "complete"

modunda ise dizi elemanları geçici saklayıcılara ayrılır ve daha sonra bunları daha geniş bir kelime içeren bir dizi haline getirir.



Şekil 3.16. Diziyi yeniden tanımlama yöntemleri (Xilinx 2013)

3.3.2.3. HLS'de döngü tanımlamaları

HLS varsayılan olarak tüm döngüleri yuvarlar. Tüm açılmış çevrimsiz FSM'ler tasarımda en az bir durum oluşturur. Birden fazla ardışık döngü olduğunda gereksiz ek döngüler oluşabilir. Ayrıca iç içe geçmiş döngüler için fazladan çevrimler gereklidir.

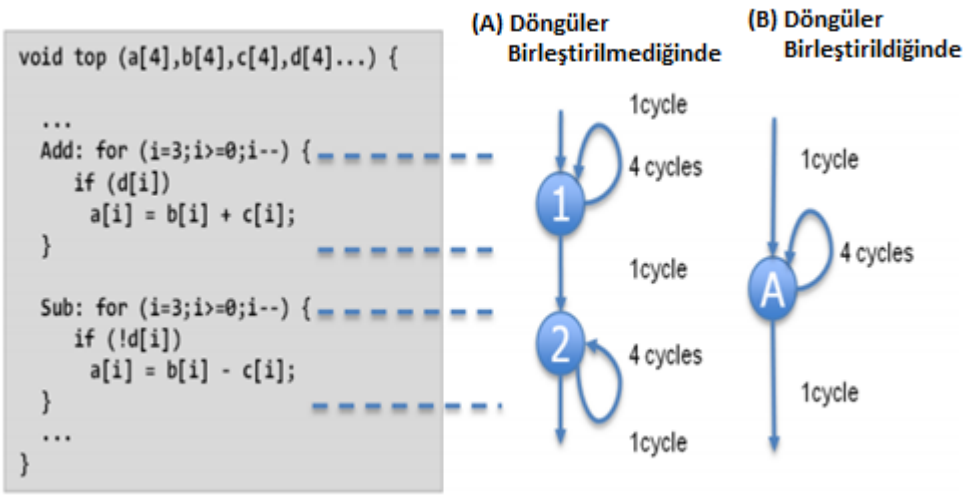
Çizelge 3.2. Döngü optimizasyonu için kullanılan direktifler (Baguma 2014)

Direktif	Açıklama
Unrolling	Tekli işlemlerden ziyade çoklu bağımsız işlemler oluşturmak için döngüleri açmak için kullanılır.
Merging	Toplam gecikmeyi azaltmak, paylaşımı ve optimizasyonu arttırmak için ardışık döngüleri birleştirmek için kullanılır.
Flattening	İç içe olan döngülerin tek bir döngüye

	sığdırılmasını sağlar.
Dataflow	Sıralı döngülerin aynı anda çalışmasına izin verir.
Pipelining	Eş zamanlı işlemler yaparak başlatma aralığını iyileştirmek için kullanılır.
Dependence	Döngüler ile ilgili ek bilgiler sağlamak için kullanılır.
Latency	Döngü işlemi için bir çevrim gecikmesi belirtir.

Yukarıdaki çizelgelerde belirtilen direktifler uygun olarak kullanıldıklarında, daha az donanım kullanımı ve daha fazla hız artışı sağlamaktadır.

Şekil 3.17'de sıralı döngülerin tek bir döngü olarak nasıl birleştirildiği bir örnek ile gösterilmiştir. Bu işlem için "LOOP_MERGE" direktifi kullanılır. Böylece döngü başlarında ve sonlarındaki gereksiz zaman kayıpları azaltılmış olur. Döngüler de sıralı bellek erişimi kullanılıyorsa bu direktif kullanılamaz. Çünkü birleştirme işlemi sıralı bellek okuma işlevini bozacaktır. Ayrıca, birleştirilen döngülerden en yüksek döngü sınırına sahip olan değer yeni döngünün sınırı olarak kabul edilir.



Şekil 3.17. Döngülerin birleştirilmesi (Xilinx 2013)

3.3.2.4. HLS'de kullanılan alanın optimize edilmesi

Kullanılan alan optimize edilerek, FPGA üzerinde bulunan donanımın en ideal şekilde kullanılıp performans alınması amaçlanmaktadır. "ALLOCATE" direktifi kullanılarak kullanılacak donanım miktarı kısıtlanabilir. Bu direktif ile belirli işlevleri veya işlemleri uygulamak için kullanılan RTL yapılarının sayısı tanımlanır veya sınırlanabilir. Örnek olarak C/C++ dosyası, "mult (çarpma)" fonksiyonunun beş örneğine sahipse, "set_directive_allocation" komutu ile tüm yapı tek RTL ile oluşturulabilir. Böylece her bir örnek için ayrı bir RTL yapısı tanımlanmasının önüne geçilir.

Instance	Module	BRAM_18K	DSP48A	FF	LUT
grp_Reg_ap_int_8_s_fu_201	Reg_ap_int_8_s	0	0	9	0
grp_mult_fu_153	mult	0	4	150	144
Total	2	0	4	159	144

Instance	Module	BRAM_18K	DSP48A	FF	LUT
grp_Reg_ap_int_8_s_fu_187	Reg_ap_int_8_s	0	0	9	0
grp_mult_fu_146	mult	0	4	81	144
grp_mult_fu_154	mult	0	4	81	144
grp_mult_fu_162	mult	0	4	81	144
grp_mult_fu_170	mult	0	4	81	144
grp_mult_fu_178	mult	0	4	81	144
Total	6	0	20	414	720

Şekil 3.18. HLS'de alan kullanımının azaltılması (Baguma 2014)

Şekil 3.18'de kaynak kullanımı bir örnek ile gösterilmiştir. Direktif kullanılmadan önce (alttaki) işlemler her biri 4 adet DSP48A'dan oluşan 5 tane yapı ile temsil edilirken, direktif kullanılarak (üstteki) kaynak kullanımı bir yapılı yani 4 adet DSP48A ile oluşturulmuştur. C++ kodu içerisine "**#pragma** HLS ALLOCATION instances=mult limit=1 function" şeklinde yazılarak direktif kullanılabilir. Ayrıca kullanılan FF sayısı ve LUT sayısı da azalarak kaynak tüketimi kısılmıştır.

```
dout_t array_arith (dio_t d[317]) {
  static int acc;
  int i;
#pragma HLS ALLOCATION instances=mul limit=256 operation

  for (i=0;i<317;i++) {
#pragma HLS UNROLL
  acc += acc * d[i];
  }
  rerun acc;
}
```

Yukarıdaki örnek kodda kullanılacak donanımın sınırlandırılması bir örnek ile gösterilmiştir. Örnek incelendiğinde döngü içerisinde 317 tane çarpma işleminin olduğu

görülmektedir. Burada çarpma işlemleri için kullanılacak donanım sayısı 256 olarak sınırlandırılmıştır. Yani sentezleme sırasında en fazla 256 tane çarpma donanımı kullanacak şekilde ayarlanır. Eğer direktif de belirtilen değerden daha az donanım kullanılarak sentezleme gerçekleştirilebiliyorsa HLS otomatik olarak kendi belirlediği donanım sayısı ile işlemlerine devam eder.

3.3.2.5. HLS ile oluşturulan modüllerin Vivado'ya aktarılması

HLS'de oluşturulan modüller test edilip sentezlendikten sonra "ip-core" olarak paketlenerek Vivado ortamına aktarılabilir. Böylece Vivado ortamında oluşturulacak olan FPGA projelerinde kütüphane dosyası olarak kullanılabilir.

HLS'de kullanılan giriş ve çıkışlar "s_axi" arabirimi kullanılarak tanımlandığında, HLS ile oluşturulan modüllerin bir GPU veya işlemci (ZYNQ, vb.) ile kontrol edilmesi sağlanabilir. Böylelikle HLS modülleri ile işlemci arasında oluşturulan veri yolu ile işlemci modülleri kontrol edebilir. Oluşturulan veri yolu sinyalleri işlemci tarafından kontrol edileceği için, işlemciye ait yazılım kütüphaneleri kullanılarak bu kontroller tanımlı olan fonksiyonlar ile gerçekleştirilir.

HLS'de modüller oluşturulduktan sonra ilk adım olarak, sentezlenen modüller "ip-core" olarak oluşturulur. ".zip" uzantılı olarak paketlenmiş şekilde oluşturulan bu dosya, Vivado'ya eklenir. Vivado kütüphanesine eklenen modül ismi ile çağrılarak oluşturulacak projelerde kullanılabilir.

3.4. OPENSURF Algoritmasının Matlab İle Oluşturulması

OPENSURF kütüphanesi SURF algoritmasının bir uygulamasıdır (Evans 2009). Christopher EVANS tarafından C# ve C++ dillerinde oluşturulmuştur. Dirk-Jan Kroon tarafından Matlab'a uyarlanmıştır. Bu tez de Dirk-Jan Kroon tarafından uyarlanan Matlab kodu referans alınmıştır.

İlk olarak kütüphanenin matlab ile analizi yapılmıştır. Gerekli düzenlemeler yapıldıktan sonra HLS için gerekli olacak modüller belirlenerek kullanılmayacak olan fonksiyonlar çıkarılmıştır. Saklayıcıların en uygun bit uzunluklarını belirleyebilmek için matlab üzerinde çeşitli simülasyonlar yapılarak olabilecek en küçük bit uzunlukları belirlenmiştir. Buradaki amaç, minimum kayıp ile verileri olabilecek en küçük hafıza alanında saklayarak performans artışı sağlamak ve hafıza kullanımının azaltmaktır.

Matlab'daki sistem son halini aldıktan sonra modüller halinde belirlenen fonksiyonlar HLS'ye uyarlandı ve gerekli düzenlemeler yapılarak sentezlendi. Sentezleme sonuçlarının optimum olması için HLS'de var olan direktiflerden uygun olanlar kullanıldı. Bu direktifler ile sistemin yapısına karar verilebilir. Örnek olarak fonksiyonların sıralı veya paralel olarak çalışması, saklayıcıların tutulduğu hafıza blokları, haberleşme protokolleri, vs. gibi sistemin tam olarak nasıl olacağına karar verilebilir. Farklı ayarlar için sentezleme sonuçlarının nasıl değiştiği ve matlab sonuçlarına göre çalışma süresinin ne kadar değiştiği gözlemlenmiştir. Ayrıca, saklayıcıları tanımlarken veri tipi olarak kayan-nokta (floating-point) veya sabit nokta

(fixed-point) seçildiği durumda ne gibi değişiklikler olduğu hem matlab hem de HLS sentezleme sonuçlarında analiz edilmiştir.

OPENSURF kütüphanesinin matlab üzerinde çalışan versiyonu referans alınarak analizler yapılmıştır (Kroon 2010). Algoritma temel olarak iki aşamadan oluşmaktadır. İlk olarak referans tanımlayıcı vektörlerin çıkarılacağı referans görüntüsü belirlenmelidir. Bu, görüntü içerisinde belirlenen bir nesne veya görüntü üzerinde herhangi bir ROI olabilir. Belirlenen resim SURF algoritmasının işlemlerinden geçirilerek, belirlenen parametrelere göre tanımlayıcı vektörleri elde edilir. Elde edilen vektörler depolanır ve daha sonra arama işlemlerinde kullanılırlar.

İkinci olarak vektörleri tespit edilen nesnenin aranacağı görüntü için benzer işlemler tekrarlanır. Yine SURF algoritmasının işlemlerinden geçirilir ve tanımlayıcı vektörler elde edilir. Aranan nesnenin görüntü üzerinde hangi bölge içerisinde olabileceği tahmin edilebiliyorsa görüntünün sadece o kısmının işlenmesi hız açısından daha iyi sonuç verecektir. Herhangi bir tahmin söz konusu değilse, görüntünün tamamının tanımlayıcı vektörleri çıkartılarak referans tanımlayıcılarla karşılaştırılması gerekecektir.

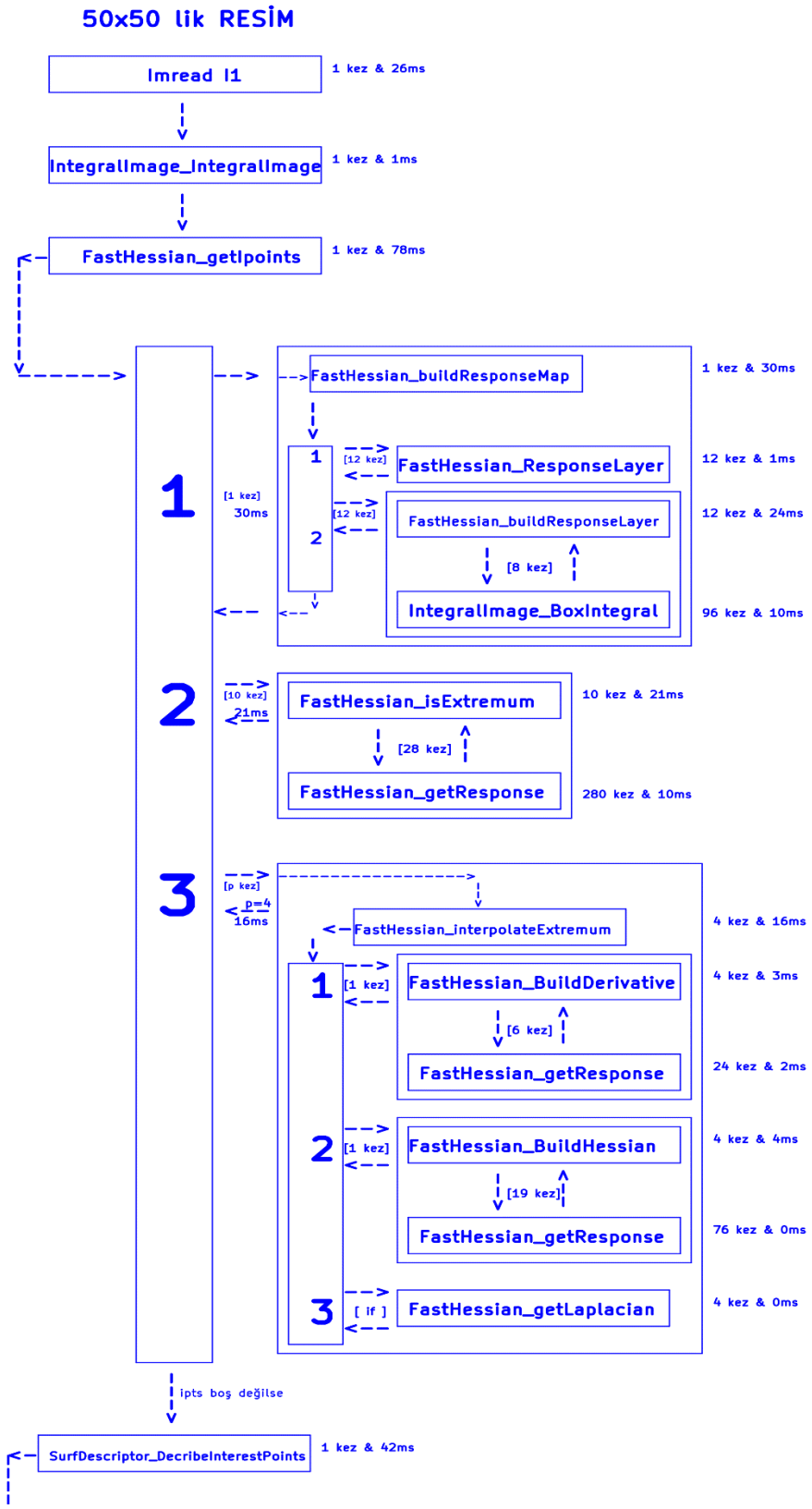
Elde edilen vektörler ile referans vektörler arasındaki benzerliklere göre bulunan ilgi noktaları sıralanır. En çok benzer olan ilk sırada olacak şekilde bir sıralama yapılır. Yani, SURF algoritmasına göre iki görüntü arasında en çok benzeyen iki pikselin koordinatları tespit edilmiş olur. Bu piksellerin sayısı referans görüntü üzerinde ve arama yapılan görüntü üzerinde bulunan ilgi noktalarının sayısına göre değişiklik göstermektedir. Ayrıca görüntü üzerinde herhangi bir ilgi noktası da bulunamayabilir.

OPENSURF kütüphanesinde gerekli düzenlemeler yapılarak sistemin tez için uygun olarak çalışması sağlandı. Bu kapsamda, kütüphanede kullanılan fonksiyonlar incelenerek yapılan matematik işlemlerinin analiz edilmesi, sistem analizlerini daha rahat yapabilmek için kütüphanenin çeşitli kısımlarına bilgi mesajlarının eklenmesi, örnek veri seti için çekilen video görüntüsünün sisteme entegre edilip referans ilgi noktalarının oluşturulması ve her bir çerçevede bu ilgi noktalarının aranması, görüntünün tamamında arama yapmamak için en uygun ROI'nin tespit edilmesi ve arama işlemlerinde ilgili bölgenin işlenmesi, veri kaybı en az olacak şekilde veri tiplerinin sabit-nokta (fixed-point) olarak değiştirilmesi, matlab ile son hali belirlenen sistemin modüller halinde HLS'e entegre edilmesi işlemleri gerçekleştirildi. HLS ile yapılan sentezleme işlemlerinde donanım olarak "Zedboard Zynq Evaluation and Development Kit (xc7z020clg484-1)" kullanılmıştır ve sistem frekansı olarak 100 MHz seçilmiştir.

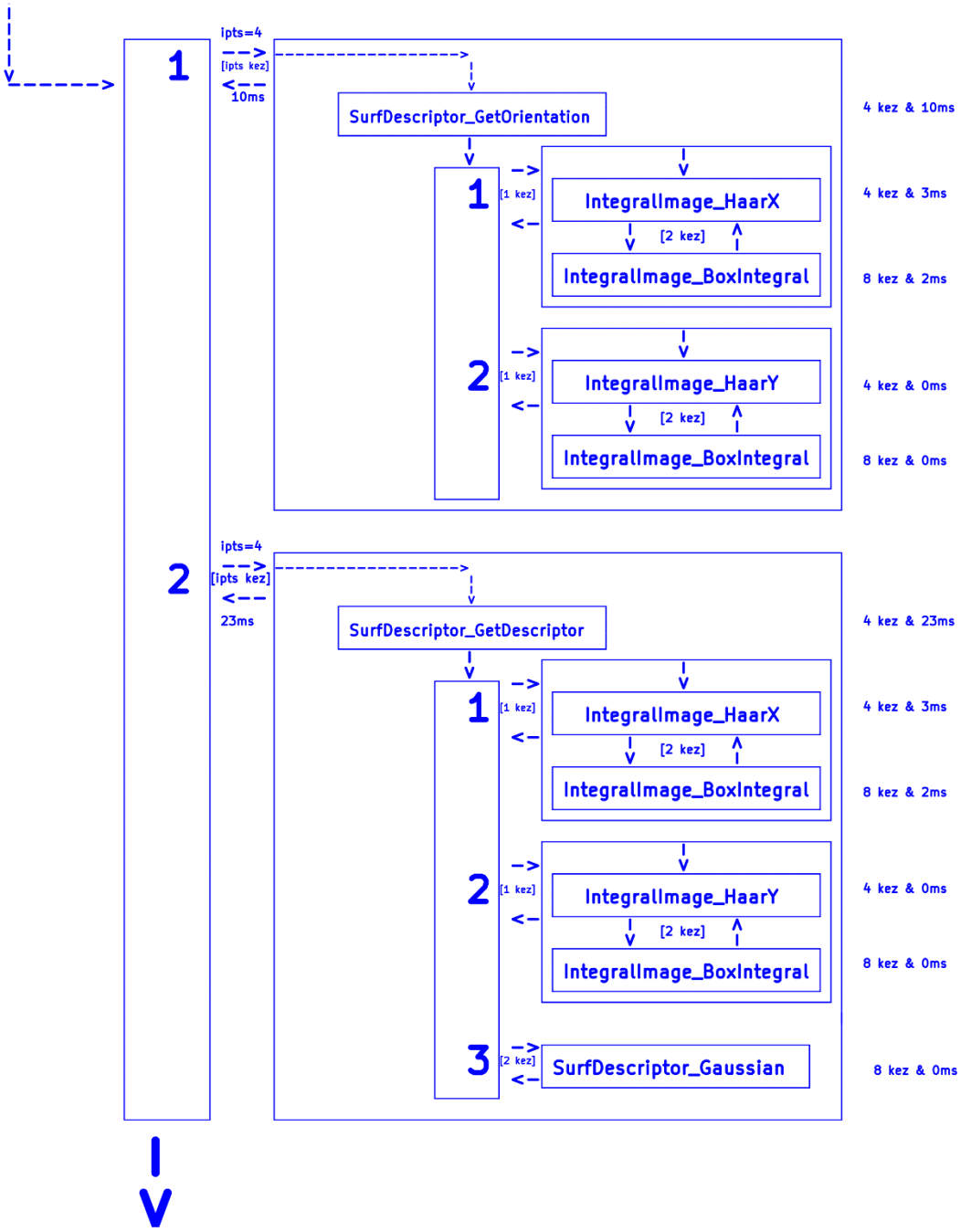
Yapılan çalışmada dönme kontrolü olarak Upright SURF (U-SURF) olarak bahsedilen ve ± 15 dereceye kadar görüntü rotasyonları için dayanıklılığı koruyan mod kullanılmıştır. Ayrıca çok fazla ölçek değişimi olmadığı için oktav değeri olarak 1 seçilmiştir. Oktav değeri, ölçek değişimi daha fazla olan görüntüler için artırılarak ilgi noktalarının daha başarılı olarak tespiti sağlanabilir. Fakat oktav değerinin artırılması işlem yükünü arttırmakta ve dolayısıyla sistemin daha yavaş çalışmasına sebep olmaktadır.

Şekil 3.19, 3.20, 3.21, 3.22'de sistemin genel yapısı ve kullanılan genel fonksiyonlar gösterilmiştir. Burada 50x50 ve 200x200 boyutlarındaki resimler referans alınarak analizler yapılmıştır. Ayrıca fonksiyonların kaç kez kullanıldıkları ve kaç milisaniyede tamamlandıklarının bilgileri de gösterilmiştir. 50x50 boyutlarındaki resim referans resim olarak kullanılmıştır ve 200x200 boyutlarındaki resim üzerinde ilgili referans resminin ilgi noktaları aranmıştır. Son olarak iki resim arasındaki benzeşen ilgi noktaları tespit edilip sıralanarak eşleştirilmiştir.

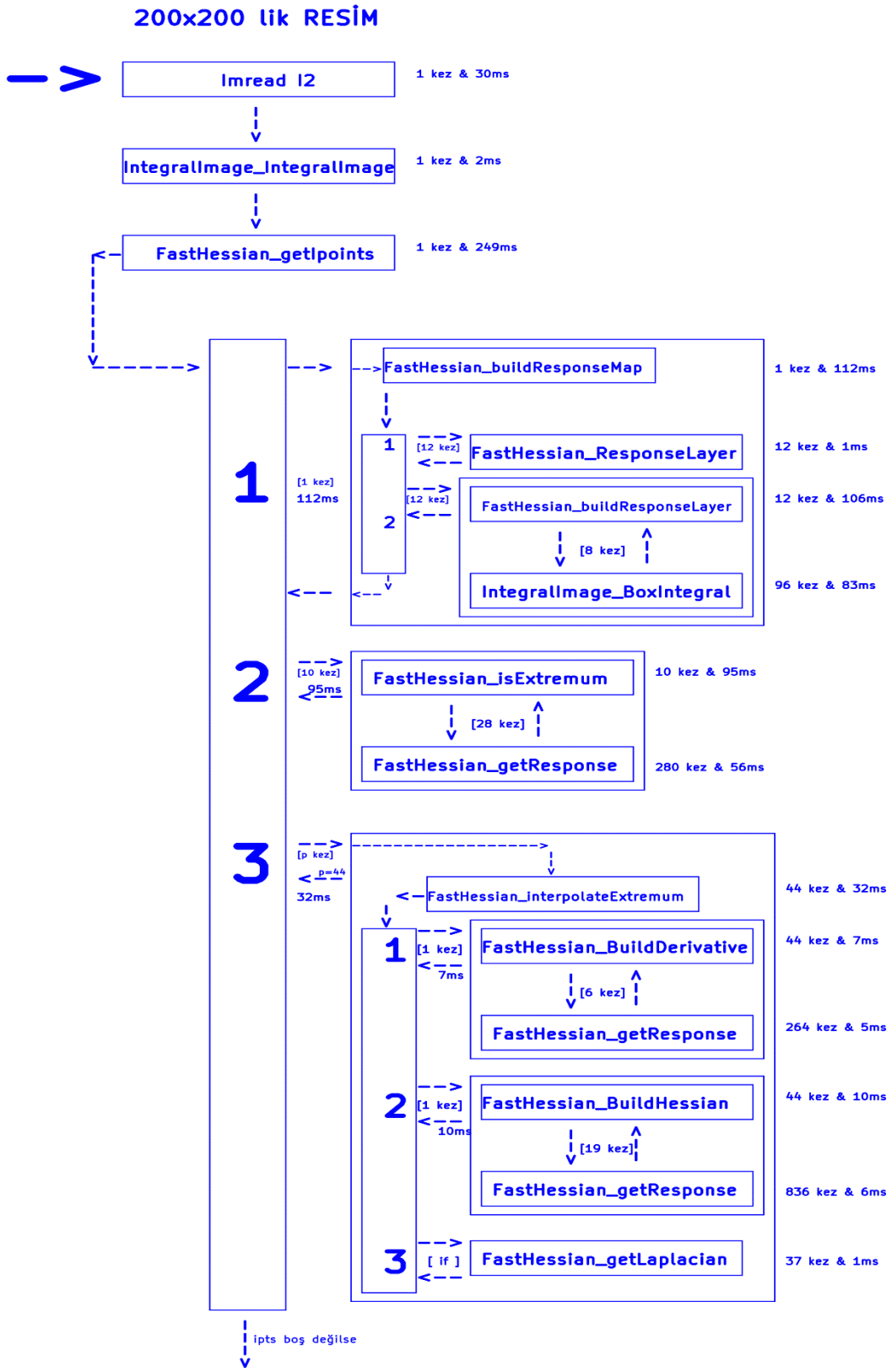
Çekilen bir video görüntüsünün resimleri test veri seti olarak kullanılmıştır. İlk çerçeve üzerinde referans ilgi noktaları tespit edilir ve diğer çerçeveler üzerinde ilgi noktaları tespit edildikten sonra eşleştirilerek ilgi noktaları sıralanır. Sistemin başarısını hareketli çekimler için test etmek amacıyla, video görüntüsü kaydedilirken hafif sarsıntı yapılmıştır. Böylece sarsıntılı bir ortamda çekilen video görüntüsünün SURF algoritması ile analizi yapılmıştır. Bu kapsamda 50 çerçeveden oluşan video görüntüsü test verisi olarak kullanılmıştır. İlk önce çerçevelerin kayma miktarları manuel olarak tespit edilmiştir. Ardından OPENSURF algoritması ile ilgi noktalarına göre kayma miktarları tespit edilmiştir. Manuel olarak ve algoritma tarafından belirlenen kayma miktarlarının yaklaşık olarak benzer olması gerekmektedir. Bazı video görüntü çerçevelerinin net olmaması veya ilgili ROI'nin dışına kayması gibi sebeplerden dolayı kayma miktarları farklılık göstermiştir. Analizler sırasında bu değerler kullanılmamış, manuel olarak tespit edilen ve otomatik olarak tespit edilen kayma değerlerinin örtüştüğü görüntü çerçeveleri kullanılmıştır. Manuel ve otomatik olarak tespit edilen sonuçların örtüşmesinin gerekliliği, OPENSURF algoritmasının her çerçeve de doğru pikseli tespit edip etmediğinin kontrol edilmesinden dolayıdır.



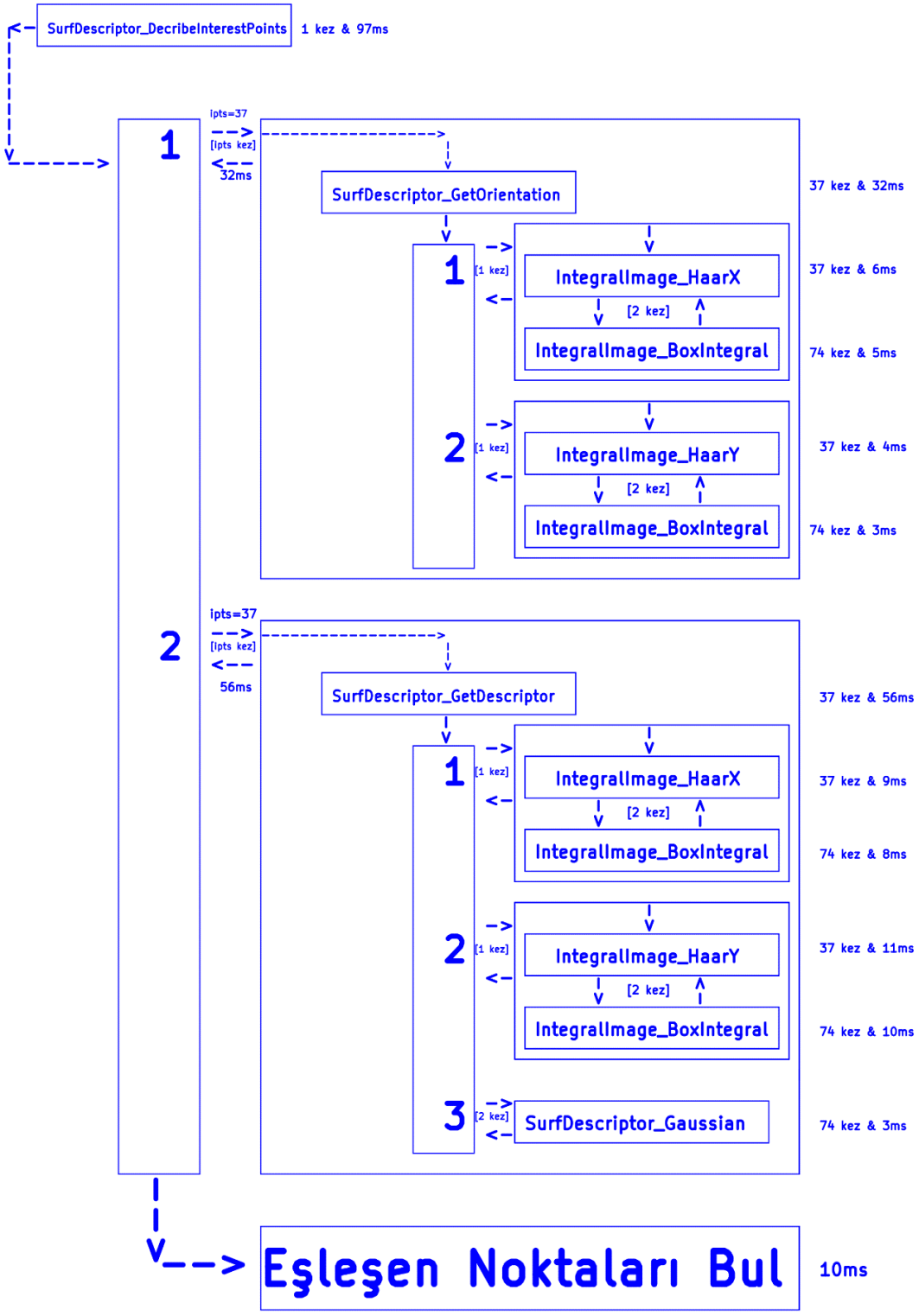
Şekil 3.19. OPENSURF algoritması (MATLAB) – 1



Şekil 3.20. OPENSURF algoritması (MATLAB) - 2



Şekil 3.21. OPENSURF algoritması (MATLAB) – 3



Şekil 3.22. OPENSURF algoritması (MATLAB) - 4

3.5. Genetik Algoritma (GA)

Günümüzde kullanılan yapay zeka teknikleri arasında önde gelenlerden birisi de Genetik Algoritmalar (GA) olarak karşımıza çıkmaktadır. Genetik algoritma genler üzerinde meydana gelen biyolojik olayların günümüz problemlerine uyarlanması sonucu ortaya çıkmıştır (Yeniay 2001). Genetik algoritmalar John Holland tarafından bulunmuştur. Genetik algoritmalar çok değişkenli fonksiyonların optimizasyon uygulamalarında kullanılan sezgisel tabanlı algoritmalardır. Genetik algoritma kör bir arama motoruna benzetilebilir (Batık vd. 2014). Problemin karmaşıklığının hiçbir önemi yoktur. Genetik algoritma problemin zorluk veya kolaylık derecesi ile ilgilenmeden problemde tanımlanmış olan karar değişkenlerini kullanarak çözüm için faydalı olacak olan tanımlamaların yapıldığı uygunluk fonksiyonunu kullanarak işlemlerine devam eder. Burada karar değişkenleri ve tasarlanan uygunluk fonksiyonun probleme doğru ve hızlı çözüm üretilmesi için önemi büyüktür.

Yapılan çalışmada GA kullanılarak OPENSURF algoritmasında kullanılan eşik değeri ve kutu filtrelerin büyüklüklerinin optimize edilmesi amaçlanmıştır. Bunun için GA, eşik değeri ve kutu filtre boyutlarını değiştirerek doğru eşleşen ilgi noktalarının sayısını kontrol eder. Doğru eşleşen ilgi noktalarının sayısını maksimum yapmaya çalışır.

Yapılan çalışmada GA ile SURF algoritmasında bulunan beş parametrenin optimum değeri belirlenmiştir. Bu parametreler, eşik değeri ve oktav-1 (4 adet) değerleridir. Genetik Algoritma'nın literatürde belirtilen, etkin arama yaparak çok kısa sürede çözüme ulaşması, çözümlerden oluşan popülasyonu eş zamanlı inceleyerek yerel en iyi çözümlere takılmamaları ve problemin başarısını etkileyen parametrelerin fazla olduğu durumlarda kullanılması gibi avantajları (Karasoy vd. 2016) olduğu için yapılan çalışmada tercih edilmiştir.

Optimizasyon için varsayılan ayarlar kullanılmıştır. Popülasyon büyüklüğü olarak 50, seçim fonksiyonu olarak "stochastic uniform" ve mutasyon oranı olarak "0.01" seçilmiştir. Optimizasyonu sınırlamak için yapılan ayarlar eşik değeri için seçilen alt ve üst sınır 0.0001 – 0.01, oktav-1 değerleri için seçilen alt ve üst sınırlar [4 5 10 15] – [10 15 27 27] şeklindedir.

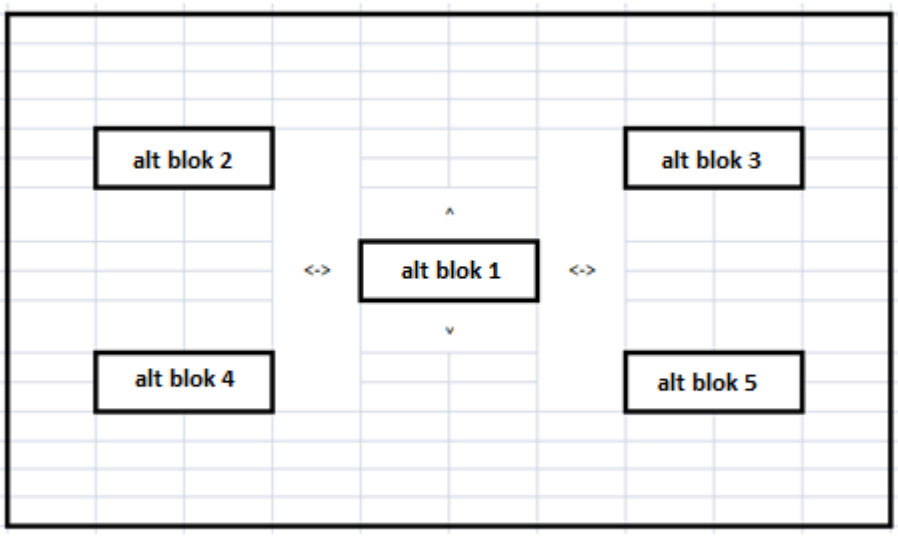
4. BULGULAR

Bu bölümde, algoritmanın test edilerek en uygun parametrelerin (eşik değeri , filtre boyutları) belirlenmesi ve donanım tarafında tespit edilen bu değerlerin kullanılması amaçlanmıştır. Bunun için 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 rakamlarından oluşan referans görüntüleri oluşturulmuştur. Test görüntüleri olarak da bu rakamların farklı büyüklükleri ve 10 derece döndürülmeleri ile oluşan görüntüler kullanılmıştır. En uygun değerlerin tespiti için matlab optimizasyon aracı olarak genetik algoritması kullanılmıştır. SURF algoritması ile dönme ve büyüklükten bağımsız olarak rakamların tespit edilmesi ve başarı oranlarının tespit edilerek incelenmesi amaçlanmıştır. Yapılan değişikliklerin sonuç üzerindeki etkileri ortaya çıkarılmıştır.

4.1. OPENSURF Algoritmasının HLS İle Oluşturulması

4.1.1. Çerçeve büyüklüğünün belirlenmesi

Referans olarak belirlenen ilgi noktalarının aranması için bir bölge belirlenmelidir. Bu bölge, çerçevenin tamamı olabileceği gibi belirli bir bölgesi de olabilir. Şekil 4.1'de kullanılan yapı gösterilmiştir. Burada, görüntü üzerinde 5 alt bölge oluşturulur. Her alt bölgeye SURF algoritması uygulanarak ilgi noktaları tespit edilir. 1920*1080 boyutlarındaki video görüntüsü düşünülürse, 16k*9k oranlarında bir çerçeve seçimi uygun olacaktır.



Şekil 4.1. Uygun çerçevenin belirlenmesi

Çeşitli ebatlarda çerçeveler belirlenerek bulunan ilgi noktalarının sayısı incelendiğinde en küçük ve en uygun olabileceği düşünülen boyut olarak 45x80 seçilmiştir. Çizelge 4.1'de alt blok 1 için bu sayının nasıl değiştiği görülmektedir.

Çizelge 4.1. Çerçeve büyüklüğüne göre bulunan ilgi noktalarının sayısı

	9x16	18x32	27x48	36x64	45x80	54x96	63x112
alt blok 1	**0**	**0**	**0**	**3**	**12**	**17**	**30**

Yapılan çalışmada alt blok 1 kullanılmıştır. Diğer bloklar da işlenerek blokların ortalama verileri kullanılıp doğruluğu daha yüksek sonuçlar ortaya çıkarılabilir. Burada önemli olan diğer bir nokta ise, çalışma alanının merkeze yakın seçilmesidir. Merkeze yakın bir alandan seçilen çalışma bölgesindeki kayma miktarı daha az olacaktır.

4.1.2. Manuel kayma miktarlarının belirlenmesi

Veri seti olarak bir video görüntüsü kullanıldı. Görüntü çerçeveleri tek tek okunarak SURF algoritmasının işlemleri görüntüler üzerinde gerçekleştirildi. Video çekilirken, pencerenin ne kadar kaydığı yaklaşık olarak tespit edilerek Çizelge 4.2'de gösterilmiştir.

Çizelge 4.2. Manuel ve otomatik olarak bulunan kayma miktarları

	abs_x	abs_y	Manuel Kayma - x	Manuel Kayma - y	Bulunan özellik sayısı
çerçeve 5	0,02	0,19	15	-4	3
çerçeve 7	0,27	0,89	24	7	7
çerçeve 14	0,26	0,80	3	7	6
çerçeve 15	0,65	0,04	-1	-1	8
çerçeve 16	0,42	0,14	-13	7	8
çerçeve 17	0,85	0,82	-19	9	8
çerçeve 19	0,93	0,76	-19	2	8
çerçeve 20	1,95	0,84	-9	-6	9

Bulunan ilgi noktasının doğruluğu Çizelge 4.2'de gösterildiği gibi 'abs_x' ve 'abs_y' değerleri ile anlaşılabilmektedir. Bu iki değer mutlak kayma değerlerini göstermektedir. Yani manuel olarak tespit edilen kayma miktarları ile SURF algoritması tarafından tespit edilen kayma miktarlarının farklarının mutlak değerlerini temsil etmektedir. Kayma miktarının sıfır olması için bu iki değerinde sıfır olması gerekir. Çeşitli frame'lerden örnek verilen Çizelge 4.2'e göre x eksenindeki kayma miktarı +24 ve -19 arasında, y eksenindeki kayma miktarı +9 ve -6 arasında kabul edilebilir. Ayrıca ilgili bölgelerde bulunan özellik sayıları da gösterilmiştir.

4.1.3. Referans ilgi noktalarının tespiti

Video görüntüsünün ilk çerçevesi referans ilgi noktalarını tanımlamak için kullanıldı. İlk çerçeve, SURF algoritması kullanılarak işlenir ve bulunan ilgi noktaları ile temsil edilir. Buradaki işlemler "alt blok 1" adlı bölge üzerinde gerçekleştirilmiştir.

4.1.3.1. İntegral görüntünün oluşturulması

SURF algoritmasının temelinde integral görüntü vardır. İlk olarak tüm resim çerçevesinin integral görüntüsü çıkarılmalı ve ilgili dizilerde saklanmalıdır. Çünkü sonraki işlem adımlarında sürekli bu değerler kullanılacaktır. İntegral görüntüyü hesaplamak için ilgili görüntü, gri seviyeye dönüştürülür ve sıfır ile bir arasında normalize olacak şekilde düzenlenir. Normalize edilen değerler kullanılarak integral görüntü değerleri hesaplanır.

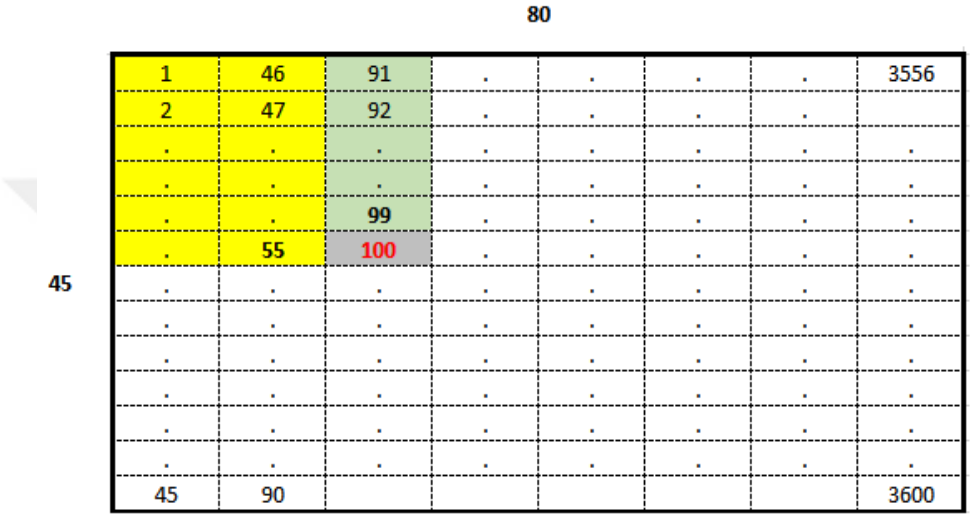
İntegral görüntü her piksel için hesaplanır. İlgili piksel dahil olmak üzere, ilgili pikselin sol tarafında ve üst tarafında kalan bölge içerisinde bulunan piksellerin renk değerlerinin toplanmasıyla o piksel için integral görüntü değeri elde edilir. Bu işlemler sırasında aynı piksel değerlerinin birden çok benzer işlemle toplandığı görülmektedir. Her piksel için en baştan piksel değerlerinin toplanması sistemin performansı açısından istenmeyen bir durum oluşturur. Bunun için yapılan ortak işlemlerin sonuçları kullanılarak diğer hesaplamalar devam ettirilir.

Matlab ile integral görüntü "pic = cumsum(cumsum(I,1),2);" kodu ile gerçekleştirilebilmektedir. Yapılan denemeler ve analizler sonucunda integral görüntü işlemleri için kullanılan saklayıcıların ondalık kısmı 12 bit ile ifade edilmiştir. Matlab da değişkenler "double" tipinden sabit nokta (fixed-point) tipinde olacak şekilde güncellenerek bu durum simule edilmiştir. Örnek olarak, ilk piksel için "double" formatında hesaplanan integral görüntü değeri 0.251306274509804'dir. Ondalık kısmı 12 bit olacak şekilde sabit nokta (fixed-point) tipinde dönüşüm yapılırsa elde edilen sonuç 0.251220703125000 olmaktadır. Sonuç olarak bu örnek için hata miktarı 0.000085571384804 olmaktadır.

Saklayıcıların kaç bit uzunluğunda olacağına karar verildikten sonra integral görüntü hesaplama fonksiyonu HLS de sentezlenmek üzere oluşturulmuştur. Çerçevenin ilk sütunu düşünüldüğünde aşağı yönde ardışık toplama, ilk satırı düşünüldüğünde sağa doğru yine ardışık toplama olduğu görülmektedir. Yeni değerler ve bir önceki toplam değeri ile hesaplamalar gerçekleştirilir. İlk sütun ve satır dışında kalan pikseller için ise, ilgili piksel değeri ile önceki integral değeri ve toplam değerleri kullanılarak hesaplanır. Örnek olarak, Şekil 4.2'de yüz numaralı pikselin integral değerini hesaplamak için yüz numaralı pikselin değerine, ilgili sütunda ilgili piksele kadar olan piksel değerleri toplamına (91-99 arası) ve bir önceki sütunun aynı satırındaki piksel için hesaplanan integral değerine ihtiyaç vardır. Bu üç değer toplanarak ilgili piksel için integral değeri hesaplanır.

HLS planlamasında integral hesabını yapan modül HLS_0 olarak oluşturulmuştur. Şekil 4.2'de bahsedilen algoritma C kodu ile oluşturularak simülasyonu yapılmıştır. Matlab ile sonuçları karşılaştırabilmek için, matlab da

hesaplanan değerler '.dat' uzantılı dosya olarak HLS'de kullanılmıştır. Şekil 4.3 ve Şekil 4.4'de yapılan test sonuçları gösterilmiştir. Buna göre, test-1 de veri tipi olarak 'float' kullanılmıştır ve herhangi bir direktif kullanılmamıştır. Test-2 de ise veri tipi olarak sabit nokta (fixed-point) kullanılmıştır. Ayrıca uygun direktifler kullanılmıştır (#pragma HLS PIPELINE, #pragma HLS ARRAY_MAP vertical, #pragma HLS INTERFACE s_axilite). Sonuç olarak hız ve hafıza kullanımında şekilde görüldüğü gibi iyileşmeler olmuştur. Hesaplama sonuçlarındaki farklılıklar ve matlab sonuçları ile arasındaki sapma miktarları Şekil 4.4'de gösterilmiştir.



Şekil 4.2. HLS ile integral görüntünün hesaplanması

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	7.26	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
32393	75581	32394	75582	none

Detail

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	25
FIFO	-	-	-	-
Instance	16	4	702	1074
Memory	-	-	-	-
Multiplexer	-	-	-	308
Register	-	-	228	-
Total	16	4	930	1407
Available	280	220	106400	53200
Utilization (%)	5	1	~0	2

Test - 1

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.60	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
10812	10812	10813	10813	none

Detail

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	134
FIFO	-	-	-	-
Instance	14	-	431	425
Memory	-	-	-	-
Multiplexer	-	-	-	166
Register	-	-	166	13
Total	14	0	597	738
Available	280	220	106400	53200
Utilization (%)	5	0	~0	1

Test - 2

Şekil 4.3. HLS ile integral görüntü hesabının sonuçları - 1

Test - 1

```
iimg[1] = 0.251306
iimg[2] = 0.632011
iimg[3] = 1.135167
iimg[4] = 1.850066
iimg[5] = 2.530376
iimg[6] = 3.116130
iimg[7] = 3.694489
iimg[8] = 4.332838
iimg[9] = 4.972528
iimg[10] = 5.540017
```

Test - 2

```
iimg[1] = 0.251221
iimg[2] = 0.631836
iimg[3] = 1.134766
iimg[4] = 1.849609
iimg[5] = 2.529785
iimg[6] = 3.115479
iimg[7] = 3.693604
iimg[8] = 4.331787
iimg[9] = 4.971436
iimg[10] = 5.538818
```

Matlab

	1
1	0.2512
2	0.6321
3	1.1353
4	1.8501
5	2.5303
6	3.1162
7	3.6946
8	4.3328
9	4.9724
10	5.5400

Şekil 4.4. HLS ile integral görüntü hesabının sonuçları - 2

4.1.3.2. 'FastHessian_buildResponseMap' fonksiyonunun oluşturulması

'FastHessian_buildResponseMap' fonksiyonu yardımıyla belirli bir ölçek-uzayı seviyesindeki determinant tepki haritası (response map) belirlenir. Ölçek uzayı olarak oktav değeri 1 seçilmiştir. Buna göre dört tane "response_map" veri seti oluşur. Oktav değeri 1 için 9, 15, 21, 27 boyutlarında filtreler kullanılarak işlemler gerçekleştirilir. "response_map" veri seti içerisinde çerçeve yüksekliği, çerçeve genişliği, örnekleme değeri, filtre boyut bilgisi, hessian matrisinin determinantı ve işaret bilgileri yer almaktadır. Normalize etmek için kullanılan çarpım değeri kullanılan filtre boyutlarının karelerine bölünerek elde edilir. Örnek olarak $\frac{1}{9^2}$ gösterilebilir.

(3.10) numaralı denklemde ifade edildiği gibi hessian matrisinin determinantı tespit edilir. Buradaki "w" değeri 0.81 olarak seçilmiştir (Bay vd. 2006). Determinantın işaret bilgisi için ($(D_{xx} + D_{yy}) >= 0$) formülü kullanılır. Sonuç olarak 45x80 ebatlarında bir bölge düşünüldüğünde, oluşacak olan "response_map" veri setinin her bir üyesinin 3600 determinant sonucu ve 3600 determinant işareti sonucu olacaktır.

Matlab da varsayılan olarak saklayıcılar "double" olarak saklandığı için donanımsal gerçekleme işlemlerinde dikkat edilmelidir. Önemli olan veri kayıplarının istenilen limiti aşmamasıdır. Çünkü değerlerde yaşanabilecek büyük sapma değerleri sonuçların hatalı olmasına sebep verebilir.

Fonksiyonun genel yapısı Şekil 4.20'de "HLS_1" modülü olarak gösterilmiştir. HLS_1 modülünde temel olarak kutu filtrelerinin (Dxx, Dyy, Dxy) hesaplanması, hesaplanan filtre değerlerinin normalize edilmesi, normalize edilen değerler kullanılarak Hessian matrisinin determinantı ve işaret bilgisinin hesaplanması adımları yapılmaktadır. İlk olarak oktav değerine göre belirlenen filtre boyutları ile kutu filtreler uygulanmıştır. Bu işlemler için HLS_1 modülünde yerel olarak "IntegralImage_BoxIntegral" fonksiyonu oluşturulmuştur. Bu yerel fonksiyonda tanımlanan saklayıcıların 'static' olarak tanımlanması gerekmektedir. Böyle bir tanımlama yapılmadığında yerel fonksiyonlar her çağrıldığında saklayıcılar için hafıza da yeni bir alan oluşturulur ve kullanılan block-ram miktarında artış olur. Matlab analizlerinde, kullanılan saklayıcılar için en uygun ondalıklı kısım bit uzunluğu olarak "IntegralImage_BoxIntegral" fonksiyonunda 12 bit seçilmiştir. HLS de kodlar bu bit uzunluğu ile oluşturulmuştur. İkinci olarak hesaplanan kutu filtre sonuçları $\frac{1}{filtreboyutu^2}$ değeri ile çarpılarak normalize edilmiştir. Buradaki filtre boyutları oktav değeri 1 için 9, 15, 21, 27 olmaktadır. Normalizasyon işlemleri sırasında ondalıklı kısım bit uzunluğu olarak 16 bit seçilmiştir. Son olarak normalize edilen değerler kullanılarak hessian matrisinin determinantı ve işareti hesaplanır. Determinant hesabı (3.10) numaralı denklemde gösterildiği gibi yapılmaktadır. İşaret bilgisi için, ($(D_{xx} + D_{yy}) >= 0$) formülünde ifade edildiği gibi eşitlik sağlanıyorsa 1, sağlanmıyorsa 0 olacak şekilde sonuçlar oluşturulur. Lojik bir tanım olduğundan dolayı işaret bilgisi HLS sentezlemesi sırasında 1 bit ile ifade edilecek şekilde tanımlanmıştır.

HLS ve VIVADO için ZYNQ işlemci ile oluşturulan modüller arasındaki haberleşme için çeşitli arabirimler mevcut. “s_axilite” veya “m_axilite” seçilerek oluşturulan bağlantılarda ZYNQ ile oluşturulan modüller arasındaki haberleşme, kontrollü bir şekilde veri yolu üzerinden gerçekleştirilir. Bu şekilde bir tanımlama yapıldığında oluşturulan modüller “ip-core” olarak VIVADO içerisine aktarılarak ZYNQ ile bir ara bağlantı aracılığıyla haberleştirilebilir. ZYNQ merkezli bir sistem için böyle bir yapının kullanımı uygun görülmüştür.

Matlab’da belirlenen bit uzunluklarına göre HLS_1 modülü HLS’de oluşturulmuştur. Daha sonra oluşturulan kod çeşitli ayarlar ve iyileştirmeler yapılarak test edilmiştir. Yapılan ayarlar ve iyileştirmeler aşağıda detaylı olarak bahsedilmiştir.

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	10.73	1.25

Latency		Interval		Type
min	max	min	max	
3225790	3340991	3225791	3340992	none

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	1984
FIFO	-	-	-	-
Instance	40	8	2564	3711
Memory	96	-	0	0
Multiplexer	-	-	-	1949
Register	-	-	2025	-
Total	136	8	4589	7644
Available	280	220	106400	53200
Utilization (%)	48	3	4	14

HLS	MATLAB
responseMap_test.responses_0[1] = -0.002037	-0,00202941894531250
responseMap_test.responses_0[2] = -0.000742	-0,00073242187500000
responseMap_test.responses_0[3] = -0.000192	-0,00018310546875000
responseMap_test.responses_0[4] = -0.000017	-1,52587890625000e-05
responseMap_test.responses_0[5] = -0.000009	0
responseMap_test.responses_0[6] = 0.000006	1,52587890625000e-05
responseMap_test.responses_0[7] = -0.000092	-9,15527343750000e-05
responseMap_test.responses_0[8] = -0.000138	-0,00013732910156250
responseMap_test.responses_0[9] = -0.000059	-6,10351562500000e-05
responseMap_test.responses_0[10] = -0.000082	-9,15527343750000e-05

Şekil 4.5. HLS_1 modülünün sentezlenmesi test – 1

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	7.83	1.25

Latency		Interval		Type
min	max	min	max	
1238710	1353911	1238711	1353912	none

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	1815
FIFO	-	-	-	-
Instance	40	56	1292	1345
Memory	266	-	0	0
Multiplexer	-	-	-	2177
Register	-	-	2748	-
Total	306	56	4040	5337
Available	280	220	106400	53200
Utilization (%)	109	25	3	10

HLS	MATLAB
responseMap_test.responses_0[1] = -0.002045	-0,00202941894531250
responseMap_test.responses_0[2] = -0.000748	-0,00073242187500000
responseMap_test.responses_0[3] = -0.000198	-0,00018310546875000
responseMap_test.responses_0[4] = -0.000031	-1,52587890625000e-05
responseMap_test.responses_0[5] = -0.000015	0
responseMap_test.responses_0[6] = 0.000000	1,52587890625000e-05
responseMap_test.responses_0[7] = -0.000107	-9,15527343750000e-05
responseMap_test.responses_0[8] = -0.000153	-0,00013732910156250
responseMap_test.responses_0[9] = -0.000061	-6,10351562500000e-05
responseMap_test.responses_0[10] = -0.000092	-9,15527343750000e-05

Şekil 4.6. HLS_1 modülünün sentezlenmesi test – 2

İlk olarak “float” veri tipi kullanılıp herhangi bir direktif kullanılmadan test yapılmıştır. Test sonuçları Şekil 4.5’de gösterilmiştir. Test sonucuna göre modül çalışma zamanı yaklaşık olarak 32.2 milisaniye sürmekte ve kullanılan ram miktarı %48 olarak ölçülmektedir.

Test – 2’de veri tipi olarak sabit-nokta (fixed-point) seçilerek sentezleme yapılmıştır. Sonuçlar Şekil 4.6’da gösterilmiştir. Buna göre kullanılan ram miktarı %109’a, DSP48 miktarı %25’e yükselmiştir. Buradan görüldüğü gibi donanımın sahip olduğu ram miktarı aşılmıştır. Çalışma zamanı ise yaklaşık olarak 12.5 milisaniyeye düşmüştür. Kullanılan ram miktarlarının fazla olma sebebi araştırıldığında ise HLS’de oluşturulan yerel fonksiyonlarda kullanılan değişkenlerin “static” olarak tanımlanmamasından dolayı olduğu tespit edilmiştir. Yani “static” olarak tanımlama yapılmadığında, yerel fonksiyon ana fonksiyon tarafından her çağrıldığında yerel fonksiyonda kullanılan değişkenler hafızada tekrar tanımlanarak gereksiz depolama alanına sebep olmaktadır. “static” olarak tanımlama yapıldığı zaman ram kullanımı %52’e kadar düşmektedir.

Test – 3’de, yerel fonksiyonlarda kullanılan değişkenler “static” olarak ayarlandı. Kullanılan bazı diziler ortak kullanılacak şekilde yeniden düzenlendi ve gereksiz olan değişkenler kaldırıldı. “ARRAY_MAP” direktifi kullanılarak diziler iki grup (dikey ve yatay) altında toplandı ve ram kullanımı optimize edildi. Çalışma süresini azaltmak için döngülerde “PIPELINE” direktifi kullanılarak döngülerin paralel çalışması sağlandı. “config_bind” ve “config_schedule” ayarları “high” olarak seçildi. Tüm bu iyileştirmelerden sonra sentezleme sonuçları şu şekilde oluşmuştur (Şekil 4.7):

- Çalışma süresi 12.5ms den 4.5ms e düşmüştür.
- BRAM kullanımı %109 dan %18 e düşmüştür.
- DSP48E kullanımı %25 den %8 e düşmüştür.
- FF kullanımı %3 den %2 e düşmüştür.
- LUT kullanımı ise %10 olarak kalmıştır.

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.51	1.25

Latency		Interval		Type
min	max	min	max	
454155	454155	454156	454156	none

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	469
FIFO	-	-	-	-
Instance	16	18	870	957
Memory	36	-	0	0
Multiplexer	-	-	-	4155
Register	-	-	1667	252
Total	52	18	2537	5833
Available	280	220	106400	53200
Utilization (%)	18	8	2	10

HLS	MATLAB
responseMap_test.responses_0[1] = -0.002045	-0,00202941894531250
responseMap_test.responses_0[2] = -0.000748	-0,000732421875000000
responseMap_test.responses_0[3] = -0.000198	-0,000183105468750000
responseMap_test.responses_0[4] = -0.000031	-1,52587890625000e-05
responseMap_test.responses_0[5] = -0.000015	0
responseMap_test.responses_0[6] = 0.000000	1,52587890625000e-05
responseMap_test.responses_0[7] = -0.000107	-9,15527343750000e-05
responseMap_test.responses_0[8] = -0.000153	-0,000137329101562500
responseMap_test.responses_0[9] = -0.000061	-6,10351562500000e-05
responseMap_test.responses_0[10] = -0.000092	-9,15527343750000e-05

Şekil 4.7. HLS_1 modülünün sentezlenmesi test – 3

Test – 4'de ise küçük bir değişiklik yapılarak ram kullanımı ve çalışma zamanı bir miktar daha azaltılmıştır. Burada diziler iki grup altında değil tek grup (dikey) altında toplanmıştır. Bunun sonucunda ram kullanımı %17, çalışma zamanı yaklaşık olarak 3.4ms olmuştur (Şekil 4.8).

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.51	1.25

Latency		Interval		Type
min	max	min	max	
338923	338923	338924	338924	none

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	393
FIFO	-	-	-	-
Instance	16	18	870	957
Memory	34	-	0	0
Multiplexer	-	-	-	4026
Register	-	-	2229	457
Total	50	18	3099	5833
Available	280	220	106400	53200
Utilization (%)	17	8	2	10

HLS	MATLAB
responseMap_test.responses_0[1] = -0.002045	-0,00202941894531250
responseMap_test.responses_0[2] = -0.000748	-0,00073242187500000
responseMap_test.responses_0[3] = -0.000198	-0,00018310546875000
responseMap_test.responses_0[4] = -0.000031	-1,52587890625000e-05
responseMap_test.responses_0[5] = -0.000015	0
responseMap_test.responses_0[6] = 0.000000	1,52587890625000e-05
responseMap_test.responses_0[7] = -0.000107	-9,15527343750000e-05
responseMap_test.responses_0[8] = -0.000153	-0,00013732910156250
responseMap_test.responses_0[9] = -0.000061	-6,10351562500000e-05
responseMap_test.responses_0[10] = -0.000092	-9,15527343750000e-05

Şekil 4.8. HLS_1 modülünün sentezlenmesi test – 4

4.1.3.3. 'FastHessian_isExtremum' fonksiyonunun oluşturulması

HLS_2 modülünde aday ilgi nokta setini bulmak için En Büyük Olmayanı Bastırma (non-maximal suppression) yöntemi gerçekleştirilir. Bunu işlem için, ölçek alanındaki her piksel yerel ölçekteki 8 nokta, üst ve alt ölçeklerin her birinde 9 nokta olmak üzere toplam 26 komşu nokta ile karşılaştırılır. Ayrıca, belirlenen eşik değeriyle ve aynı zamanda ölçek-uzayındaki yerel maksimum-minimum değerine göre bir dizi ilgi noktası bulunur.

Matlab analizleri yapılarak saklayıcıların aldığı değer aralıkları incelendi (Şekil 4.9). Buna göre ondalıklı değer alabilecek dört saklayıcı tespit edildi. -1 ile +1 arasında değişen değerlere sahip bu saklayıcılar 17 bit sabit-nokta (fixed-point) ile temsil edilerek denemeler yapılmıştır (işaret bilgisi için 1 bit, ondalıklı kısım için 16 bit). Diğer saklayıcılar ondalıklı değerler içermediği için veri büyüklüğüne göre bit uzunlukları belirlenmiştir. Burada veri tipleri ve bit uzunlukları belirlendikten sonra ilgili modül HLS'de oluşturulmuştur.

FastHessian_isExtremum' fonksiyonu sonucunda oluşan 3600 (45x80 formatlı resim) elemanlı dizi içerisinde sonuçlar lojik 1 veya 0'dan oluşmaktadır. Burada, ilgili noktanın yerel maksimum veya minimum olması 1 ile ifade edilir. Dolayısıyla Matlab'da bu dizinin elemanları hesaplandıktan sonra bir de hangi elemanların 1 olduğu da ayrıca hesaplanmaktadır. Bu işlemlerin tamamı HLS_2 olarak ifade edilen HLS modülünde gerçekleştirilmiştir. Test verisinin sonuçları Şekil 4.10'da gösterilmiştir. Burada, Matlab ve HLS'de sonucu 1 çıkan dizi indeks numaraları gösterilmiştir.

HLS_2 MAIN function an=FastHessian_isExtremum(r, c, t, m, b,FastHessianData) 4 kez [2 tane 45x80 resim için]					FIXED - POINT			
Registers	MIN	MAX	TYPE	SIZE	SIGN	INTEGER	FRAC	WORD
r	0,00	44,00	uint	3600x1	0	6	0	6
c	0,00	79,00	uint	3600x1	0	7	0	7
t.width	80,00	80,00	uint	1	0	7	0	7
t.height	45,00	45,00	uint	1	0	6	0	6
t.step	1,00	1,00	uint	1	0	1	0	1
t.filter	21,00	27,00	uint	1	0	5	0	5
m.width	80,00	80,00	uint	1	0	7	0	7
m.responses	-0,03	0,05	double	3600x1	1	0	16	17
b.width	80,00	80,00	uint	1	0	7	0	7
b.responses	-0,03	0,05	double	3600x1	1	0	16	17
FastHessianData.thresh	0.0001 => parameter 0.0000 0000 0000 0111			1	1	0	16	17
layerBorder	11,00	14,00	uint	1	0	4	0	4
candidate	-0,03	0,05	double	3600x1	1	0	16	17
bound_check_fail	logical			3600x1	0	1	0	1
treshold_fail	logical			3600x1	0	1	0	1
check1	logical			3600x1	0	1	0	1
check2	logical			3600x1	0	1	0	1
check3	logical			3600x1	0	1	0	1
check4	logical			3600x1	0	1	0	1
an	logical			3600x1	0	1	0	1

Şekil 4.9. HLS_2 modülünde kullanılan saklayıcılar

İlk olarak "float" veri tipi kullanılıp herhangi bir direktif kullanılmadan test yapılmıştır (s_axilite arabirimi seçilmiştir.). Bunun sonucunda ortaya çıkan sonuçlar Şekil 4.10'da, sentezleme sonuçları ise Şekil 4.11'de gösterilmiştir. Çalışma süresi yaklaşık olarak 2.5ms, ram kullanımı %13, LUT kullanımı ise %3 olmuştur.

Matlab

	1
1	1181
2	1296
3	1387
4	1705
5	2048
6	2257
7	2454
8	2542
9	2577

HLS

```
an_sifir_olmayanlar[1181] = 1
an_sifir_olmayanlar[1296] = 1
an_sifir_olmayanlar[1387] = 1
an_sifir_olmayanlar[1705] = 1
an_sifir_olmayanlar[2048] = 1
an_sifir_olmayanlar[2257] = 1
an_sifir_olmayanlar[2454] = 1
an_sifir_olmayanlar[2542] = 1
an_sifir_olmayanlar[2577] = 1
```

Şekil 4.10. HLS_2 modülünün sonuçları

Test - 2'de "float" veri tipi yerine sabit-nokta (fixed-point) kullanılmıştır. Döngülerdeki zaman kayıpları "PIPELINE" direktifi kullanılarak azaltılmaya çalışılmıştır. Bunun sonucunda çalışma süresi azaltılmıştır. Ram kullanımını azaltmak

için "ARRAY_MAP" direktifi kullanılmıştır ve saklayıcılar iki grup altında birleştirilerek düzenlenmiştir. Yapılan iyileştirmeler sonucunda sentezleme sonucunun son hali şu şekilde olmuştur (Şekil 4.12):

- Çalışma süresi 2.5ms den 400us e düşmüştür.
- BRAM kullanımı %13 den %8 e düşmüştür.
- FF sayısı 993 (%0) den 538 (%0) e düşmüştür.
- LUT kullanımı %3 den %1 e düşmüştür.

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.19	1.25

Latency		Interval		Type
min	max	min	max	
255627	255627	255628	255628	none

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	284
FIFO	-	-	-	-
Instance	30	0	725	1187
Memory	9	-	0	0
Multiplexer	-	-	-	324
Register	-	-	268	-
Total	39	0	993	1795
Available	280	220	106400	53200
Utilization (%)	13	0	~0	3

Şekil 4.11. HLS_2 modülünün sentezlenmesi test – 1

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.64	1.25

Latency		Interval		Type
min	max	min	max	
39611	39611	39612	39612	none

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	333
FIFO	-	-	-	-
Instance	20	-	287	260
Memory	4	-	0	0
Multiplexer	-	-	-	252
Register	-	-	251	80
Total	24	0	538	925
Available	280	220	106400	53200
Utilization (%)	8	0	~0	1

Şekil 4.12. HLS_2 modülünün sentezlenmesi test – 2

4.1.3.4. ‘FastHessian_interpolateExtremum’ fonksiyonun oluşturulması

İnterpolasyon işlemi bilinen iki değeri kullanarak, aradaki bilinmeyen üçüncü bir değerin hesaplanması işlemidir. HLS_3 modülünde genel olarak bir interpolasyon işlemi yapılmaktadır. Yapılan işlemler sonucunda tespit edilen aday ilgi noktasının, gerçek bir ilgi noktası olarak kabul edilip edilmeyeceğine karar verilir.

Bu modülün diğerlerinden farkı matris çarpmaları olarak ifade edilebilir. Üçlü matris çarpmaları yapıldığında işlem sonuçlarının ondalıklı kısımları bazı hesaplamalarda çok küçülebilmektedir. Buradaki hassasiyeti korumak için ilgili saklayıcıların yaklaşık olarak 63 bit ile falan ifade edilmesi gerekiyor (sabit-nokta (fixed-point) veri tipi seçildiğinde, 1 bit işaret biti + 47 bit ondalıklı kısım + 15 bit tam sayı kısmı’). Böyle bir tanımlama ile hesaplama sonuçlarındaki sapmalar aza indirgenmiş olur. Veri tipi olarak sabit-nokta (fixed-point) seçildiğinde ise bu büyüklükteki verilerin çarpma ve bölme işlemlerini yapmak donanım açısından sıkıntılara sebep olmaktadır (donanım kapasitesinin aşılması). Bundan dolayı ilgili modül HLS’de oluşturulurken modüldeki bazı ondalıklı veriler sabit-nokta (fixed-point) veri tipi yerine “float” veri tipi olarak seçilmiştir.

Şekil 4.13’de tüm saklayıcıların veri tipi sabit-nokta (fixed-point) seçildiğinde ortaya çıkan sentezleme sonucu gösterilmiştir. Buna göre kullanılan DSP48E, FF ve LUT miktarlarının donanım tarafından karşılanamadığı ve limitlerin aşıldığı gözlenmiştir.

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	4251
FIFO	-	-	-	-
Instance	14	329	182144	182756
Memory	-	-	-	-
Multiplexer	-	-	-	794
Register	-	-	6975	-
Total	14	329	189119	187801
Available	280	220	106400	53200
Utilization (%)	5	149	177	353

Şekil 4.13. HLS_3 modülünün sentezlenmesi test – 1

Test – 2’de bazı saklayıcıları “float” veri tipi olarak, bazı saklayıcıları da sabit-nokta (fixed-point) olarak tanımladığımızda ortaya çıkan sentezleme sonucu gösterilmiştir (Şekil 4.14). Ayrıca uygun direktifler de kullanılmıştır (saklayıcılar iki grupta birleştirilmiştir.[ARRAY_MAP]). Böylece donanım limitlerinin aşılma sorunu ortadan kalkmıştır.

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	2472
FIFO	-	-	-	-
Instance	14	47	6299	11391
Memory	-	-	-	-
Multiplexer	-	-	-	3785
Register	-	-	900	-
Total	14	47	7199	17648
Available	280	220	106400	53200
Utilization (%)	5	21	6	33

Şekil 4.14. HLS_3 modülünün sentezlenmesi test – 2

Test – 3'de ise ilave olarak "LATENCY" direktifi kullanılarak iyileştirme yapılmıştır. Buna göre sentezlemenin son hali Şekil 4.15'de gösterilmiştir.

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.63	1.25

Latency		Interval		
min	max	min	max	Type
87	105	88	106	none

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	3685
FIFO	-	-	-	-
Instance	14	29	5112	8937
Memory	-	-	-	-
Multiplexer	-	-	-	1196
Register	-	-	3234	-
Total	14	29	8346	13818
Available	280	220	106400	53200
Utilization (%)	5	13	7	25

Şekil 4.15. HLS_3 modülünün sentezlenmesi test – 3

Yapılan iyileştirmeler sonucunda sentezleme sonucunun son hali şu şekilde olmuştur:

- Çalışma süresi yaklaşık 1us olmuştur.
- BRAM kullanımı %5 olmuştur.

- DSP48E kullanımı %149 dan %13 e düşmüştür.
- FF kullanımı %177 den % 7 e düşmüştür.
- LUT kullanımı %353 den %25 e düşmüştür.

4.1.3.5. 'SurfDescriptor_GetDescriptor' fonksiyonunun oluşturulması

HLS_4 modülünde, daha önceden belirlenen ilgi noktaları için tanımlayıcı vektörleri oluşturulur. Her bir ilgi noktasını temsil eden 64 boyutlu vektörler oluşturulur. Bu vektörler -1 ile +1 arasında değişen değerlere sahiptir.

Oluşturulan HLS modülünün sentezleme sonuçları Şekil 4.16 ve Şekil 4.17'de gösterilmiştir. Veri tipi olarak "float" seçildiğinde ve varsayılan optimizasyonlar ile sentezleme yapıldığında kullanılan donanım bilgileri Şekil 4.16 (solda)'da, optimizasyon ayarları yapıldığında ise kullanılan donanım bilgileri Şekil 4.17 (sağda)'de gösterilmiştir. Veri tipi olarak sabit-nokta (fixed-point) seçildiğinde ve optimizasyon ayarları yapıldığında ise kullanılan donanım bilgileri Şekil 4.17'de gösterilmiştir.

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.58	1.25

Latency		Interval		Type
min	max	min	max	
266457	301449	266458	301450	none

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	4	0	1476
FIFO	-	-	-	-
Instance	2	42	7911	12436
Memory	33	-	384	63
Multiplexer	-	-	-	2675
Register	-	-	2754	-
Total	35	46	11049	16650
Available	280	220	106400	53200
Utilization (%)	12	20	10	31

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	10.73	1.25

Latency		Interval		Type
min	max	min	max	
239283	239283	239284	239284	none

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	4	0	1463
FIFO	-	-	-	-
Instance	2	42	8117	12626
Memory	29	-	384	63
Multiplexer	-	-	-	2733
Register	-	-	3293	130
Total	31	46	11794	17015
Available	280	220	106400	53200
Utilization (%)	11	20	11	31

Şekil 4.16. HLS_4 modülünün sentezlenmesi test – 1

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.64	1.25

Latency		Interval		Type
min	max	min	max	
158733	158733	158734	158734	none

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	4	-	-
Expression	-	4	0	3571
FIFO	-	-	-	-
Instance	2	43	8614	12497
Memory	23	-	174	27
Multiplexer	-	-	-	2528
Register	-	-	3841	158
Total	25	51	12629	18781
Available	280	220	106400	53200
Utilization (%)	8	23	11	35

Şekil 4.17. HLS_4 modülünün sentezlenmesi test – 2

Yapılan iyileştirmeler sonucunda sentezleme sonucunun son hali şu şekilde olmuştur:

- Çalışma süresi yaklaşık 1.58ms olmuştur.
- BRAM kullanımı %8 e düşmüştür.
- DSP48E kullanımı %20 den %23 e çıkmıştır.
- FF kullanımı %10 dan % 11 e çıkmıştır.
- LUT kullanımı %31 den %35 e çıkmıştır.

4.1.3.6. Eşleşen noktaların bulunması

Referans görüntü için ve arama yapılacak görüntü için ayrı ayrı SURF algoritması kullanılarak ilgi noktaları tespit edilir (ilk beş adım her iki resim için de uygulanır). İki resimde tespit edilen ilgi noktalarının benzerlikleri tanımlayıcı vektörler kullanılarak belirlenir. Her ilgi noktası 64 boyutlu bir vektör ile temsil edildiğinden dolayı, iki resimde bulunan her bir vektör arasındaki mesafe bilgisi hesaplanır ve en yakın olan vektörden başlanarak ilgi noktaları tekrar sıralanır. Bu sıralama sonucuna göre iki resimde bulunan ilgi noktaları eşleşme potansiyeline göre sıralanır.

Oluşturulan HLS modülünün sentezleme sonuçları Şekil 4.18'de gösterilmiştir. Veri tipi olarak "float" seçildiğinde ve varsayılan optimizasyon ayarları yapıldığında kullanılan donanım bilgileri Şekil 4.18'de (solda) , veri tipi olarak sabit-nokta (fixed-point) seçildiğinde ve optimizasyon ayarları yapıldığında kullanılan donanım bilgileri ise Şekil 4.18'de (sağda) gösterilmiştir.

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.40	1.25

Latency		Interval		Type
min	max	min	max	
37199	37529	37200	37530	none

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	167
FIFO	-	-	-	-
Instance	-	2	0	0
Memory	3	-	66	6
Multiplexer	-	-	-	222
Register	-	-	239	-
Total	3	2	305	395
Available	280	220	106400	53200
Utilization (%)	1	~0	~0	~0

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.97	1.25

Latency		Interval		Type
min	max	min	max	
24947	25277	24948	25278	none

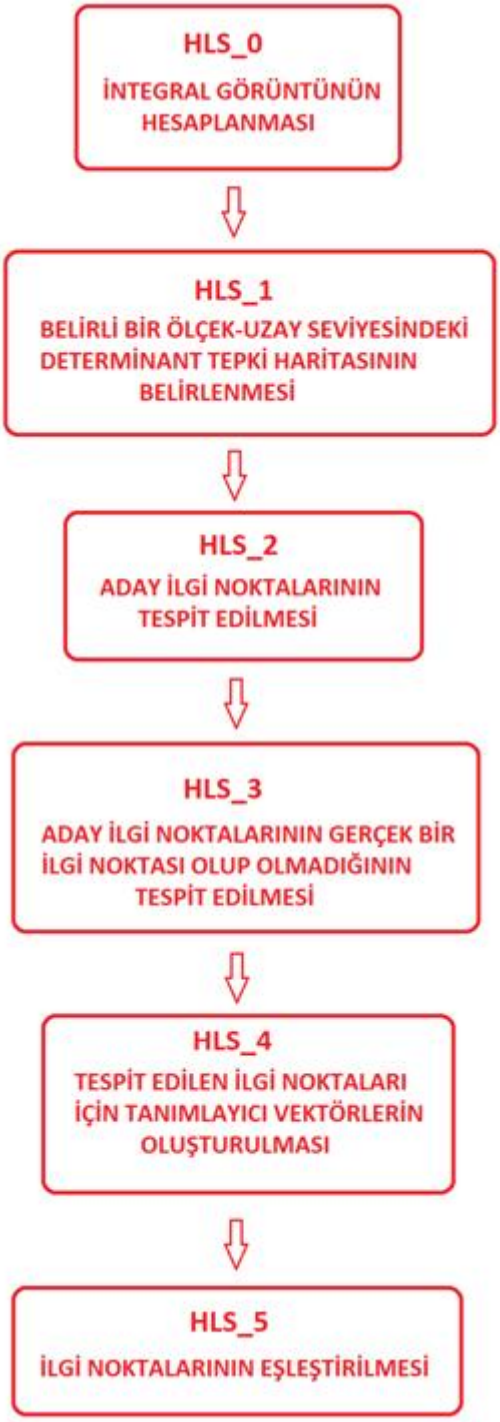
Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	169
FIFO	-	-	-	-
Instance	8	2	356	320
Memory	3	-	66	6
Multiplexer	-	-	-	264
Register	-	-	301	-
Total	11	2	723	759
Available	280	220	106400	53200
Utilization (%)	3	~0	~0	1

Şekil 4.18. HLS_5 modülünün sentezlenmesi

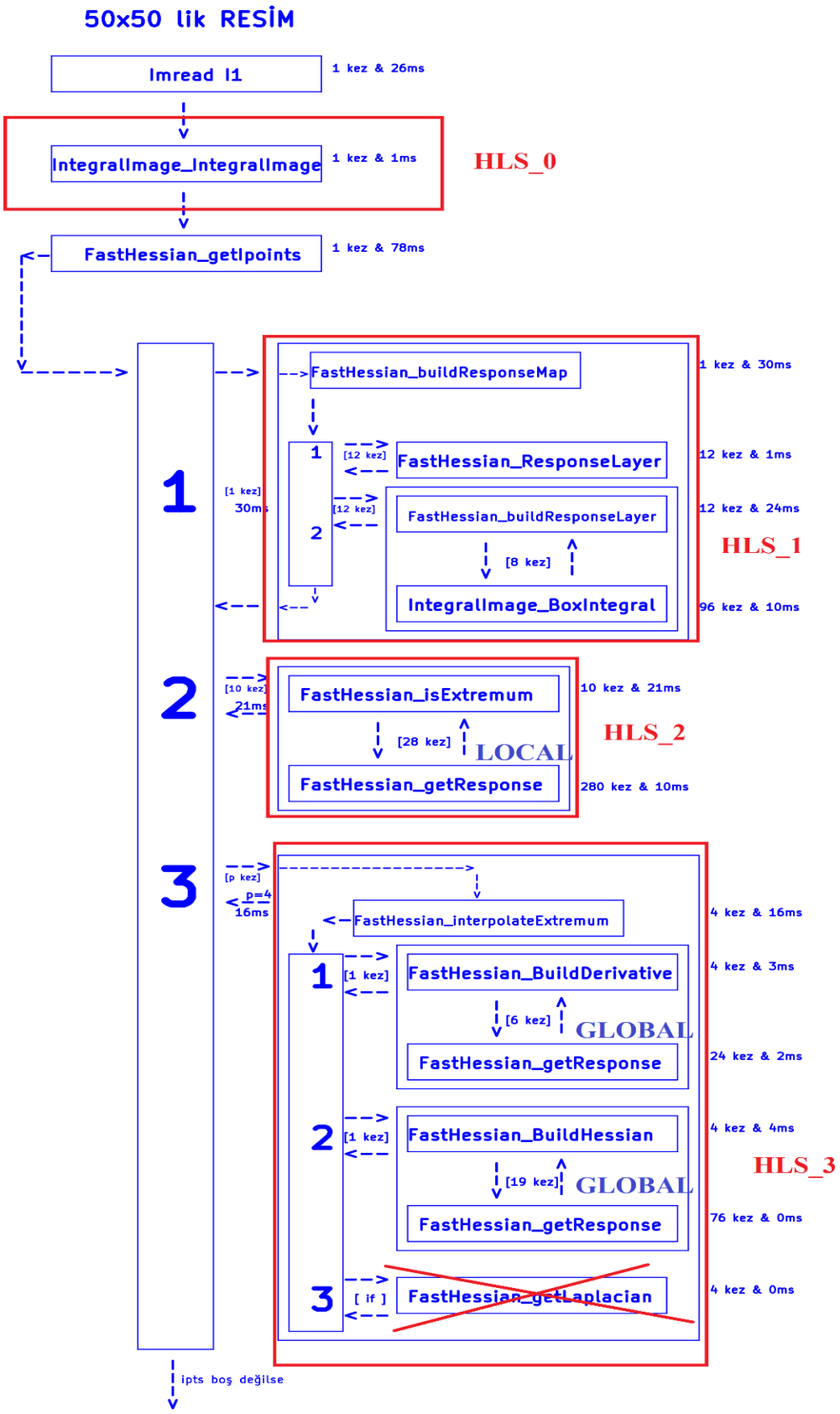
Yapılan ayarlar sonucunda sentezleme sonucunun son hali şu şekilde olmuştur:

- Çalışma süresi yaklaşık 250us olmuştur.
- BRAM kullanımı %3 e çıkmıştır.
- DSP48E kullanımı 2 olmuştur.
- FF kullanımı 305 den 723 e çıkmıştır.
- LUT kullanımı 395 den 759 a çıkmıştır.

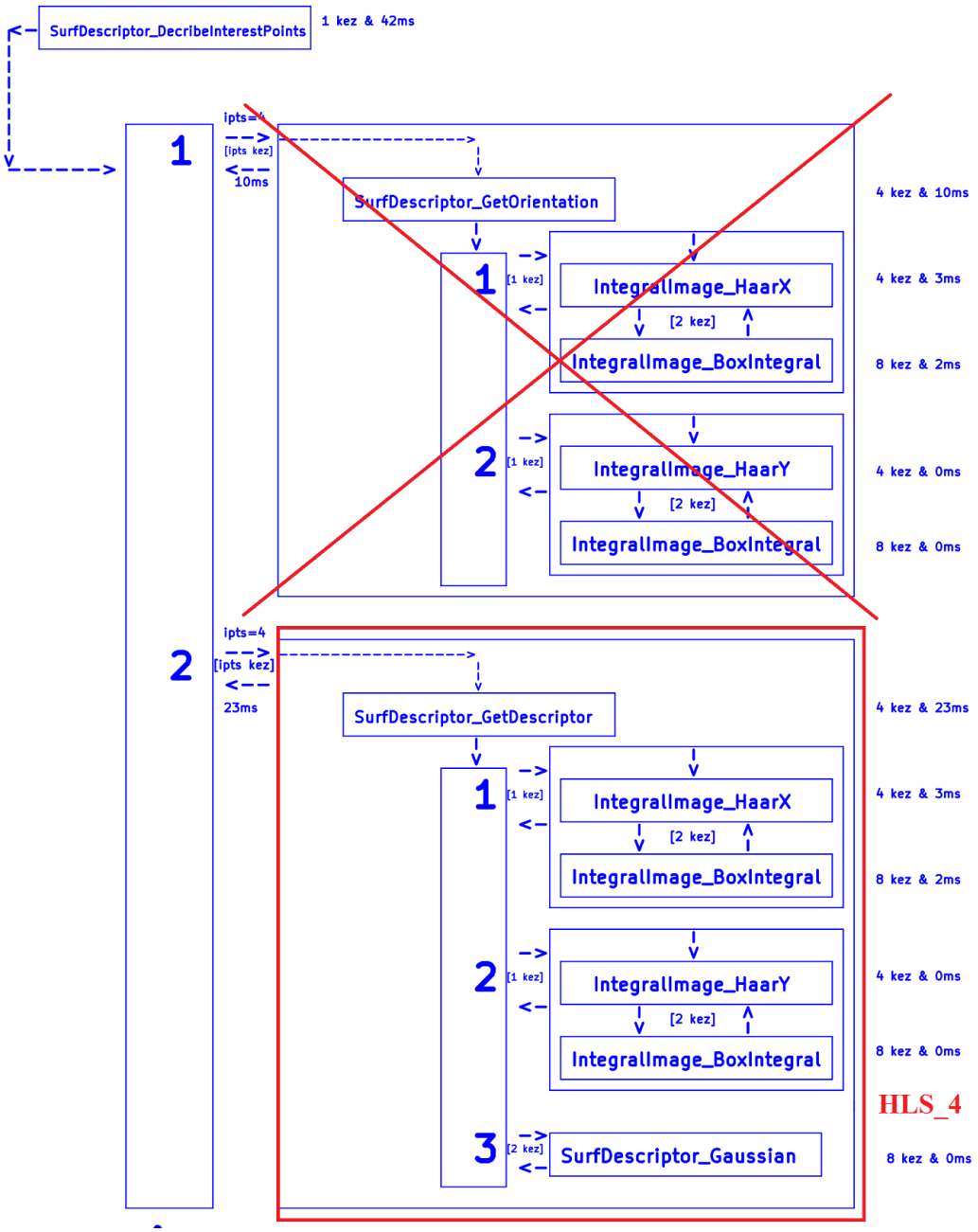
Yukarıdaki sonuçlara göre, yapılan ayarlar sonucunda çalışma süresinin azaldığı gözlemlenmiştir. Fakat donanım kullanımlarında artış olmuştur.



Şekil 4.19. SURF algoritmasının genel akış diyagramı



Şekil 4.20. OPENSURF algoritması (HLS) - 1



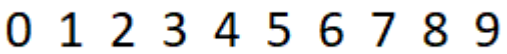
Şekil 4.21. OPENSURF algoritması (HLS) – 2

Şekil 4.22. OPENSURF algoritması (HLS) - 3

4.2. Referans Veri Setinin Oluşturulması

SURF algoritmasının yapısı gereği, referans resmine ihtiyaç duyulmaktadır. Referans görüntüler üzerinde bulunan ilgi noktaları hafızada saklanarak test görüntüleri üzerinde bulunan ilgi noktaları ile karşılaştırılarak eşleştirmeler gerçekleştirilir.

Referans veri seti için rakamlar kullanılmıştır. Her bir rakam, yazı tipi boyutu 20 ve yazı biçimi "calibri" olacak şekilde (rotasyon 0 derece) 45x80 ebatlarında resim olarak oluşturulmuştur. Referans için kullanılan resimler Şekil 4.23'de toplu halde gösterilmiştir.



Şekil 4.23. Referans görüntüleri için kullanılan rakamlar

4.3. Test Veri Setinin Oluşturulması

Parametre değişikliklerinin algoritma sonucunu nasıl değiştirdiğini gözlemlemek için test veri seti oluşturulmuştur. Ayrıca algoritmanın başarısının ölçülmesi ve optimizasyon sırasında kullanılmak üzere bu veri seti oluşturulmuştur. Optimizasyon aracı referans görüntülerini ve test görüntülerini kullanarak en uygun parametreleri belirlemeye çalışmaktadır.

Test veri seti için, her bir rakam, yazı tipi boyutu 18 ve 22, yazı biçimi "calibri", rotasyonu +10 derece, 0 derece ve -10 derece olacak şekilde ve yazı tipi boyutu 20, yazı biçimi "calibri", rotasyonu +10 derece ve -10 derece olacak şekilde 45x80 ebatlarında resim olarak oluşturulmuştur. Burada önemli olan referans olarak kullanılan resimlerin test veri seti içerisinde yer almamasıdır. Sonuç olarak her bir rakamın bir tane referans resmi, sekiz tane de test resmi oluşturulmuştur. (10 tane referans resmi, 80 tane test resmi) Şekil 4.24'de sıfır rakamı için oluşturulan test resimleri toplu halde gösterilmiştir. (test veri setinde referans olarak kullanılan resim ile aynı resim kullanılmamıştır.)

Şekil 4.24. Test veri seti örneği

4.4. Doğrulama Veri Setinin Oluşturulması

Bu bölümde OPENSURF algoritmasının başarısının ve optimizasyon ile belirlenen parametrelerin başarısının tespiti için yeni veri seti oluşturulması amaçlanmıştır. Bu veri seti test veri seti için hazırlanan 80 referans görüntüyü kullanarak rastgele olacak şekilde oluşturulur. Yani, örnek doğrulama görüntüleri 45x800 ebatlarında olacak şekilde oluşturulur. Test veri setindeki gibi benzer şekilde 10 adet rakam içermektedir. Veri setinden rastgele seçilerek her bir rakam için ikişer resim olacak şekilde 20 tane doğrulama görüntüsü oluşturulur. Bu resimlerle yapılan test sonucu bize optimize edilen ve varsayılan değerler kullanılarak elde edilen başarı oranlarını verecektir.

4.5. Test Aşaması

Referans ve test görüntüleri kullanılarak OPENSURF algoritmasının başarı oranının ölçülmesi ve bu değere göre optimizasyon algoritmasının çalıştırılması amaçlanmıştır. Her bir rakam için referans olarak kullanılmak üzere bir tane 45x80 ebatlarında resim kullanılmıştır. Referans resmine OPENSURF algoritması uygulanarak ilgi noktaları tespit edilmiştir. Daha sonra test görüntülerine OPENSURF algoritması uygulanarak başarı oranları ölçülmüştür. Test görüntüleri test veri setinden seçilecek şekilde 45x800 ebatlarında oluşturulmuştur. Yani 45x80 ebatlarında olan test resimlerinden 10 tane resmin birleştirilmesiyle oluşturulmuştur. 10 adet rakamı temsilen on eşit parçadan oluşmaktadır. Burada her bir rakam bulunacak şekilde test görüntüleri oluşturulmuştur. 45x80 ebatlarında olan referans görüntüden elde edilen ilgi noktaları, 45x800 ebatlarında olan test görüntüsünden elde edilen ilgi noktaları ile eşleştirilerek ilgili rakam tespit edilmeye çalışılır. Burada referans olarak alınan rakam ve bu rakamın test görüntüsü üzerinde hangi bölgede temsil edileceği bilgisi bilinmektedir. Dolayısı ile eşleştirilen ilgi noktalarının kaç tanesinin bu bölgeye denk geldiği tespit edilir. Bu bilgi otomatik olarak güncellendiği için ilgili rakamın doğruluk oranını temsil eden değişken de güncellenerek, tüm görüntülerden gelen bu değerlerin bilgisi optimizasyon algoritmasının girişine aktarılır. Böylece optimizasyon algoritması bu sayıyı arttırmak için işlemlerini yapar.

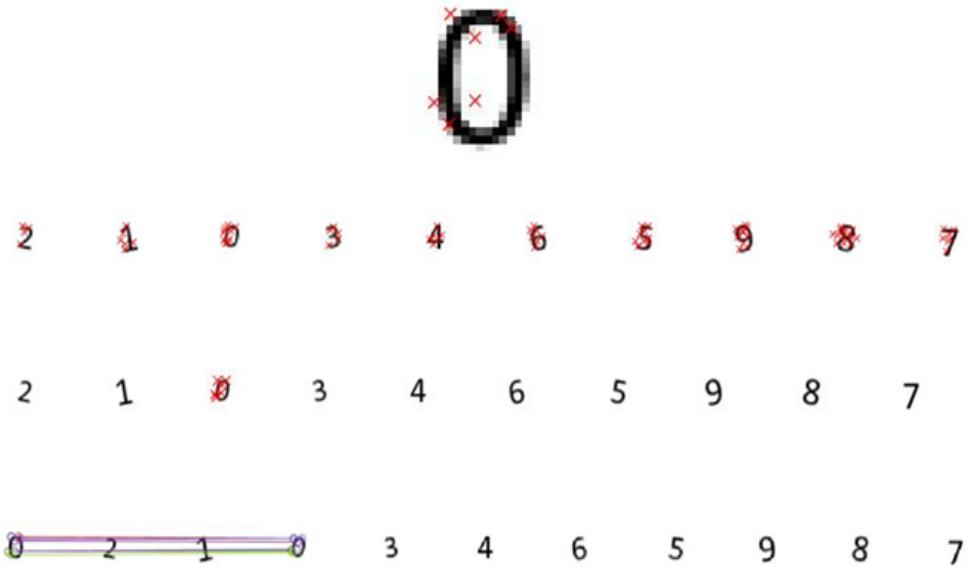
Test resimleri için, test veri seti için hazırlanan 45x80 ebatlarındaki resimler kullanılarak, her bir rakam için olmak üzere toplam 6 adet 45x800 ebatlarında resim oluşturulmuştur. Yani tüm rakamlar için toplamda 60 adet 45x800 ebatlarında test görüntüsü oluşturulmuştur. Şekil 4.25'de örnek bir test görüntüsü gösterilmiştir. OPENSURF algoritması ile eşleşen ilgi noktalarının koordinat bilgilerine ulaşılabildiği için, hangi ilgi noktasının hangi koordinatlarda olduğu tespit edilebilmektedir. Test görüntüsü üzerinde aranan rakamın hangi koordinat bölgelerinde olduğu da bilindiği için, bu bilgilerle ilgili rakamın doğru olarak tespit edilip edilmediği otomatik olarak tespit edilebilmektedir. Burada referans görüntü üzerinde tespit edilen ilgi noktalarının tamamı, test görüntüsü üzerinde ilgili rakamı temsil eden bölge içerisinde tespit edilirse başarı oranı %100 olarak hesaplanmış olur.

Optimizasyon algoritması kullanılarak her bir referans görüntünün, farklı boyut ve dönmelerdeki on rakamı da içeren test görüntüsü ile doğru eşleşme sayısını

maksimum yapmak amaçlanmıştır. Bu sayıyı bulmak için, her test sonunda referans resminde bulunan ilgi noktalarının en az %50 sinin test görüntüsünde ilgili bölgede olup olmadığına bakılır. İlgili bölgede yeteri kadar ilgi noktası bulunduysa başarı sayacı 1 artırılır. Toplamda 60 test görüntüsü için, her bir görüntünün ilgili bölgesinde en az %50 ilgi noktası bulunursa, başarı sayacı 60 olacaktır ve test başarısı %100 (60/60) olacaktır. Optimizasyon algoritması bu başarı sayacını giriş olarak alıp işlemlerini yürütür.

Şekil 4.25. Örnek test görüntüsü (45x800)

Şekil 4.26'da sıfır rakamı için yapılan testlerden biri gösterilmiştir. En üstte referans olarak kullanılan sıfır rakamı için bulunan ilgi noktaları gösterilmiştir (ilgi noktalarının görünmesi için resmin ölçeği değiştirilmiştir). İkinci sırada ise 45x800 ebatlarında oluşturulan test görüntülerinden birinde bulunan ilgi noktaları gösterilmiştir. Üçüncü sırada ise referans görüntüde bulunan ilgi noktaları ile test görüntüsünde bulunan ilgi noktalarından eşleşenler gösterilmiştir. Dördüncü sırada ise referans görüntü ve test görüntüsünde eşleşen ilgi noktaları çizim ile eşleştirilerek gösterilmiştir.



Şekil 4.26. Sıfır rakamının test sonuçları

OPENSURF algoritmasında varsayılan olarak oktav – 1 değerleri 9 – 15 – 21 – 27, eşik değeri ise 0.0001 olarak oluşturulmuştur. Burada eşik değeri aday ilgi noktalarının hangilerinin ilgi noktası olarak kabul edileceğini temsil eder. Oktav değerleri ise kullanılacak kutu filtre boyutlarını temsil eder. Yani oktav – 1 olarak varsayılan değerler kullanıldığında, 9x9, 15x15, 21x21, 27x27 ebatlarında kutu filtrelerle işlemler gerçekleştirilir. Optimizasyon algoritması ile bu filtrelerin boyutlarındaki değişimlerin başarı oranına olan etkisinin tespit edilmesi amaçlanmıştır.

4.6. Testlerin Gerçekleştirilmesi

Testler ilk olarak Matlab ile varsayılan parametre değerleri kullanılarak yapılmıştır. Bu değerler oktav – 1 için 9, 15, 21, 27, eşik değeri için 0.0001'dir. Çizelge 4.3'de gerçekleştirilen test sonuçları gösterilmiştir. Burada, her bir rakam için ayrı olarak test başarısı ve genel testin başarısı gösterilmiştir. Buna göre yapılan toplam 60 test içerisinde %50 (tespit edilen ilgi noktalarının sayısına göre) den daha başarılı olan test sayısı 18 olarak tespit edilmiştir. Buradaki yüzde değerleri ilgili rakamın doğru bulunma başarısını temsil ettiği için, %50 eşik değeri, ilgili rakamın en az %50 doğrulukta bulunduğunu ifade eder. Sonuç olarak varsayılan parametreler kullanılarak matlab ile gerçekleştirilen test sonucuna göre elde edilen başarı oranı %30 olmuştur.

Çizelge 4.3. Varsayılan parametreler ile yapılan matlab testi – 1

Oktav - 1 [9 , 15 , 21 , 27] , Eşik Değeri = 0.0001 olarak seçildiğinde:								
Rakamlar	Referans Görüntüsünde Bulunan Özellik Sayısı	Test Görüntülerindeki Rakamların Bulunma Başarısı (Her Rakam İçin Ayrı Olarak %)						%50 den daha başarılı olan test sayısı
		Test - 1	Test - 2	Test - 3	Test - 4	Test - 5	Test - 6	
0	8	25,00	62,50	50,00	37,50	37,50	25,00	1
1	2	100,00	50,00	0,00	0,00	0,00	50,00	1
2	6	33,33	50,00	33,33	50,00	50,00	83,33	1
3	5	0,00	60,00	40,00	20,00	20,00	60,00	2
4	4	100,00	50,00	75,00	25,00	25,00	75,00	3
5	4	75,00	50,00	50,00	25,00	25,00	50,00	1
6	3	66,67	66,67	33,33	33,33	33,33	66,67	3
7	4	50,00	50,00	75,00	75,00	100,00	100,00	4
8	3	0,00	33,33	100,00	33,33	33,33	0,00	1
9	2	100,00	50,00	50,00	50,00	50,00	50,00	1
*** 60 test içerisinde %50 den daha başarılı olan test sayısı 18 olmuştur. Sonuç = 18/60 => **%30**								**18**

Çizelge 4.4'de başarı oranı %70'den daha başarılı olanların kabul edildiği durumu göstermektedir. Buna göre yapılan 60 test içerisinde en az %70 doğruluk

oranında karakterin bulunduğu test sayısı 12 olarak tespit edilmiştir. Sonuç olarak başarı oranı %20 olmuştur.

Çizelge 4.4. Varsayılan parametreler ile yapılan matlab testi – 2

Oktav - 1 [9 , 15 , 21 , 27] , Eşik Değeri = 0.0001 olarak seçildiğinde:								
Rakamlar	Referans Görüntüsünde Bulunan Özellik Sayısı	Test Görüntülerindeki Rakamların Bulunma Başarısı (Her Rakam İçin Ayrı Olarak %)						%70 den daha başarılı olan test sayısı
		Test - 1	Test - 2	Test - 3	Test - 4	Test - 5	Test - 6	
0	8	25,00	62,50	50,00	37,50	37,50	25,00	0
1	2	100,00	50,00	0,00	0,00	0,00	50,00	1
2	6	33,33	50,00	33,33	50,00	50,00	83,33	1
3	5	0,00	60,00	40,00	20,00	20,00	60,00	0
4	4	100,00	50,00	75,00	25,00	25,00	75,00	3
5	4	75,00	50,00	50,00	25,00	25,00	50,00	1
6	3	66,67	66,67	33,33	33,33	33,33	66,67	0
7	4	50,00	50,00	75,00	75,00	100,00	100,00	4
8	3	0,00	33,33	100,00	33,33	33,33	0,00	1
9	2	100,00	50,00	50,00	50,00	50,00	50,00	1
*** 60 test içerisinde %70 den daha başarılı olan test sayısı 12 olmuştur. Sonuç = 12/60 => **%20**								**12**

Başarı oranlarını arttırmak için Oktav – 1 ve eşik değerlerinin daha uygun değerlerinin tespit edilmesi amaçlanmıştır. Bunun için matlab optimizasyon araçları içerisinde bulunan genetik algoritma aracı kullanılmıştır. Optimizasyon algoritması, ilk olarak %50 (tespit edilen ilgi noktalarının sayısına göre)'den daha başarılı olan test sayısını artıracak şekilde çalıştırılmıştır. Buna göre bulunan Oktav – 1 değerleri 4, 9, 18, 23 ve eşik değeri 0.007 olarak bulunmuştur. Çizelge 4.5'de test sonuçları gösterilmiştir. Buna göre yapılan 60 test içerisinde %50'den daha başarılı olan test sayısı 59 olmuştur. Sonuç olarak başarı oranı %98.3 olarak hesaplanmıştır. Çizelge 4.6'da ise %70'den daha başarılı olan testlerin sayısı 54 olarak tespit edilmiştir. Buradaki başarı oranı ise %90 olarak hesaplanmıştır. Optimizasyon algoritması %70'den daha başarılı olan test sayısını artıracak şekilde çalıştırıldığında tespit edilen Oktav – 1 değerleri 4, 9, 18, 19 ve eşik değeri 0.0099 olmuştur. Çizelge 4.7 ve Çizelge 4.8'de sonuçlar gösterilmiştir.

Çizelge 4.5. Optimizasyon algoritmasının sonuçları ile yapılan matlab testi – 1

Oktav - 1 [4 , 9 , 18 , 23] , Eşik Değeri = 0.007 olarak seçildiğinde: NOT: Bu değerler optimizasyon algoritmasının bulduğu değerlerdir.								
Rakamlar	Referans Görüntüsünde Bulunan Özellik Sayısı	Test Görüntülerindeki Rakamların Bulunma Başarısı (Her Rakam İçin Ayrı Olarak %)						%50 den daha başarılı olan test sayısı
		Test - 1	Test - 2	Test - 3	Test - 4	Test - 5	Test - 6	
0	4	100,00	100,00	100,00	100,00	100,00	100,00	6
1	5	100,00	80,00	100,00	80,00	80,00	60,00	6
2	6	83,33	100,00	66,67	83,33	83,33	83,33	6
3	7	57,14	42,86	85,71	57,14	57,14	85,71	5
4	5	80,00	80,00	100,00	100,00	100,00	100,00	6
5	7	71,43	71,43	85,71	71,43	71,43	71,43	6
6	5	80,00	80,00	100,00	100,00	100,00	100,00	6
7	4	100,00	100,00	75,00	100,00	100,00	100,00	6
8	10	90,00	80,00	100,00	100,00	100,00	100,00	6
9	7	100,00	71,43	85,71	100,00	100,00	85,71	6
*** 60 test içerisinde %50 den daha başarılı olan test sayısı **59** olmuştur. Sonuç = 59/60 => **%98,3**								**59**

Çizelge 4.6. Optimizasyon algoritmasının sonuçları ile yapılan matlab testi – 2

Oktav - 1 [4 , 9 , 18 , 23] , Eşik Değeri = 0.007 olarak seçildiğinde: NOT: Bu değerler optimizasyon algoritmasının bulduğu değerlerdir.								
Rakamlar	Referans Görüntüsünde Bulunan Özellik Sayısı	Test Görüntülerindeki Rakamların Bulunma Başarısı (Her Rakam İçin Ayrı Olarak %)						%70 den daha başarılı olan test sayısı
		Test - 1	Test - 2	Test - 3	Test - 4	Test - 5	Test - 6	
0	4	100,00	100,00	100,00	100,00	100,00	100,00	6
1	5	100,00	80,00	100,00	80,00	80,00	60,00	5
2	6	83,33	100,00	66,67	83,33	83,33	83,33	5
3	7	57,14	42,86	85,71	57,14	57,14	85,71	2
4	5	80,00	80,00	100,00	100,00	100,00	100,00	6
5	7	71,43	71,43	85,71	71,43	71,43	71,43	6
6	5	80,00	80,00	100,00	100,00	100,00	100,00	6
7	4	100,00	100,00	75,00	100,00	100,00	100,00	6
8	10	90,00	80,00	100,00	100,00	100,00	100,00	6
9	7	100,00	71,43	85,71	100,00	100,00	85,71	6
*** 60 test içerisinde %70 den daha başarılı olan test sayısı **54** olmuştur. Sonuç = 54/60 => **%90**								**54**

Çizelge 4.7. Optimizasyon algoritmasının sonuçları ile yapılan matlab testi – 3

Oktav - 1 [4 , 9 , 18 , 19] , Eşik Değeri = 0.0099 olarak seçildiğinde: NOT: Bu değerler optimizasyon algoritmasının bulduğu değerlerdir.								
Rakamlar	Referans Görüntüsünde Bulunan Özellik Sayısı	Test Görüntülerindeki Rakamların Bulunma Başarısı (Her Rakam İçin Ayrı Olarak %)						%50 den daha başarılı olan test sayısı
		Test - 1	Test - 2	Test - 3	Test - 4	Test - 5	Test - 6	
0	2	100,00	100,00	100,00	100,00	100,00	100,00	6
1	4	100,00	75,00	100,00	75,00	75,00	75,00	6
2	4	100,00	100,00	75,00	100,00	100,00	75,00	6
3	4	50,00	50,00	100,00	75,00	75,00	100,00	4
4	5	80,00	80,00	100,00	100,00	100,00	100,00	6
5	7	71,43	71,43	85,71	71,43	71,43	71,43	6
6	4	100,00	75,00	100,00	100,00	100,00	100,00	6
7	4	100,00	100,00	75,00	100,00	100,00	100,00	6
8	9	88,89	77,78	100,00	100,00	100,00	100,00	6
9	5	100,00	80,00	80,00	100,00	100,00	100,00	6
*** 60 test içerisinde %50 den daha başarılı olan test sayısı **58** olmuştur. Sonuç = 58/60 => **%96,67**								**58**

Çizelge 4.8. Optimizasyon algoritmasının sonuçları ile yapılan matlab testi – 4

Oktav - 1 [4 , 9 , 18 , 19] , Eşik Değeri = 0.0099 olarak seçildiğinde: NOT: Bu değerler optimizasyon algoritmasının bulduğu değerlerdir.								
Rakamlar	Referans Görüntüsünde Bulunan Özellik Sayısı	Test Görüntülerindeki Rakamların Bulunma Başarısı (Her Rakam İçin Ayrı Olarak %)						%70 den daha başarılı olan test sayısı
		Test - 1	Test - 2	Test - 3	Test - 4	Test - 5	Test - 6	
0	2	100,00	100,00	100,00	100,00	100,00	100,00	6
1	4	100,00	75,00	100,00	75,00	75,00	75,00	6
2	4	100,00	100,00	75,00	100,00	100,00	75,00	6
3	4	50,00	50,00	100,00	75,00	75,00	100,00	4
4	5	80,00	80,00	100,00	100,00	100,00	100,00	6
5	7	71,43	71,43	85,71	71,43	71,43	71,43	6
6	4	100,00	75,00	100,00	100,00	100,00	100,00	6
7	4	100,00	100,00	75,00	100,00	100,00	100,00	6
8	9	88,89	77,78	100,00	100,00	100,00	100,00	6
9	5	100,00	80,00	80,00	100,00	100,00	100,00	6
*** 60 test içerisinde %70 den daha başarılı olan test sayısı **58** olmuştur. Sonuç = 58/60 => **%96,67**								**58**

Optimizasyon algoritması ile tespit edilen parametre değerlerini doğrulamak ve sistemin başarısını tespit etmek için test veri setinden rastgele seçim ile oluşturulan 45x800 ebatlarında resimler kullanılarak testler gerçekleştirilmiştir (45x80 ebatlarındaki resimlerden seçilerek oluşturuldu). Toplam 200 tane doğrulama verisi için elde edilen sonuçlar Çizelge 4.9'da gösterilmiştir. Buna göre optimizasyon algoritması ile tespit edilen parametre değerlerinin sistemin başarı oranını önemli ölçüde artırdığı gözlemlenmiştir.

Çizelge 4.9. Doğrulama verisi sonuçları

Oktav - 1 [9 , 15 , 21 , 27] , Eşik Değeri = 0.0001 olarak seçildiğinde:		
Toplam Doğrulama Resim Sayısı	%50(ilgi noktası sayısına göre) den daha başarılı olan test sayısı	%70(ilgi noktası sayısına göre) den daha başarılı olan test sayısı
200	60	38
	30%	**19%**
Oktav - 1 [4 , 9 , 18 , 23] , Eşik Değeri = 0.007 olarak seçildiğinde: NOT: Bu değerler optimizasyon algoritmasının bulduğu değerlerdir.		
Toplam Doğrulama Resim Sayısı	%50(ilgi noktası sayısına göre) den daha başarılı olan test sayısı	%70(ilgi noktası sayısına göre) den daha başarılı olan test sayısı
200	192	168
	96%	**84%**
Oktav - 1 [4 , 9 , 18 , 19] , Eşik Değeri = 0.0099 olarak seçildiğinde: NOT: Bu değerler optimizasyon algoritmasının bulduğu değerlerdir.		
Toplam Doğrulama Resim Sayısı	%50(ilgi noktası sayısına göre) den daha başarılı olan test sayısı	%70(ilgi noktası sayısına göre) den daha başarılı olan test sayısı
200	193	189
	96,5%	**94,5%**

Çizelge 4.10. Varsayılan parametreler ile yapılan HLS testi – 1

Oktav - 1 [9 , 15 , 21 , 27] , Eşik Değeri = 0.0001 olarak seçildiğinde:							
Rakamlar	Referans Görüntüsünde Bulunan Özellik Sayısı	TEST - 1			TEST - 2		
		Test Görüntüsünde Bulunan Özellik Sayısı	Başarılı Olarak Bulunan Özellik Sayısı	Başarı Oranı %	Test Görüntüsünde Bulunan Özellik Sayısı	Başarılı Olarak Bulunan Özellik Sayısı	Başarı Oranı %
0	8	4	3	37,50	5	3	37,50
1	2	2	1	50,00	7	0	0,00
2	6	3	3	50,00	5	4	66,67
3	7	5	4	57,14	5	2	28,57
4	5	3	3	60,00	9	3	60,00
5	4	5	4	100,00	4	4	100,00
6	3	4	2	66,67	2	2	66,67
7	6	7	4	66,67	3	3	50,00
8	5	5	3	60,00	4	3	60,00
9	3	5	1	33,33	4	2	66,67
				58%			**54%**

Çizelge 4.11. Optimize edilen parametreler ile yapılan HLS testi - 1

Oktav - 1 [4 , 9 , 18 , 23] , Eşik Değeri = 0.007 olarak seçildiğinde:							
Rakamlar	Referans Görüntüsünde Bulunan Özellik Sayısı	TEST - 1			TEST - 2		
		Test Görüntüsünde Bulunan Özellik Sayısı	Başarılı Olarak Bulunan Özellik Sayısı	Başarı Oranı %	Test Görüntüsünde Bulunan Özellik Sayısı	Başarılı Olarak Bulunan Özellik Sayısı	Başarı Oranı %
0	4	5	4	100,00	3	2	50,00
1	5	3	3	60,00	5	5	100,00
2	6	3	3	50,00	6	4	66,67
3	8	6	6	75,00	10	7	87,50
4	5	5	5	100,00	4	4	80,00
5	7	7	7	100,00	8	7	100,00
6	5	9	5	100,00	4	4	80,00
7	4	4	4	100,00	3	3	75,00
8	10	10	8	80,00	11	8	80,00
9	7	8	5	71,43	4	4	57,14
				84%			**78%**

Çizelge 4.12. Optimize edilen parametreler ile yapılan HLS testi – 2

Oktav - 1 [4 , 9 , 18 , 19] , Eşik Değeri = 0.0099 olarak seçildiğinde:							
Rakamlar	Referans Görüntüsünde Bulunan Özellik Sayısı	TEST - 1			TEST - 2		
		Test Görüntüsünde Bulunan Özellik Sayısı	Başarılı Olarak Bulunan Özellik Sayısı	Başarı Oranı %	Test Görüntüsünde Bulunan Özellik Sayısı	Başarılı Olarak Bulunan Özellik Sayısı	Başarı Oranı %
0	2	3	2	100,00	2	2	100,00
1	4	2	2	50,00	3	3	75,00
2	4	2	2	50,00	3	2	50,00
3	5	4	4	80,00	6	5	100,00
4	5	5	5	100,00	3	3	60,00
5	7	6	6	85,71	7	7	100,00
6	4	7	4	100,00	3	3	75,00
7	4	4	4	100,00	3	3	75,00
8	9	10	8	88,89	9	9	100,00
9	5	5	4	80,00	3	3	60,00
				83%			**80%**

Matlab optimizasyon aracı ile belirlenen uygun parametre değerlerinin HLS ile yapılan testlerdeki etkisi Çizelge 4.10, 4.11, 4.12'de gösterilmiştir. Burada yapılan testlerde 45x80 ebatlarında referans resim ve 45x80 ebatlarında test resmi kullanılmıştır. Örnek olarak sıfır rakamı punto 20, rotasyon 0 olacak şekilde referans olarak kabul edilirken, birinci testte punto 22, rotasyon 10, ikinci testte punto 18, rotasyon 0 olarak seçilmiştir (Ahmad R. ve arkadaşlarının yapmış olduğu çalışmada kullanılan test resimleri arasındaki büyüklük farkı da 2 puntodur). Daha sonra HLS ile bulunan özellikler matlab üzerinden karşılaştırılarak doğru özellik sayıları tespit edilmiştir. Yapılan test sonuçlarına göre varsayılan parametre değerleri ile elde edilen başarı ortalaması %56 iken, optimizasyon ile hesaplanan parametre değerleri kullanılarak elde edilen başarı ortalaması %81 olmuştur.

5. TARTIŞMA

SURF algoritmasının yüksek çözünürlüklü bir resmin tamamına uygulanması ilgi noktalarının tespiti ve eşleştirilmesi açısından fazla çalışma zamanı gerektirir. Bunun yerine daha düşük çözünürlükte resimlerle veya yüksek çözünürlüklü bir resmin ilgili bölgesi ile çalışmak daha uygundur. Bunun için yapılan testlerde ilgi noktalarının sayısının çok az olmadığı bir durum olan 45x80 ebatlarında bir resim ile çalışılmasına karar verilmiştir (Çizelge 4.1).

SURF algoritmasının HLS ile gerçeklenmesi amaçlanmış ve bu kapsamda algoritma modüller halinde bölünmüştür (Şekil 4.20, Şekil 4.21, Şekil 4.22). Genel akış diyagramı Şekil 4.19'da gösterilmiştir. SURF algoritması (U-SURF) ± 15 dereceye kadar oluşan dönmelerden etkilenmediği için, algoritma içerisinde yer alan dönme kontrolü ile ilgili kısımlar çıkarılarak algoritmanın daha hızlı çalışması sağlanmıştır. Yapılacak çalışmada dönme miktarının 15 dereceden daha fazla olabileceği varsayılıyorsa, dönme ile ilgili kısımların çıkarılması uygun olmayacaktır.

SURF algoritmasının gerçeklenmesi için veri tipi olarak kayan-nokta (floating-point) yerine sabit-nokta (fixed-point) seçilmiştir. Böylece kullanılan donanım miktarının azaltılması ve sistemin hızlanması sağlanmıştır (Φαλιάγκας, K. yaptığı çalışmasında veri tipi olarak kayan-nokta (floating-point) seçmiştir). Yapılan testler ile saklayıcıların ondalık kısımlarının 12bit veya 16bit ile temsil edilmesine karar verilmiştir. Bu durumun kullanılan donanım miktarını ve çalışma süresini önemli ölçüde iyileştirdiği gözlemlenmiştir.

SURF algoritmasının başarısına önemli derecede etki eden iki önemli parametre belirlenmiştir (eşik değeri ve oktav değerleri). Eşik değeri, bulunan aday ilgi noktalarının gerçek ilgi noktası olarak kabul edilip edilmeyeceğini belirlerken, oktav değerleri ise kullanılan kutu filtrelerinin boyutlarını belirler. Bu iki parametrenin başarı oranı üzerinde önemli bir etkisi olduğu gözlemlenmiştir.

Eşik değeri ve oktav değerlerinin en uygun değerlerini tespit edebilmek için Genetik Algoritma (GA) kullanılmıştır. Varsayılan değerler ile optimize edilen değerlerin başarı oranları karşılaştırıldığında önemli bir oranda artış olmuştur. Örnek olarak, varsayılan değerler ile yapılan test sonucunda ilgi noktalarının sayısının en az %50'sinin bulunduğu test başarı oranı %30 iken, optimize edilen değerler ile yapılan test sonucunda test başarı oranı %98.3 olmuştur. Daha sonra yapılacak çalışmalar için probleme göre optimizasyon işleminin tekrarlanarak en iyi değerlerin bulunması öngörülmektedir.

Optimize edilen parametreler HLS de kullanılarak HLS test başarı oranlarının da artırılması amaçlanmıştır. Örnek olarak varsayılan değerler ile elde edilen başarı oranı %56 iken optimize edilen değerler ile elde edilen başarı oranı %81 olmuştur.

Sistemin test edilmesi için rakamlardan oluşturulan resimler kullanılmıştır. Dönme ve büyüklükten bağımsız olarak rakamların tespit edilmesi amaçlanmıştır. Bunun için her bir rakamın üç farklı büyüklük ve üç farklı derece ile dönmüş hali kullanılmıştır. Önerilen yöntem ile yüksek tanıma oranları elde edilmiştir.

6. SONUÇLAR

Bu çalışmada, SURF algoritmasının HLS ile oluşturularak testlerinin yapılması amaçlanmıştır. FPGA üzerinde yazılım geliştirmek için verilog ve VHDL gibi donanım tanımlama dilleri kullanılmaktadır. HLS ile donanım tanımlama dilleri yerine C,C++ dilleri ile oluşturulan kodlar kullanılarak FPGA için yazılım geliştirilebilmektedir. Çalışmada C dili kullanılarak, donanım tanımlama dillerindeki karmaşıklık ve yazılım oluşturma süresinin azaltılması amaçlanmıştır. Ayrıca HLS ile oluşturulan modüller yine HLS araçları arasında yer alan direktifler kullanılarak donanım çalışması optimize edilebilmektedir. Bu direktifler ile donanım kullanımının ve yazılım çalışma süresinin azaltılması amaçlanmıştır.

SURF algoritmasında kullanılan işlemler için tanımlanan saklayıcıların veri tipi sabit nokta (fixed-point) seçilerek kullanılan hafıza boyutunun azaltılması ve yapılacak işlem sürelerinin azaltılarak algoritmanın daha hızlı çalışabilmesi sağlanmıştır. Veri tipi boyutları ise matlab ile yapılan test sonuçlarına göre belirlenerek veri kaybının daha az ve test başarısının çok düşmediği değerlere göre belirlenmiştir.

SURF algoritmasının başarısı rakamlar ile oluşturulan veri seti üzerinde test edilmiştir. Bu kapsamda 45x80 ebatlarında resimler kullanılarak testler yapılmıştır.

Algoritmada kullanılan parametrelerin genetik algoritma ile optimize edilerek en uygun değerlerinin tespit edilmesi amaçlanmıştır. Bunun için oktav – 1 değerleri (SURF algoritmasında kullanılan filtre boyutlarını temsil eder) ve eşik değerinin optimize edilmesi sağlanmıştır. Optimize algoritmasına, olabilecek değer aralıkları verilerek en uygun değerler tespit edilmiştir ve SURF algoritmasında bu değerler kullanılmıştır. Optimize edilen parametrelerin değerleri kullanılarak başarı oranı önemli ölçüde artırılmıştır.

Bölüm 4'te HLS'de oluşturulan modüllerin sonuçları detaylı olarak açıklanmıştır. Burada, yapılan düzenlemelerle çalışma süresi ve kaynak kullanımının azaltıldığı gösterilmiştir. Bölüm 5'de ise bunlara ilave olarak SURF algoritmasında kullanılan eşik değeri ve oktav-1 değerlerinin genetik algoritması ile optimizasyonu yapılarak başarı oranının daha yüksek olduğu değerlerin tespit edilmesi ve bulunan değerlerin sonuçlar üzerindeki etkisi gösterilmiştir. Başarı oranının önemli ölçüde arttığı tespit edilmiştir.

Bu çalışmada, HLS kullanımı ile C dilinde SURF algoritmasının gerçekleştirilmesi, veri tipi olarak sabit-nokta (fixed-point) kullanılması ve parametrelerin genetik algoritması ile optimize edilmesi amaçlanmıştır. Yapılan çalışmada, önerilen yöntem kullanılarak büyüklük ve dönmeden (rotation) bağımsız bir rakam tanıma uygulaması gerçekleştirilmiştir. Önerilen yöntem nesne tanıma, el yazısı karakter tanıma gibi farklı uygulamalar için de kullanılabilir. Yapılacak uygulamaya göre başarı oranının yüksek olabilmesi için SURF algoritmasında kullanılan parametrelerin (eşik değeri, oktav-1 değerleri) tekrar optimize edilerek kullanılması önerilmiştir. Ayrıca, literatürde bulunan hazır görüntü setleri kullanılarak ve GA'dan farklı bir optimizasyon kullanılarak da sistemin başarı oranı tespit edilebilir.

7. KAYNAKLAR

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. SURF:Speeded up robust features. ECCV. pp. 404-417.

Lindeberg, T. 1998. Feature detection with automatic scale selection. International Journal of Computer Vision. 30 (2): 79-116.

Lowe, D. 2004. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision. 60: 91-110.

Mikolajczyk, K. and Schmid, C. 2002. An affine invariant interest point detector. European Conference on Computer Vision. pp.128-142, Denmark.

Ke, Y. and Sukthankar, R. 2004. PCA-SIFT: A more distinctive representation for local image descriptors. CVPR. (2): 506-513.

Tuytelaars, T. and Van Gool, L. 2000. Wide baseline stereo based on local, affinely invariant regions. BMVC. pp. 412-422.

Matas, J., Chum, O., M., U. and Pajdla, T. 2002. Robust wide baseline stereo from maximally stable extremal regions. BMVC. pp.384-393.

Harris, C. and Stephens, M. 1988. A combined corner and edge detector. Proceedings of the Alvey Vision Conference. pp. 147-151.

Lowe, D. 1999. Object recognition from local scale-invariant features. ICCV.

Kadir, T. and Brady, M. 2001. Scale, saliency and image description. IJCV. 45 (2): 83-105.

Jurie, F. and Schmid, C. 2004. Scale-invariant shape features for recognition of object categories. CVPR, volume II. pp. 90-96.

Mikolajczyk, K. and Schmid, C. 2005. A performance evaluation of local descriptors. PAMI. 27: 1615-1630.

Alpaslan, N. 2013. Gradyan tabanlı heterojen öznitelik çıkarma yöntemlerine yeni yaklaşımlar. Yüksek lisans tezi, İnönü Üniversitesi, Malatya, 78+ix s.

Viola, P. and Jones, M. 2001. Rapid object detection using a boosted cascade of simple features. CVPR (1):511-518.

Kutluk, S. 2012.Yerel öznitelikler ile mamografi görüntülerinde doku yoğunluğunun sınıflandırılması. Yüksek lisans tezi, İstanbul Teknik Üniversitesi, İstanbul, 49 s.

Bouris D. and et al. 2010. Fast and efficient FPGA-based feature detection employing the SURF algorithm. FCCM 10:3-10.

Viola, P. and Jones, Michael. 2001. "Rapid object detection using a boosted cascade of simple features", Computer Vision and Pattern Recognition, 2001.

Lindeberg, T. 1996. Scale-space: A framework for handling image structures at multiple scales.. http://www.nada.kth.se/~tony/cern-review/cern-html/ [Son erişim tarihi: 01.07.1997].

Witkin, A. P. 1983. "Scale-space filtering", Proc. 8th Int. Joint Conf. Art. Intell., pp.1019-1022, Germany.

Brown, M. and Lowe, D.G. 2002. “Invariant features from interest point groups”. British Machin Vision Conference. pp.656-665.

Evans, C. 2009. Notes on the OPENSURF Library. “http://www.cs.bris.ac.uk/publications/papers/2000970.pdf”

Kroon, D.J. 2010. OpenSURF (including image warp). https://www.mathworks.com/matlabcentral/fileexchange/28300-opensurf--including-image-warp- [Son erişim tarihi: 06.09.2010].

Xilinx. 2013. Vivado Design Suite User Guide on High Level Synthesis, UG902 (v2013.2).

Baguma, G. 2014. High level synthesis of FPGA-based digital filters. Student thesis, Uppsala University, Swedish, 75 p.

Yıldırım, Ö., Erişti, B., Erişti, H. ve Demir, Y. 2012. Gerçek zamanlı akım ve gerilim sinyallerinin izlenmesi için FPGA tabanlı bir sistem tasarımı. ELOCO ‘2012 Elektrik-Elektronik ve Bilgisayar Mühendisliği Sempozyumu. ss. 654-658. Bursa.

Yeniay, Ö. 2001. “An Overview of Genetic Algorithms”. Anadolu Üniversitesi Bilim ve Teknoloji Dergisi. Cilt: 2, Sayı: 1, s. 37-49.

Batık, Z., Kaçar, S., Çavuşoğlu, Ü., Akgül, A. ve Sevin A. 2014. Kontrolör tasarımı için GA kullanıldığı Matlab ve .NET tabanlı bir windows uygulaması. APJES II-I. s.24-34.

Φαλιάγκας, K. 2013. Σύνθεση Υψηλού Επιπέδου Του Αλγορίθμου Opensurf. Student thesis, National Technical University of Athens, Greek, 99 p.

Zahedi, M. And Eslami S. 2011. Farsi/Arabic optical font recognition using SIFT features. Procedia Computer Science 3:1055-1059.

Jamjuntr, P. and Dejdumrong, N. 2012. Thai font type recognition using SIFT. 2012 Ninth International Conference on Computer Graphics, Imaging and Visualization. pp. 57-60.

Ahmad, R., Afzal M.Z., Rashid S.F., Liwicki M. and Breuel T. 2015. Scale and Rotation Invariant OCR for Pashto Cursive Script using MDLSTM Network. 13th International Conference on Document Analysis and Recognition (ICDAR). pp. 1101-1105.

Karasoy, O. ve Ballı, S. 2016. Google Maps ve Genetik Algoritmalarla GSP Çözümü İçin Öneri. XVIII Akademik Bilişim Konferansı. Adnan Menderes Üniversitesi Aydın.

Güneren, H. 2010. FPGA Üzerinde Kayan Nokta Sayı Formatı Kullanılarak Yapay Sinir Ağı Tabanlı Sınıflandırma İşlemi. Bitirme Projesi, Yıldız Teknik Üniversitesi, İstanbul, 44 s.

ÖZGEÇMİŞ

HÜSEYİN ÖZDEMİR

ieee_huseyin@hotmail.com

ÖĞRENİM BİLGİLERİ

Lisans 2009-2013	Akdeniz Üniversitesi Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü, Antalya
Yüksek Lisans 2015-2018	Akdeniz Üniversitesi Fen Bilimleri Enstitüsü, Elektrik-Elektronik Mühendisliği Anabilim Dalı, Antalya