EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

SAYISAL GÖRÜNTÜ İYİLEŞTİRME ALGORİTMALARININ GELİŞTİRİLMESİ VE BU ALGORİTMALARIN GERÇEK ZAMANLI GÖMÜLÜ SİSTEMLERDE GERÇEKLENMESİ

Ozan TEKDUR

Tez Danışmanı: Yrd. Doç. Dr. Şebnem BORA

Bilgisayar Mühendisliği Anabilim Dalı

Bilim Dalı Kodu : 619.01.00

Sunuş Tarihi : 25.07.2012

BORNOVA – İZMİR
2012

Ozan TEKDUR tarafından YÜKSEK LİSANS tezi olarak sunulan "**SAYISAL GÖRÜNTÜ İYİLEŞTİRME ALGORİTMALARININ GELİŞTİRİLMESİ VE BU ALGORİTMALARIN GERÇEK ZAMANLI GÖMÜLÜ SİSTEMLERDE GERÇEKLENMESİ**" başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve **25.07.2012** tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	**: Yrd. Doç. Dr. Şebnem BORA**	**.....**
Raportör Üye	**: Doç. Dr. Aybars UĞUR**	**.....**
Üye	**: Yrd. Doç. Dr. Cengiz Güngör**	**.....**

ÖZET

SAYISAL GÖRÜNTÜ İYİLEŞTİRME ALGORİTMALARININ GELİŞTİRİLMESİ VE BU ALGORİTMALARIN GERÇEK ZAMANLI GÖMÜLÜ SİSTEMLERDE GERÇEKLENMESİ

TEKDUR, Ozan

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı
Tez Danışmanı: Yrd. Doç. Dr. Şebnem BORA
Temmuz 2012, 87 sayfa

Sayısal görüntü işleme uygulamaları günümüzde birçok alanda etkin olarak kullanılmaktadır. Havacılık ve uzay sanayi, tıp, meteoroloji, jeoloji, parçacık fiziği, tüketici elektroniği vb. gibi pek çok ve farklı disiplinlerde, değişik amaçlar için sayısal görüntü işleme tekniklerine ihtiyaç duyulmaktadır.

Bu çalışmada, gerçek zamanlı, sabit veya hareketli resimler üzerinde görüntü işleme ve iyileştirme algoritmaları uygulanmıştır. Karşıtlık iyileştirme, histogram hesaplama, kenar belirleme, hareket tespiti ve gürültü azaltma gibi uygulamalar ALTERA DE2-70 FPGA geliştirme kartı üzerinde çalıştırılmıştır. Tasarım dili olarak VERILOG kullanılmıştır. Analog NTSC kompozit video bilgisi sayısallaştırılarak FPGA donanımına aktarılmış, görüntü üzerinde geliştirilen algoritmalar uygulanmış ve sayısal bilgi 640x480@60Hz çözünürlüğünde analog formata çevrilerek VGA monitöre gönderilmiştir.

Çalışmanın amacı gerçek zamanlı görüntü işleyen gömülü sistemler için algoritmalar geliştirmek ve bu algoritmaların FPGA üzerinde gerçeklenebildiğini göstermektir.

Anahtar sözcükler: Gerçek zamanlı gömülü sistemler, FPGA, VERILOG, ALTERA DE2-70, görüntü iyileştirme, görüntü işleme, karşıtlık, histogram, gürültü azaltma, kenar tespiti, hareket tespiti.

ABSTRACT

DEVELOPMENT OF DIGITAL IMAGE ENHANCEMENT ALGORITHMS AND IMPLEMENTATION OF THE DEVELOPED ALGORITHMS ON EMBEDDED REAL TIME SYTEMS

TEKDUR, Ozan

MSc. in Computer Engineering
Supervisor: Assistant Prof. Dr. Şebnem BORA
July 2012, 87 pages

Digital image processing applications are effectively used in countless different areas in today's world. Many different disciplines like aerospace industry, medical science, meteorology, geology, quantum physics, consumer electronics etc. use digital image processing techniques for different purposes.

In this thesis image processing algorithms are applied on still and moving pictures. Contrast enhancement, histogram calculation, edge detection, motion detection and noise reduction algorithms are successfully implemented on ALTERA DE2-70 FPGA development and education board. VERILOG hardware description language is used in code design. Analog NTSC composite video data is digitized and transferred to FPGA, algorithms are applied in FPGA and the enhanced-processed digital image is converted to 640x480@60Hz analog format. A VGA monitor is used to display the achieved results.

The main goal of the work is to develop algorithms for embedded real time image processing systems, and to realize these developed algorithms on FPGA.

Keywords: Real time embedded systems, FPGA, VERILOG, ALTERA DE2-70, image enhancement, image processing, contrast, histogram, noise reduction, edge detection, motion detection,

TEŞEKKÜR

Çalışmamda değerli yorum ve önerileri ile katkıda bulunan tez danışmanım Sayın Yrd. Doç. Dr. Şebnem Bora'ya teşekkürü bir borç bilirim.

Bu çalışmam boyunca bana maddi manevi destek ve teşviklerinden öte, sabırlarından dolayı eşim ve kızıma teşekkür ederim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

Şekil **Sayfa**

ŞEKİLLER DİZİNİ (devam)

Şekil **Sayfa**

ŞEKİLLER DİZİNİ (devam)

Şekil **Sayfa**

ŞEKİLLER DİZİNİ (devam)

Şekil **Sayfa**

ŞEKİLLER DİZİNİ (devam)

Şekil **Sayfa**

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler	Açıklama

Kısaltmalar

ASIC	Application Specific Integrated Circuit
ASSP	Application Specific Standard Products
CVBS	Composite Video Baseband Signal
CLB	Configurable Logical Blocks
DCM	Digital Clock Management
DSP	Digital Signal Processor
EAV	End of Active Video
FPGA	Field Programmable Gate Array.
HD	High Definition
IOB	Input output Blocks
NTSC	National Television Standards Committee
PAL	Phase Alternating Line
RGB	Red-Green-Blue
SAV	Start of Active Video
SD	Standard Definition

SECAM	Sequential Couleur Avec Memoire
SRAM	Synchronous Random Access Memory
VGA	Video Graphics Array
VHDL	Very high speed integrated circuit Hardware Description Language.
Y	Luminance

1. GİRİŞ

Sayısal görüntü işleme uygulamaları günümüzde sayısız alanda etkin olarak kullanılmaktadır. Havacılık ve uzay sanayi, tıp, meteoroloji, jeoloji, parçacık fiziği, tüketici elektroniği vb. gibi pek çok ve farklı disiplinlerde, değişik amaçlar için sayısal görüntü işleme tekniklerine ihtiyaç duyulmaktadır. Tüm bu uygulamalarda yapılan işlem kaynak görüntü üzerinde çeşitli matematiksel yöntemler kullanılarak anlamlı verinin elde edilmeye çalışılmasıdır.

Kaynak görüntü sabit resim veya hareketli resim (video) olabilir. Sabit resim uygulamalarında (örneğin bir röntgen filminde sorunlu bölgenin belirginleştirilmesi) kullanılan teknikler ile hareketli resim uygulamalarında (örneğin bir anti-tank roketinin hedef takip yazılımında, hedefin resim içerisindeki diğer bileşenlerden ayrılarak gerçek zamanlı takip edilmesinde) kullanılan teknikler benzerlik gösterir.

Gömülü sistemlerde sayısal veri işleme uygulamaları daha çok Sayısal Sinyal İşlemcileri (DSP) aracılığı ile yapılmaktadır. DSP'lerin avantajı C ve benzeri programlama dilleri kullanılarak tasarımcının amacına uygun olarak esnek şekilde programlanabilmeleridir. DSP'ler tasarım sürecini hızlandırarak ürünün pazara çıkış süresini azaltmaktadır. Ancak programlanabilir DSP işlemcilerinin performansı düşüktür. Bunun nedeni de DSP işlemcisinin çok fonksiyonlu yapısından dolayı pek çok alternatifi aynı anda hesaba katmak zorunda olmasıdır.

Özellikle gerçek zamanlı görüntü işleme uygulamalarında veri boyutu çok büyük olduğu için DSP'lerin performansı yetersiz kalmaktadır.

Bu problemin üstesinden gelmek için Uygulamaya Özel Entegre Devre (ASIC) yongaları sayısal görüntü işleyen gömülü sistemlerde kullanılabilir. Bir ASIC, hem işlevsel hem de fiziksel açıdan özel amaçlı, yani belli bir uygulamaya hizmet eden bir yongadır. ASIC'ler DSP'lere göre yüksek performansa sahip olmalarına rağmen, DSP'lerin esnek yapısında değillerdir. Yeniden programlanma özellikleri olmadığı için yeni bir fonksiyon istendiğinde veya yazılımda bir iyileştirme yapıldığında ASIC yeniden tasarlanmalıdır. Bu maliyeti arttıran ve süreci uzatan olumsuz bir durum olarak tasarımcıları zorlamaktadır.

Sahada Programlanabilir Kapı Dizileri (FPGA), DSP'lerin esnek programlanabilir yapısı ile ASIC'lerin yüksek performans özelliğini birleştiren bir teknolojidir. Bir FPGA, binlerce mantık elemanından ve küçük boyutta bir Rastgele Erişimli Bellek'ten (RAM) oluşur. Tüm bu birimler FPGA içerisinde

birbirine bağlıdır. Tasarımcı bu mantık ve hafıza birimlerini VERILOG veya VHDL gibi donanım tanımlama dillerini kullanarak istediği şekilde programlar. FPGA yapısı gereği teorik olarak sınırsız sayıda yeniden programlanmaya izin vermektedir. Bu da FPGA ile yapılan tasarımın esnekliğini arttırmaktadır.

Bu çalışmada, gerçek zamanlı, sabit veya hareketli resimler üzerinde görüntü işleme ve iyileştirme algoritmaları uygulanmış ve bu algoritmalar ALTERA DE2-70 FPGA geliştirme kartı üzerinde gerçeklenerek başarıyla çalıştırılmıştır. Tasarım dili olarak VERILOG kullanılmıştır. Analog NTSC kompozit video bilgisi sayısallaştırılarak FPGA donanımına aktarılmış, görüntü üzerinde geliştirilen algoritmalar uygulanmış ve sayısal bilgi 640x480@60Hz çözünürlüğünde analog formata çevrilerek VGA monitöre gönderilmiştir. Çalışmanın amacı gerçek zamanlı görüntü işleyen gömülü sistemler için algoritmalar geliştirmek ve bu algoritmaları FPGA üzerinde gerçeklemektir.

Tez çalışmasının katkılarından biri gerçeklenen algoritmaların askeri sistemler, sivil güvenlik ve medikal gibi sektörlerde kullanılabilir olmasıdır. Ayrıca gömülü sistemlerde gerçek zamanlı görüntü işleme uygulamaları konusunda çalışmak isteyen araştırmacılar için faydalı bir kaynak olacağı düşünülmektedir.

Geliştirilen algoritmalar tez çalışmasında kaba kod (pseudocode) olarak sunulmuştur. Ek olarak algoritmayı anlaşılır kılmak için gereken yerlerde açıklamalarda bulunulmuştur.

Tez çalışması altı bölümden oluşmaktadır.

Birinci bölümde Tez konusu hakkında genel bir tanıtım yapılmıştır.

İkinci bölümde Sahada Programlanabilir Kapı Dizileri (FPGA) hakkında genel bilgiler verilmiş, avantaj ve dezavantajları tartışılmıştır. FPGA programlama dilleri anlatılmıştır. Tez çalışmasında kullanılan ALTERA DE2-70 FPGA geliştirme kartı donanım ve yazılım mimarileri açıklanmıştır. FPGA kullanılarak sayısal görüntü işleme alanında yapılan çalışmalar incelenmiş, gerçeklenen uygulamalardan örnekler verilmiştir.

Üçüncü bölümde temel analog ve sayısal video bilgileri sunulmuştur. Binişimsizleştirme (de-interlacing) işlemi anlatılmış, kompozit video işareti incelenmiştir. RGB, YPbPr ve ITU-R BT 656 gibi sayısal video formatları anlatılmıştır.

Dördüncü bölümde tez çalışmasında kullanılan temel sayısal görüntü işleme teknikleri tartışılmıştır. Histogram hesaplama, karşıtlık iyileştirme, gürültü azaltma, kenar tespit edilmesi, daraltma ve genleştirme gibi morfolojik hesaplamalar ve renk düzlemi operasyonları incelenmiştir.

Beşinci bölümde geliştirilen algoritmalar sunulmuştur. Hareketli ve sabit resimler için ilgili algoritmalar test edilmiş ve çıktılar incelenmiştir.

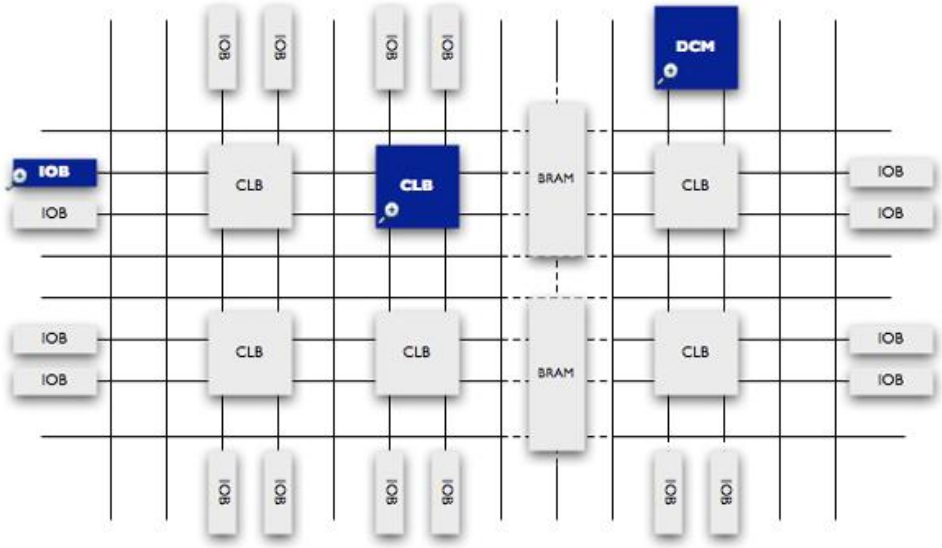
Altıncı bölümde tez çalışmasının sonuçları ve ileride yapılabilecek çalışmalar tartışılmıştır.

2. SAHADA PROGRAMLANABİLİR KAPI DİZİLERİ (FPGA)

2.1 FPGA Nedir?

Sahada Programlanabilir Kapı Dizileri (FPGA), programlanabilir mantık elemanları dizilerinden ve bu elemanları birbiri ile ilişkilendirmeye yarayan programlanabilir ara bağlantı anahtarlarından oluşur. FPGA'ler de iki seviye programlama bulunmaktadır. Her mantık elemanı birbirinden bağımsız şekilde, istenen basit mantıksal işlemleri yapmak için programlanabilir. Daha sonra FPGA içerisindeki anahtarlar programlanarak, bu mantık elemanlarını birbiri ile ilişkilendirir ve istenen karmaşık fonksiyonları yürüten mantık devresi oluşturulur (Bezen, 1999).

Tüm FPGA'ler için aynı mimariden söz edilemez. Her bir üreticinin mimari tasarımı farklılık göstermektedir.Şekil 2-1'de Xilinx firmasının FPGA mimarisi gösterilmiştir.



Şekil 2-1 Xilinx FPGA mimarisi (Xilinx, 2012)

Xilinx FPGA mimarisine göre FPGA'de öncelikle konfigüre edilebilir mantıksal bloklar (CLB) birbirinden bağımsız olarak programlanır, ikinci aşamada CLB'ler arasındaki bağlantılar programlanarak istenen kompleks fonksiyonları gerçekleştirecek devre oluşturulur. Programın girdi ve çıktıları giriş/çıkış ara yüz blokları (IOB) tarafından yönetilir ve CLB'lere yine ara bağlantı hatlarının programlanması suretiyle bağlanır. Çalışma senkronizasyonunu sağlamak için sayısal saat yönetimi bloğu (DCM) kullanılır. Tüm bu işlemler

esnasında ihtiyaç duyulan hafıza yönetimi SRAM bloğu tarafından sağlanır (Xilinx, 2012).

2.2 FPGA'in Avantajları

Gömülü sistemlerde sistem mimarisi bakımından çeşitli alternatifler mevcuttur. Bunlar standart ASIC'ler, ASSP'ler (uygulamaya özel standart ürünler), DSP veya FPGA gibi programlanabilir çözümlerdir. İdeal mimarinin şu özelliklerde olması beklenir: Yüksek performans, esnek programlanabilme, zahmetsiz güncelleme, düşük bakım maliyeti, güvenirlilik, düşük geliştirme maliyeti, zamanında pazara çıkabilme ve üretim adedi arttıkça düşük maliyete geçebilme yeteneği. (Altera, 2007; National Instruments, 2012).

Gerçek zamanlı sayısal görüntü işleme uygulamalarında veri boyutu ihtiyacı gün geçtikçe artmaktadır. Güvenlik uygulamalarında kullanılan standart 352 x 288 çözünürlüğü 704 x 576 olarak değişmektedir. Endüstride kullanılan kamera standardı 1280 x 720 çözünürlüğü seviyesine ulaşmıştır. Askeri güvenlik, medikal görüntüleme ve makine görüntüleme uygulamalarında daha da yüksek çözünürlüklere geçilmektedir. Düşük çözünürlüklü (SD) videodan yüksek çözünürlüklü (HD) videoya geçildiğinde veri boyutu altı kat artmaktadır (Altera, 2007).

Yukarıda bahsedilen sistem mimarisi alternatiflerin her biri için olumlu veya olumsuz argümanlar sunulabilir. Bu tez çalışmasında yüksek veri boyutu, yüksek performans ve esnek programlanabilme ön planda olduğu için FPGA kullanılarak gerçek zamanlı görüntü işleme uygulamaları geliştirilmiştir. FPGA'in avantajları aşağıdaki gibi özetlenebilir.

2.2.1 Yüksek performans

FPGA'ler, paralel donanım mimarisi kullanmanın avantajı ile bir birim saat döngüsünde sıralı işletim mimarisine sahip DSP'lerden daha fazla hesaplamayı yaparlar. Ek olarak giriş çıkışların donanım seviyesinde kontrol edilmesi sayesinde, özelleştirilmiş uygulamalara daha hızlı cevap vermektedirler (National Instruments, 2012).

Örneğin 1GHz'de çalışan bir DSP, H.264 HD formatında bir videoyu çözememektedir. Çünkü tüm işlemci gücünü H.264 HD formatındaki sayısal veriyi çözmeye harcayan DSP'de çözülen veri üzerinde yapılması gereken boyutlandırma, binişimsizleştirme, filtreleme ve renk düzlemi değiştirme

uygulamaları için yeterli işlemci gücü kalmamaktadır. FPGA'ler günümüzde programlanabilir mimari alternatifleri içerisinde bu sorunu gideren tek çözüm olarak karşımıza çıkmaktadır. Bazı durumlarda en iyi sonuç bir FPGA ile harici bir DSP'nin birlikte kullanılması ile alınmaktadır (Altera, 2007).

2.2.2 Esneklik, kolay güncellenebilme ve düşük bakım maliyeti

Teknoloji hızla ilerlediği için mimariler de esnek ve kolay güncellenebilir olmalıdır. Bu durum ASIC ve ASSP'lerin programlanabilir mimarilere göre olan dezavantajını ortaya koymaktadır. (Altera, 2007).

Sahada Programlanabilir Kapı Dizileri (FPGA) mantıksal blokların ve ara bağlantıların imalat aşamasından sonra tasarımcı tarafından programlanabilmesi sebebiyle sahada programlanabilir adını taşımaktadır. Tasarımcının ihtiyacı olan mantıksal fonksiyonları gerçekleştirebilmesi amacıyla sahada programlanabilir olarak üretilmiştir (Gacar, 2009).

Sık karşılaşılan bir durum, sayısal haberleşme protokollerinin zaman içinde değişmesi ve güncellenmesidir. ASIC kullanımında sistem bu değişikliğe ayak uyduramazken FPGA kullanılması durumunda ilgili sistem sahada yeniden programlanarak istenen değişiklikler yapılabilir. Bu şekilde yeniden ASIC tasarımı maliyeti olmadan sitemin bakımı çok daha ucuza yapılmaktadır (National Instruments, 2012).

2.2.3 Güvenirlik

DSP uygulamalarında yazılım geliştirme araçları programlama için gerekli ortamı sağlarlarken, FPGA yongası donanımsal olarak programlanırlar. İşlemci temelli mimariler görevleri yönetmek, birden çok işlemci varsa bunlar için kaynakları yönetmek gibi uygulamaları içerir. Bu mimarilerde sürücü katmanı donanım kaynaklarını yönetirken işletim sistemi hafıza ve işlemci operasyonlarından sorumludur. Bir işlemci bir birim zamanda ancak tek bir komut işleyebilir. Dolayısı ile bu mimarilerde kritik görevler için her zaman işlemci kuyruğunda bekleme riski bulunmaktadır. FPGA'ler işletim sistemi kullanmadıklarından dolayı ve paralel komut işleyebilme yetenekleri sayesinde bu tür güvenirlik risklerini en aza indirgerler (National Instruments, 2012).

2.2.4 Düşük geliştirme maliyeti ve zamanında pazara ulaşabilme

Standart 90-nm bir ASIC için özel bir uygulama geliştirme maliyeti ortalama otuz milyon Amerikan dolarıdır. Sadece çok yüksek hacimlerde üretim yapan tüketici elektroniği firmaları bu rakamları karşılayabilmektedir (Altera, 2007).

FPGA'ler ortak kullanım için tasarlanan yongalardır. FPGA üreticileri bir yongayı geliştirdiklerinde bundan milyonlarca adet üreterek geliştirme maliyetlerini çok fazla sayıdaki müşteriye bölme imkânına sahiptir. Bu nedenle FPGA'ler için yüksek geliştirme maliyeti tüketici tarafından karşılanmaz. Ek olarak programlanabilir yapıda olmalarından dolayı hedeflenen zamanda pazara çıkmak FPGA kullanımında çok daha kolay olmaktadır (Bezen, 1999).

2.3 Programlama Dili

FPGA'ler de algoritmaları oluşturmak için kullanılan programlama dillerine genel olarak "donanım tanımlama dili" denir. Donanım tanımlama dilleri FPGA içerisindeki mantıksal blokların ve ara bağlantı anahtarlarının uygun şekilde programlanmasında kullanılır. Standart programlama dillerine benzer ara yüzü bulunur. En sık kullanılan diller VERILOG ve VHDL'dir.

2.3.1 Verilog

Verilog elektronik sistemleri modellemek için kullanılan bir donanım tanımlama dilidir. Verilog analog, sayısal ve karışık işaretli devrelerin tasarımını, doğrulanmasını ve yürütülmesini değişik düzeylerde desteklemektedir.

Verilog C programlama diline yakın bir söz dizimine sahiptir. Verilog geleneksel programlama dilleri gibi basamaklarını tam olarak ardışık bir şekilde yürütmez. Verilog tasarımı modüller arasında bir hiyerarşi bulundurur. Modüller bir takım giriş, çıkış ve çift yönlü portlar şeklinde tanımlanır. Bir modül içinde yazmaç ve kablo listesi bulunur. Eş zamanlı ve ardışık ifadeler modülün davranışını; portların, kabloların ve yazmaçların arasındaki ilişki ile tanımlar.

Ardışık ifadeler bir begin/end bloğuna konur ve blokla beraber ardışık olarak yürütülür. Tüm eş zamanlı ifadeler ve begin/end blokları koşut olarak yürütülür. Tasarımdaki modüller sadece sentezlenebilir ifadeler içeriyorsa bu tasarımın donanımda gerçekleştirilecek temel bileşenlerini ve bağlantılarını içeren

bağlantı listesi, yazılım sayesinde sentezlenir. Elde edilen bu bağlantı listesi bir FPGA'i tanımlamak amacıyla kullanılır (Wikipedia, 2012).

2.3.2 VHDL

VHDL(Very high speed integrated circuit Hardware Description Language) "Çok yüksek hızda yonga Donanım Tanımlama Dili" olarak ifade edilir. VHDL ilk olarak 1987 yılında IEEE tarafından IEEE 1076 standardı olarak tanımlanmıştır. 1993 yılında yine IEEE tarafından revize edilmiştir. Temel olarak yongalarda kullanılan mantık elemanlarının davranışlarını tanımlar. Kayıtçı transfer seviyesinde yazılan algoritmayı kapı programlama seviyesine sentezleyerek (otomatik olarak oluşturarak) tasarımcının daha anlamlı ve kullanıcı dostu bir ortamda çalışmasına olanak tanır. (Doulos, 1995).

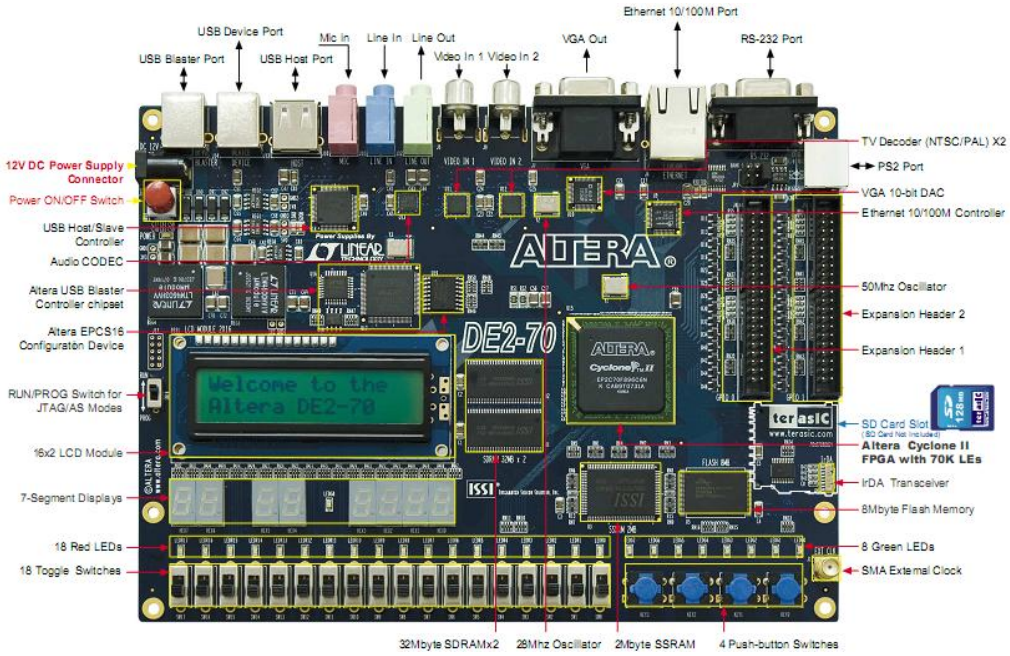
Bu çalışmada donanım tanımlama dili olarak C programlama diline benzemesi sebebi ile Verilog tercih edilmiştir.

2.4 Tez Çalışmasında Kullanılan FPGA Çalışma Kartı

Tez çalışmasında geliştirilen algoritmalar DE2-70 çalışma ve geliştirme kartı üzerinde gerçeklenmiştir. Seçilen kart geliştirilen algoritmaların uygulanması için gerekli ara yüzlere sahiptir. Gerçek zamanlı video işleme için yeterli bir FPGA içermesi, analog kompozit video girişinin olması ve analog VGA çıkışının bulunması tasarım yapmayı kolaylaştırmıştır.

2.4.1 DE2-70 mimarisi

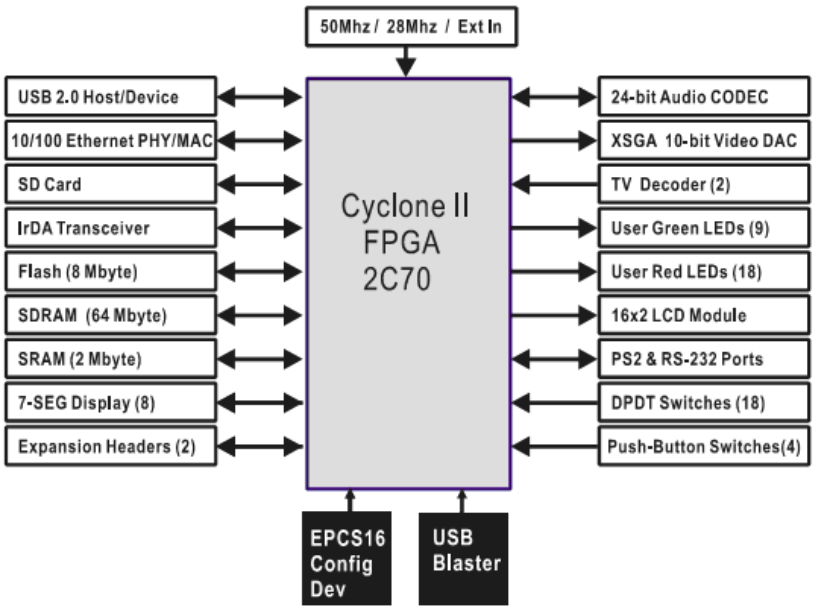
DE2-70 çalışma ve geliştirme kartı FPGA üzerinde çalışmak için çok faydalı bir araçtır. 70.000 mantık elemanı içeren Altera Cyclone® II 2C70 FPGA içermesi ve üzerinde bulunan ek devreler ile kullanıcıların pek çok basit veya karmaşık algoritmayı gerçekleştirmelerine olanak sağlar. Şekil 2-2'de DE2-70 kartının bir resmi görülmektedir. Kartın üzerindeki devrelerin fiziksel olarak yerleşimi, giriş-çıkış ara yüzleri görülmektedir (Terasic, 2009).



Şekil 2-2 The DE2-70 çalışma ve geliştirme kartı (Terasic, 2009)

2.4.2 DE2-70 blok şeması

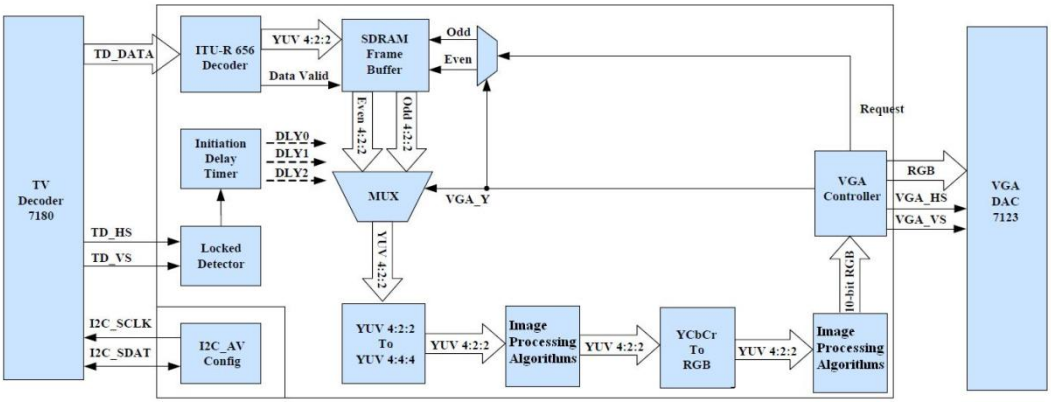
Şekil 2-3'de DE2-70 kartının blok şeması verilmiştir. Kullanıcı için en üst seviye esneklik sağlaması açısından tüm bağlantılar Cyclone II FPGA yongasından getirilmiştir. Bu sayede kullanıcı FPGA'i konfigüre ederek istediği tasarımı oluşturabilir (Terasic, 2009).



Şekil 2-3 DE2-70 blok şeması (Terasic 2009).

2.4.3 Tez Çalışmasında Kullanılan Donanım Konfigürasyonu

DE2-70 geliştirme kartının tez çalışmasında kullanılan konfigürasyonu Şekil 2-4'de belirtilmiştir. Bu konfigürasyonda analog kompozit video girişinden alınan kaynak resim ITU-R-656 formatına dönüştürülür. ITU-R-656 dekoderi YCbCr 4:2:2 formatındaki video verisini binişimleştirilmiş (interlaced) halde SDRAM'de saklar. Giriş video satırlar halinde geldikçe bu işlem devam eder. Bir çerçeve tamamlandıktan sonra video üzerinde binişimsizleştirme işlemi uygulanarak elde edilen resim bir sonraki bloğa aktarılır. Bu blokta sayısal video formatı 4:2:2 den 4:4:4'e çevirilir. Bir sonraki aşamada YCbCr sayısal video verisi sayısal RGB verisine dönüştürülür. Son olarak VGA denetim bloğu senkronizasyon sinyallerini üretir ve sayısal video RGB ile birlikte tüm veriyi sayısal-analog çeviriciye gönderir. Analog RGB formatına çevrilen video VGA çıkışından dış monitöre gönderilir (Terasic, 2009).



Şekil 2-4 Tez çalışmasında kullanılan donanım konfigürasyonu

2.4.3.1 TV decoder 7180

DE2-70 çalışma kartında iki adet ADV7180 TV sinyal çözücü yongası bulunmaktadır. ADV7180 gelen CVBS sinyalinin standardını (NTSC, PAL) otomatik olarak tespit eden ve bu sinyali 8-bit 4:2:2 ITU-R BT.656 ara yüzüne sayısallaştıran bir yongadır. ADV7180 I2C seri haberleşme protokolü kullanılarak konfigüre edilmektedir.

2.4.3.2 VGA DAC 7123

DE2-70 çalışma kartında bir adet 16-pin D-SUB VGA konektörü bulunmaktadır. VGA senkronizasyon sinyalleri Cyclone II FPGA tarafından üretilmektedir. ADV7123 üç kanallı (R, G, B) 10-bit video sayısal-analog sinyal çevirici olarak kullanılmaktadır. ADV7123 maksimum 1600x1200@100 Hz

analog video çözünürlüğünü desteklemektedir. Bu tez çalışmasında video çıkış çözünürlüğü olarak 800x600@65Hz kullanılmıştır.

ADV7180 ve ADV7123 yongaları dışında Şekil 2-4'de gösterilen diğer tüm modüller Cyclone II FPGA içerisinde oluşturulmaktadır. Bölüm 2.4.3.3 – bölüm 2.4.3.12 bu blokları açıklamaktadır.

FPGA'in içerisindeki bu modüllerin üst blok mimari algoritması Şekil 2-5 ve Şekil 2-6'da verilmiştir. Örnek algoritma olarak 5.1.2 kısmında detayları verilen adaptif medyan filtreleme algoritması üst bloğu mimarisi seçilmiştir. 5. Bölümde anlatılan her bir ayrı başlık için ayrı bir üst blok mimarisi mevcuttur. Şekil 2-4 içindeki "Tez çalışmasında kullanılan görüntü işleme ve iyileştirme algoritmalarının uygulanması" ifadesi ile belirtilen kısımdaki modüller, yapılan çalışmaya uygun olacak şekilde her uygulama için değişmektedir.

Bu mimari DE2-70 TV çalışma kartı ile birlikte gelen temel çalıştırma yazılımı üzerine yapılan değişiklikler ile oluşturulmuştur.

```
DE2_70_MAIN
   [iCLOCK,iSwitch,iKEY;
    DRAM_DQ, oDRAM0_A, oDRAM1_A,oDRAM0_LDQM0, oDRAM1_LDQM0,
    oDRAM0_UDQM1, oDRAM1_UDQM1, oDRAM0_WE_N, oDRAM1_WE_N,
    oDRAM0_CS_N, oDRAM1_CS_N, oDRAM0_CAS_N, oDRAM1_CAS_N,
    oDRAM0_RAS_N, oDRAM1_RAS_N, oDRAM0_BA, oDRAM1_BA;
    oDRAM0_CLK, oDRAM1_CLK, oDRAM0_CKE,
    I2C_SDAT, oI2C_SCLK,
    oVGA_CLOCK, oVGA_HS, oVGA_VS, oVGA_BLANK_N, oVGA_SYNC_N;
    oVGA_R, oVGA_G, oVGA_B;
    iTD1_CLK27, iTD1_HS, iTD1_VS, iTD1_D, oTD1_RESET_N]

    //Parametre atamaları yapılır
    if (VGA_Y[0] = 0)
        m1VGA_Read  = VGA_Read;
    if (VGA_Y[0] = VGA_Read)
        m2VGA_Read  = 0;
    if (VGA_Y[0] != m1YCbCr )
        mYCbCr_d    = m2YCbCr;
    mYCbCr  = m5YCbCr;
    Tmp1    =   m4YCbCr[7:0]+mYCbCr_d[7:0];
    Tmp2    =   m4YCbCr[15:8]+mYCbCr_d[15:8];
    Tmp3    =   Tmp1[8:2]+m3YCbCr[7:1];
    Tmp4    =   Tmp2[8:2]+m3YCbCr[15:9];
    m5YCbCr =   {Tmp4,Tmp3};

    //  2.4.3.3 I2C_AV_konfigürasyon modülü
    I2C_AV_Config   u1  (   .iCLOCK(iCLOCK), .RESET(iSwitch[0]),
                            .I2C_SCLK(oI2C_SCLK), .I2C_SDAT(I2C_SDAT)   );
    //  2.4.3.4 TV dedektör stabilizasyonu modülü
    TD_Detect           u2  (   .oTD_Stable(TD_Stable), .iTD_VS(iTD1_VS),
                                .iTD_HS(iTD1_HS), .RESET(iSwitch[0])    );
    //  2.4.3.5 Yeniden başlatma (reset) modülü
    Reset_Delay         u3  (   .iCLOCK(iCLOCK), .iRST(TD_Stable),
                                .oRST_0(DLY0), .oRST_1(DLY1), .
                                oRST_2(DLY2));
    //  2.4.3.6 ITU-R 656 ==> YUV 4:2:2 çözücü modülü
    ITU_656_Decoder     u4  (   .iTD_DATA(iTD1_D), .oTV_X(TV_X), .
                                oYCbCr(YCbCr), .oDVAL(TV_DVAL),
                                .iKEYap_CbCr(Quotient[0]), .iSkip(Remain==4'h0),
                                .RESET(DLY1), .iCLOCK(iTD1_CLK27));
```

Şekil 2-5 DE2-70 çalışma kartı üst blok yazılım mimarisi-1

```
42      //  2.4.3.7-8 Veri karıştırıcı ve SDRAM modülü
43      Sdram_Control_4Port u6  (   //  HOST
44                                  .REF_CLK(iTD1_CLK27), .CLK_18(AUD_CTRL_CLK),
45                                  .RESET_N(1'b1),
46                                  //  FIFO Write 1
47                                  .WR1_DATA(YCbCr), .WR1(TV_DVAL),
48                                  .WR1_FULL(WR1_FULL), .WR1_ADDR(0),
49                                  .WR1_MAX_ADDR(640*507), .WR1_LENGTH(9'h80),
50                                  .WR1_LOAD(!DLY0), .WR1_CLK(iTD1_CLK27),
51                                  //  FIFO Read 1
52                                  .RD1_DATA(m1YCbCr), .RD1(m1VGA_Read),
53                                  RD1_ADDR(640*13), .RD1_MAX_ADDR(640*253),
54                                  .RD1_LENGTH(9'h80), .RD1_LOAD(!DLY0),
55                                  .RD1_CLK(iTD1_CLK27),
56                                  //  FIFO Read 2
57                                  .RD2_DATA(m2YCbCr), .RD2(m2VGA_Read),
58                                  .RD2_ADDR(640*267), .RD2_MAX_ADDR(640*507),
59                                  .RD2_LENGTH(9'h80), .RD2_LOAD(!DLY0),
60                                  .RD2_CLK(iTD1_CLK27),
61                                  //  SDRAM
62                                  .SA(oDRAM0_A), .BA({oDRAM0_BA[1],oDRAM0_BA[0]}),
63                                  .CS_N(oDRAM0_CS_N), .CKE(oDRAM0_CKE),
64                                  .RAS_N(oDRAM0_RAS_N), .CAS_N(oDRAM0_CAS_N),
65                                  .WE_N(oDRAM0_WE_N), .DQ(DRAM_DQ),
66                                  .DQM({oDRAM0_UDQM1,oDRAM0_LDQM0}), .SDR_CLK(oDRAM0_CLK));
67      //  2.4.3.9 YUV 4:2:2 - YUV 4:4:4 çevirici modülü
68      YUV422_to_444       u7  (   .iYCbCr(mYCbCr), .oY(nY), .oCb(nCb), .oCr(nCr),
69                                  .iX(VGA_X), .iCLOCK(iTD1_CLK27), .RESET(DLY0));
70      //  2.4.3.10    Tez çalışmasında kullanılan görüntü işleme ve
71      //iyileştirme algoritmalarının uygulanması
72
73      adaptiveMEDIAN      u51(    .iData(nY), .iPb(nCb), .iPr(nCr),
74                                  .oData(filter_Y), .oPb(filter_Cb), .oPr(filter_Cr),
75                                  .iSel3(iKEY[3:0]), .iCLOCK(iTD1_CLK27), .RESET(DLY2));
76      yarimEKRAN          u70 (   .oY(ooY), .oCb(ooCb), .oCr(ooCr), .iY(filter_Y),
77                                  .iiY(nY), .iiCb(nCb), .iiCr(nCr),
78                                  .iCb(filter_Cb), .iCr(filter_Cr),
79                                  .iX_Cont(VGA_X), .iY_Cont(VGA_Y),
80                                  .iCLOCK(iTD1_CLK27), .iKEY(iKEY[4]), .iRST(!DLY2));
81
82      //  2.4.3.10    Tez çalışmasında kullanılan görüntü işleme ve
83      //iyileştirme algoritmalarının uygulanması
84
85      //  2.4.3.11 YCbCr 8-bit to RGB-10 bit çevirici modülü
86      YCbCr2RGB           u8  (   .Red(mRed), .Green(mGreen), .Blue(mBlue),
87                                  .oDVAL(mDVAL), .iY(ooY), .iCb(ooCb), .iCr(ooCr),
88                                  .iDVAL(VGA_Read), .iRESET(!DLY2), .iCLOCK(iTD1_CLK27));
89      //  2.4.3.12    VGA denetleyici modülü
90      VGA_Ctrl            u9  (   .iRed(mRed), .iGreen(mGreen), .iBlue(mBlue),
91                                  .oCurrent_X(VGA_X), .oCurrent_Y(VGA_Y), .oRequest(VGA_Read),
92                                  .oVGA_R(oVGA_R), .oVGA_G(oVGA_G), .oVGA_B(oVGA_B),
93                                  .oVGA_HS(oVGA_HS), .oVGA_VS(oVGA_VS),
94                                  .oVGA_SYNC(oVGA_SYNC_N), .oVGA_BLANK(oVGA_BLANK_N),
95                                  .oVGA_CLOCK(oVGA_CLOCK), .iCLOCK(iTD1_CLK27), .RESET(DLY2));
```

Şekil 2-6 DE2-70 çalışma kartı üst blok yazılım mimarisi-2

2.4.3.3 I2C_AV_konfigürasyon modülü

I2C_AV_Config seri haberleşme bloğu ADV7180 yongasını programlamak için kullanılmaktadır. Bu tez çalışmasında video giriş işareti olarak NTSC CVBS kullanıldığından, I2C_AV_Config bloğu buna uygun olarak konfigüre edilmektedir. I2C_AV_Config modülü olarak DE2-70 kartı ile birlikte gelen standart modül uygulaması kullanılmıştır.

2.4.3.4 TV dedektör stabilizasyonu modülü

ADV7180 TV sinyal çözücü yongası DE2-70 çalışma kartına ilk enerji verildiği anda hazır olmamaktadır. Bu yonganın hazır olabilmesi için video sinyali referansına göre dokuz dikey senkronizasyon periyoduna ihtiyaç vardır. TV Dedektör Stabilizasyonu modülü gelen video sinyalindeki dokuz dikey senkronizasyon işaretini sayar, sayaç dokuz olduğunda TD-stable_signal çıktısını üretir. Bu çıktıya bağlı olarak FPGA içerisindeki diğer tüm modüller yeniden başlatılarak sistem çalışmaya hazır hale getirilir. Bu modül Şekil 2-4'de "Locked Detector" olarak belirtilmiştir. TV Dedektör Stabilizasyonu olarak DE2-70 kartı ile birlikte gelen standart modül uygulaması kullanılmıştır.

2.4.3.5 Yeniden başlatma (reset) modülü

Tüm gömülü sistemlerde olduğu gibi DE2-70 mimarisindeki modüllerin de sisteme enerji verildikten tüm elektriksel sinyaller ve arayüzler oturduktan sonra yeniden başlatılması gerekmektedir. Bu işlem DE2-70 mimarisinde "Yeniden Başlatma Modulü" ile yapılmaktadır. DE2-70 mimarisinde üç farklı yeniden başlatma (reset) sinyali üretilmektedir. Reset0, Reset1 ve Reset2. Yeniden başlatma modulu (0)hex'den (1FFFFF)hex'e kadar sayar ve "Reset0" sinyalini üretir. Bu sinyal ile SRAM ve YUV 4:2:2 – YUV 4:4:4 Çevirici modülleri yeniden başlatılır. Reset modülü "Reset0" sinyali üretildikten sonra saymaya devam eder ve (2FFFFF)hex'e ulaştığında "Reset1" sinyali üretilir. Bu sinyal ile ITU-R656 modülü yeniden başlatılır. Son olarak reset modülü (2FFFFF)hex'den (3FFFFF)hex'e kadar sayar ve "Reset2" sinyalini üretir. Bu sinyal ile YCbCr – RGB çevirici, VGA Denetleyici ve tez çalışmasında kullanılan görüntü işleme ve iyileştirme modülleri yeniden başlatılır. Bu modül Şekil 2-4'de "Initiation Delay Timer" olarak belirtilmiştir. Yeniden başlatma modülü olarak DE2-70 kartı ile birlikte gelen standart modül uygulaması kullanılmıştır.

2.4.3.6 ITU-R 656 çözücü modülü

ADV7180 yongasından gelen ITU-R 656 formatındaki sayısal video bu modül içerisinde YUV 4:2:2 formatına çevrilir ve SRAM modülüne gönderilir. ITU-R 656 Çözücü modülü olarak DE2-70 kartı ile birlikte gelen standart modül uygulaması kullanılmıştır.

2.4.3.7 SDRAM modülü

ITU-R 656 çözücü modülünden gelen veri SDRAM bloğunda depolanır. 3.1.1 bölümünde detayları anlatıldığı üzere gelen veri tek satırlar ve çift satırlar

olmak üzere iki kısımdan oluşmaktadır. SDRAM bloğunda sırası ile 640 x 240 tek satırlar ve 640 x 240 çift satırlar olarak depolanır.

2.4.3.8 Veri karıştırıcı modülü

CVBS sinyalinden sayısallaştırılan video SDRAM'de tek ve çift satırlar olarak ayrı alanlarda tutulur. Ancak görüntü işleme ve iyileştirme algoritmalarının uygulanması ve monitör çıkışının alınabilmesi için tek ve çift satırların birleştirilerek binişimsizleştirme (de-interlacing) işlemine tabi tutulması gerekir (Bkz. 3.1.1.1). Bu işlem Veri Karıştırıcı bloğunda yapılır. VGA denetleyicisi tarafından SDRAM'den talep edilen 640 x 480 video, veri karıştırıcının tek satırlardan (640 x 240) ve çift satırlardan (640 x 240) sırasıyla birer satır alarak tüm çerçevenin taranması ile 640 x 480 çözünürlüğünde resmin edilmesiyle oluşturulur. Bu modül Şekil 2-4'de "MUX" olarak belirtilmiştir. SDRAM ve Veri Karıştırıcı modülleri olarak DE2-70 kartı ile birlikte gelen standart modül uygulaması kullanılmıştır. Standart modül üzerinde 640x480 ekran çözünürlüğüne uygun olacak şekilde ilgili parametreler ayarlanmıştır.

Ek olarak 5.4 "Gerçek Zamanlı Hareket Tespit Etme" kısmında DE2-70 kartı üzerindeki ikinci SDRAM'in kullanılmasına ihtiyaç duyulmuştur. Bu algoritma için standart SDRAM modülü alınıp ikinci SDRAM'e iki çerçeveyi hafızada tutacak şekilde uyarlanmıştır. Bu algoritmayı destekleyecek şekilde 2.4.3.2 kısmında (Şekil 2-5 ve Şekil 2-6) detayları verilen DE2-70_main algoritmasının da uyarlanması gerekmiştir.

2.4.3.9 YUV 4:2:2 – YUV 4:4:4 çevirici modülü

Veri Karıştırıcı modülü tarafından binişimsizleştirilen resim YUV 4:2:2 – YUV 4:4:4 çevirici modülüne gönderilir. Bu modül YUV 4:2:2 formatındaki resmin renk bilgisini (UV) iki piksel için tekrar ederek YUV 4:4:4 formatına çevirir (Bkz 3.2.1). Dolayısı ile bu module giren 16-bitlik veri 24-bitlik veriye dönüştürülmüş olur. Elde edilen 24-bit resim bu tez çalışmasında kullanılan görüntü işleme ve iyileştirme algoritmalarının uygulanması için ilgili modüllere gönderilir. YUV 4:2:2 – YUV 4:4:4 çevirici modülü olarak DE2-70 kartı ile birlikte gelen standart modül uygulaması kullanılmıştır.

2.4.3.10 Tez çalışmasında kullanılan görüntü işleme ve iyileştirme algoritmalarının uygulanması

6. Bölümde detayları verilen algoritmalar yapılan işlemin içeriğine göre 8-bit veya 10-bit olarak tasarlanmıştır. Bu nedenle algoritmaların bir kısmı YUV

4:2:2 – YUV 4:4:4 çevirici modülünün çıktısı olan 24-bit YUV video üzerinde, bir kısmı ise YCbCR – RGB çevirici modülü çıktısı olan 30-bit RGB video üzerinde uygulanır.

2.4.3.11 YCbCr – RGB çevirici modülü

6. Bölümde detayları verilen algoritmaların uygulanmasını takiben 24-bit YUV 4:4:4 verisi YCbCr – RGB çevirici modülünde 30-bit RGB haline dönüştürülür. Bunun için 3.2.2 bölümünde verilen YCbCr - RGB dönüşüm denklemi kullanılır. 24-bit veriyi 30-bit'e dönüştürmek için ise denklemde verilen tüm değerler 512 ile çarpılır, çıkan sonuçlar 128'e bölünür. Bu şekilde 2-bit veriye eklenmiş olur. YCbCr - RGB çevirici modülü olarak DE2-70 kartı ile birlikte gelen standart modül uygulaması kullanılmıştır.

2.4.3.12 VGA denetleyici modülü

VGA Denetleyicisi 3.1.2 bölümünde anlatılan standart analog RGB senkronizasyon sinyallerini üretmektedir. Bu tez çalışmasında 640 x 480 @60Hz analog RGB video çıkışı kullanılmıştır. VGA Denetleyici modülü olarak DE2-70 kartı ile birlikte gelen standart modül uygulaması kullanılmıştır.

2.5 Literatür Araştırması

Görüntü işleme üzerine FPGA kullanılarak gerçekleştirilen pek çok çalışma vardır.

Cho, Mirzaei, Oberg, ve Kastner (2009), çalışmalarında Haar tanımlayıcılarını kullanarak FPGA tabanlı yüz tanıma sistemi geliştirmişlerdir (Cho et al., 2009).

Draper, Beveridge, Böhm, Ross ve Chawathe (2003). FPGA'ler için görüntü işleme algoritmalarının hızlandırılması konusunda çalışmışlardır.

Nelson (2000), çeşitli görüntü işleme tekniklerini FPGA üzerinde uygulayarak aynı kaynak resim için FPGA çıktısı ile aynı algoritmayı kullanan MATLAB programının çıktısını karşılaştırmış ve FPGA'in aynı sonuçları verdiğini göstermiştir. Kimi durumlarda FPGA'in sonuca daha çabuk ulaştığını belirtmiştir.

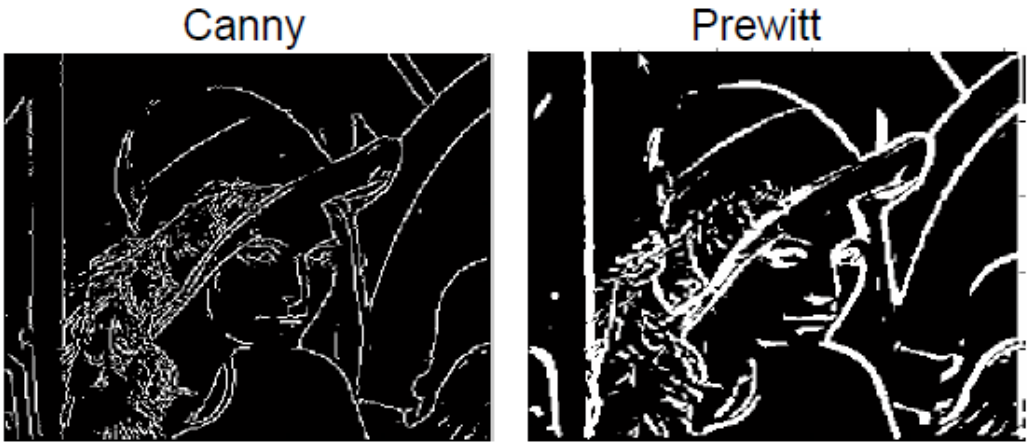
Günay (2010), FPGA üzerinde analog kompozit video görüntüsünün sayısal VGA formatına dönüştürülerek ekrana basılması, görüntü üzerinde plaka yeri saptama ve görüntü iyileştirme konusunda çalışmıştır.

Şekil 2-7 FPGA kullanarak plaka tanıma sistemi geliştirilmesi (Günay, 2010)

Zhengyang, Wenbo ve Zhilei (2010) Sobel operatörü kullanarak FPGA üzerinde gerçek zamanlı kenar tespiti gerçekleştirmiştir.

Şekil 2-8 Sobel operatörü kullanılarak kenar tespiti uygulaması (Zhengyang et al., 2010)

Benzer şekilde Hong ve Asher (2005). gerçek zamanlı video işleme için Canny ve Prewit algoritmalarını kullanarak adaptif kenar tespitini gerçekleştirmişlerdir.



Şekil 2-9 Canny ve Prewit operatörleri kullanılarak kenar tespiti uygulaması (Hong et al., 2005)

Karthigaikumar ve Baskaran (2012) FPGA kullanarak gerçek zamanlı resim içerisine görünmez filigran (watermark) eklemiş ve gizlenmiş filigranı tekrar birleştirilmiş görüntüden ayırabilmişlerdir.

Şekil 2-10 Görüntü içerisine görünmez filigran eklenmesi ve geri okunması uygulaması (Karthigukumar and Baskaran, 2012)

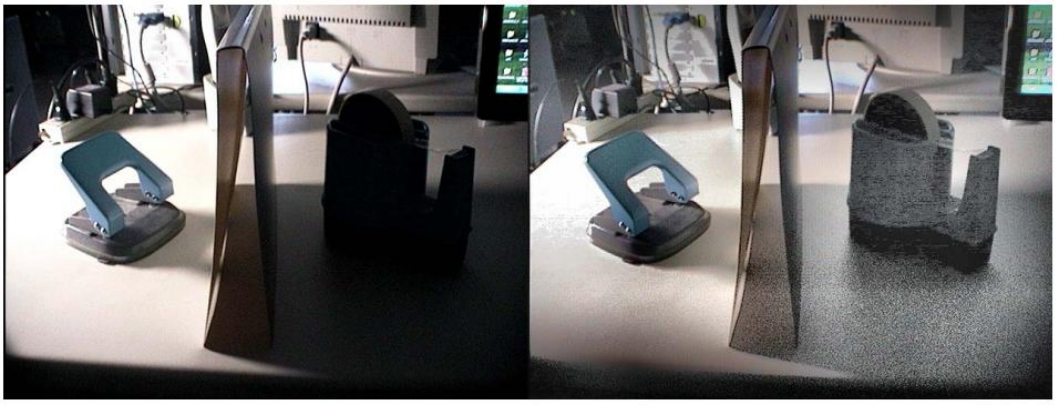
Gacar (2009), Güvenlik sistemlerinde kullanılmak üzere FPGA tabanlı görüntü arabirimi tasarlamış ve çalıştırmıştır.

Kelmelis et al. (2010) FPGA ve Grafik İşleme Birimlerini (GPU) yüksek çözünürlüklü video işleme uygulamaları açısından karşılaştırmış ve FPGA'lerin aynı işi daha küçük boyutlarda ve daha düşük güç tüketimi ile yapabildiklerini göstermiştir.

Özkan (2010), FPGA kullanarak trafik işaretlerini tanıyan bir sistem geliştirmiştir.

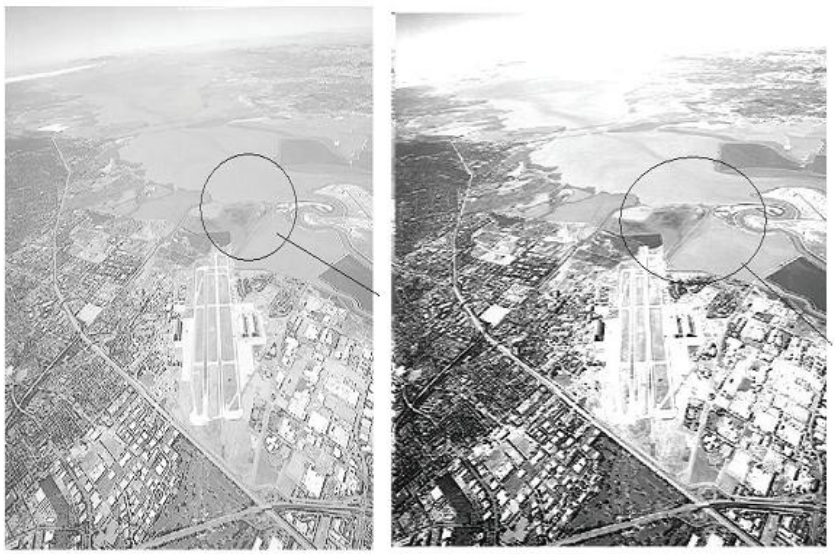
Angelopoulou et al. (2009) gerçek zamanlı, yüksek çözünürlüklü hareketli resimlerde FPGA kullanarak gürültünün iyileştirilmesi üzerine çalışmıştır.

Kokufata ve Tsutomu (2009) FPGA kullanarak gerçek zamanlı yerel karşıtlık iyileştirme üzerinde çalışmışlardır.

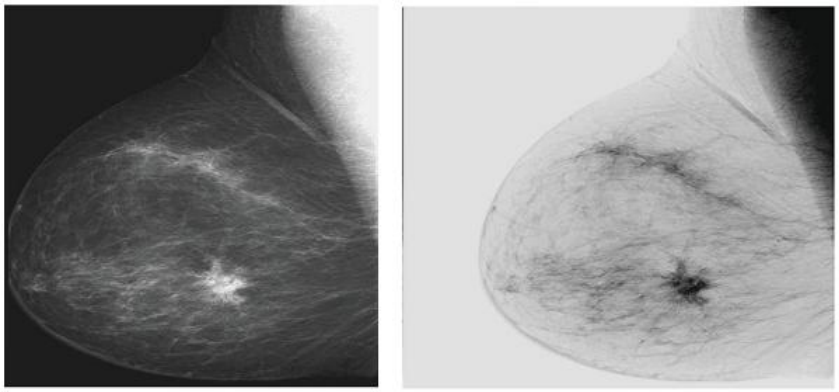


Şekil 2-11 FPGA tabanlı gerçek zamanlı yerel kontrast iyileştirme uygulaması (Kokufata and Tsutomu, 2009)

Kumar, Aditya ve Srivastava (2009) negatif alma, laplace filtreleme ve güç arttırma vb. gibi bir kısım görüntü iyileştirme tekniklerini FPGA üzerinde gerçeklemişlerdir.



Power Law Application



Negative Image

Şekil 2-12 FPGA görüntü iyileştirme uygulamaları (Kumar et al., 2009)

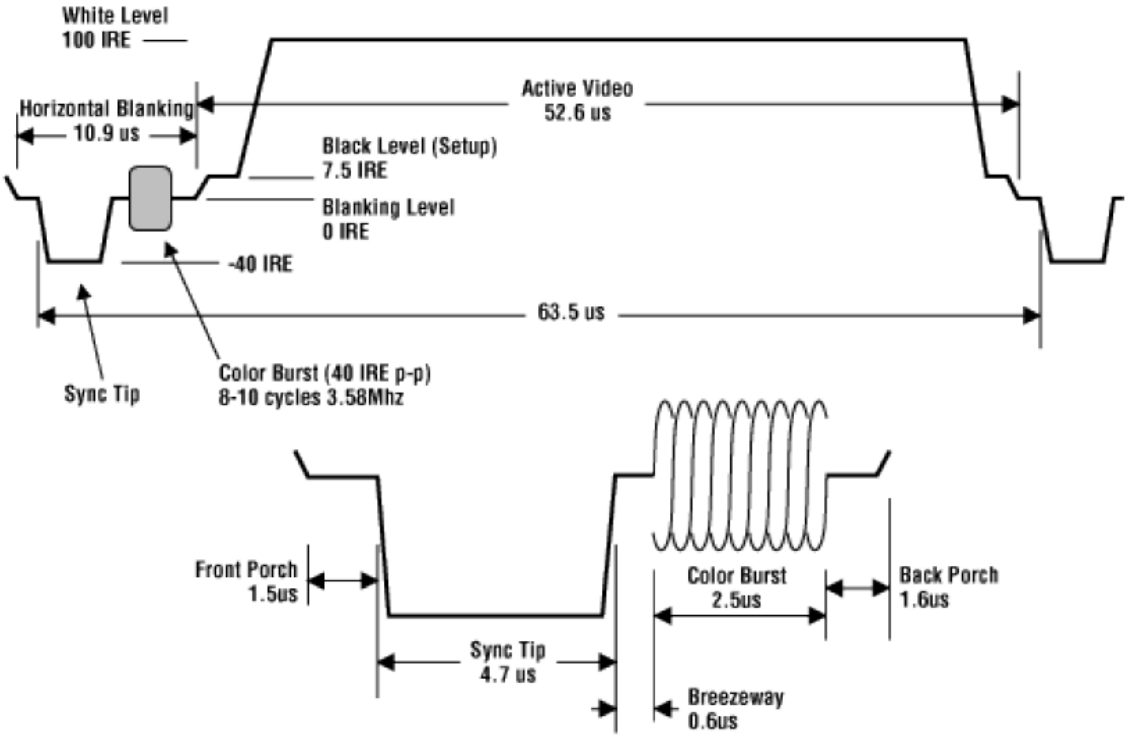
3. TEMEL VİDEO BİLGİLERİ

3.1 Analog Video

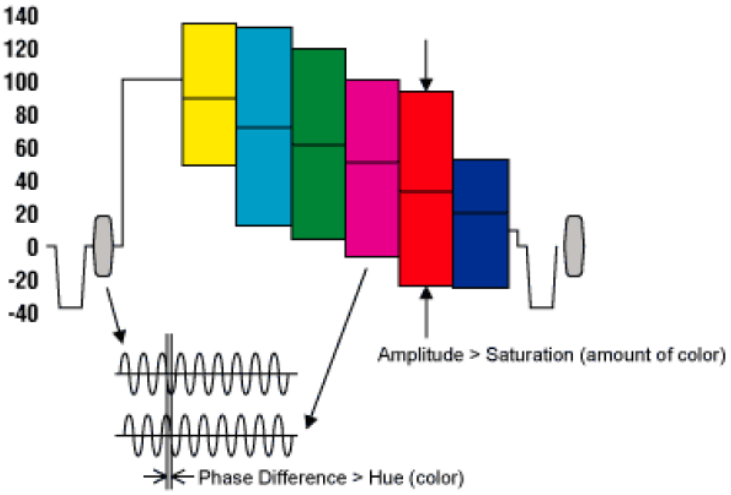
3.1.1 Kompozit video sinyali (CVBS)

Video hareketli resim olarak tanımlanır (Maxim, 2001). Sabit resimlerin arka arkaya insan gözünün algılayabileceğinden daha hızlı (en az saniyede 15 adet resim) gösterilmesi ile hareketli resim elde edilir.

İlk geliştirildiğinde video yayınları sadece siyah-beyaz görüntü bilgisinden (“Y” ile simgelenir) oluşuyordu. Renkli yayın sistemleri geliştirildiğinde mevcut altyapıyı bozmadan renkli video bilgisinin nasıl dağıtılacağı tartışıldı. Renk bilgisi “Kırmızı”, “Yeşil” ve “Mavi” renk bileşenlerinden (RGB) oluşur. RGB bileşenlerinin belirli oranlarda karıştırılması ara renkleri meydana getirir. Siyah-beyaz yayının üzerine RGB verisini eklemek kullanılan bant genişliğini üç kat arttırmak anlamına gelir. Dolayısı ile mevcut altyapıyı bozmamak adına alternatif çözüm arayışlarına girildi. Problem üç farklı sinyali (siyah-beyaz video verisini taşıyan “Y” ve renk bilgisini taşıyan “R-Y” ve “G-Y” sinyalleri) mevcut frekans bandı aralığında gönderilmesi için bir video sinyali formunun geliştirilmesi ile çözüldü. CVBS (Composite Video Baseband Signal) olarak adlandırılan kompozit video sinyali bugün NTSC, PAL, SECAM gibi farklı standartların temelini oluşturmaktadır. Kompozit video sinyali farklı bileşenlerden oluşan karmaşık bir yapıdadır. Sinyal şekli Şekil 3-1ve Şekil 3-2’de açıklanmıştır.



Şekil 3-1 Detaylı CVBS sinyali (Maxim, 2001).



Şekil 3-2 CVBS sinyali (Maxim, 2001).

Günümüzde video bilgisini göstermenin farklı metotları olsa da en temelde tüm bu yöntemler RGB bilgisinin matematiksel ifadelerinden oluşur (Keith, 2001).

3.1.1.1 Binişimleştirme ve binişimsizleştirme

Alıcıda görüntü oluşturulurken ekrandaki her bir piksel için ayrı bir veri gönderilir. 800x600 çözünürlüğü olan bir videoyu oynatmak için: (hareket katmak için saniyede en az 15 kare resim görüntülenmek zorundadır) 15x800x600x3 (her pikselde 3 renk bileşeni-RGB var) = 21.600.000 adet işlem yapılmalıdır. Tarama işlemlerinin gerçekleşme mantığı anlatıldığı gibi her bir pikselin oluşturulmasından ibarettir. Bu işlem sol üst köşeden başlar ve sağa doğru ilerler daha sonra bir alt satıra inip yine soldan sağa tekrar tarama yapar. Ancak bu teknolojinin icat edildiği tarihte kullanılan donanım bu sıklıktaki işlemlere yeterince hızlı tepkime veremiyordu. Kısaca görüntüyü bir anda ekrana vermek imkansızdı. Bu sorunu çözmek için her görüntüyü tek bir seferde görüntülemek yerine (satır:1,2,3...) bir satır atlayarak tarama yapması (satır: 1,3,5...) bir sonraki taramada da atlanan satırların (satır: 2,4,6...) taranması uygulandı. Bu sistem bir görüntünün iki farklı tarama sayesinde oluşmasını sağladı ve bu aradaki kısa beklemeler sayesinde donanım noktaları kendilerini yenileyecek zaman bulabildi. Bu sayede akıcı görüntüler elde edilebilidi. Ayrıca bu yöntem bant genişliğinin de yarıya inmesini sağladı.

Bu şekilde yapılan birbirine geçirerek tarama şekline "binişimleştirme (interlace scanning)" denir.

Binişimleştirme tekniğinde satırlar iki ayrı gruba ayrılırlar. Birinci grup satırlara (1,3,5..) "üst satırlar" ya da "tek satırlar" ikinci gruba da "alt satırlar" ya da "çift satırlar" denir. Kısaca alıcı seti içerisinde iki farklı grup ardışık olarak görüntülenir. Yani üst satırlardaki görüntünün ekranda sönmesi beklenirken alt satırlar görüntülenmeye başlanır. Bu iki görüntü arasındaki geçiş farkı çok kısa olduğu için beynimiz bu iki grubu birleştirip anlaşılır bir görüntü elde eder.

Şekil 3-3 Binişimsizleştirme (de-interlacing) işlemi

Günümüzde kullanılan donanımlar çok daha hızlı tepki verebilmektedir ve binişimleştirme işlemine ihtiyaç duymamaktadır. Binişimleştirme işleminin geri alınmasına "binişimsizleştirme (de-interlacing)" denir. Bu uygulamada gelen "tek satırlar" ve "çift satırlar" grupları hafızada tutulur ve sıralama düzgün olacak şekilde tüm ekran bilgisi yeniden düzenlenir. Şekil 3-3'de binişimsizleştirme işleminin nasıl çalıştığı gösterilmiştir. Soldaki resim sadece üst satırları gösterirken sağdaki resim iki grubun birleşmesiyle oluşan resmi göstermektedir.

3.1.1.2 PAL

Açılımı "Phase Alternating Line" (Değişebilen Faz Hattı) olan PAL'ın kullanıldığı bölgeler Avrupa, Avustralya, Güney Amerika ve Orta Asya'dır. Bu sistem ilk olarak Almanya'da ortaya çıkmıştır ve Avrupa elektrik sistemine uygun olması için tarama hızı 50hz olarak belirlenmiştir. PAL sistemi saniyede 25 bütün görüntü oluşturmaktadır. PAL sistemlerde görüntü 625 satırdan oluşmaktadır. Bunun 576 satırı canlı görüntü, kalanı ise senkronizasyon ve karartma işlemleri için kullanılır. PAL, NTSC sistemine göre daha az tarama oranına rağmen PAL yayınların görüntü kalitesi (çözünürlüğü) daha yüksektir. Ayrıca PAL yayınlarının oluşmasındaki bir diğer sebep de NTSC'de oluşan renk problemlerini ortadan kaldırmak içindir.

3.1.1.3 NTSC

NTSC'nin açılımı National Television System Committee (Ulusal Televizyon Sistemleri Komitesi)'dir. Bu sistem Kuzey Amerika, Kanada, Meksika ve Japonya'da kullanılmaktadır. Bu sistem standardı ilk defa Amerika'da kullanılmaya başlanmıştır ve Amerika'nın elektrik sistemine uygun olabilmesi

için ekran tarama hızı 60Hz olarak belirlenmiştir. Tarama hızının 60Hz olması saniyede 60 kez ekranın taranması demektir. Binişimleştirme tekniğinde her bir görüntü yaratılması için iki kez tarama yapılması gerektiğine göre (üst satırlar & alt satırlar) NTSC yayınlarında her saniyede 30 bütün görüntü oluşmaktadır. NTSC sistemlerde görüntü 525 satırdan oluşmaktadır. Bunun 480 satırı canlı görüntü, kalanı ise senkronizasyon ve karartma işlemleri için kullanılır.

Tez çalışmasında NTSC formatındaki kompozit video girişi kaynak olarak kullanılmıştır.

3.1.2 Analog RGB video sinyali

Analog RGB sinyalinde renk bilgileri ve senkronizasyon bilgileri ayrı kanallardan aktarılır. Renk bilgileri "R (Kırmızı)" "G (Yeşil)" ve "B (Mavi)" olarak isimlendirilen kanallardan, dikey senkronizasyon "V" olarak isimlendirilen kanaldan, yatay senkronizasyon "H" olarak isimlendirilen kanaldan aktarılır (Maxim, 2001).

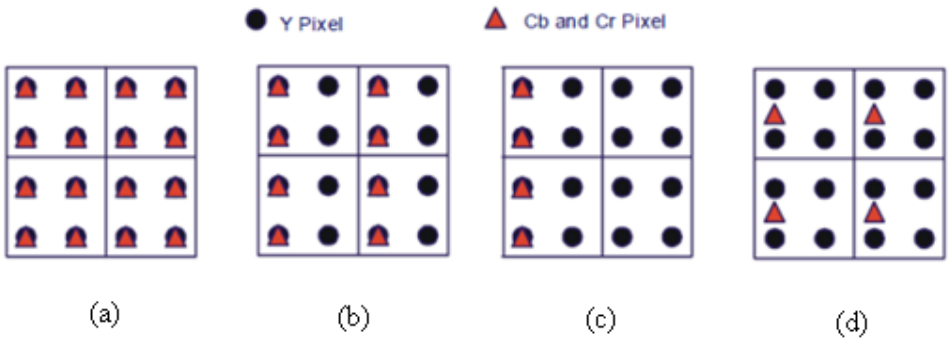
Tez çalışmasında Video çıkışı olarak analog RGB sinyali kullanılmıştır.

3.2 Sayısal Video

En sık kullanılan sayısal video sinyalleri RGB ve YCbCr'dır. RGB basitçe analog RGB video sinyalinin sayısallaştırılmış halidir. YCbCr ise analog YPbPr sinyalinin (RGB'den matematiksel olarak türetilmiş bir sinyal tipi) sayısallaştırılmış halidir (Keith, 2001).

3.2.1 YCbCr

YCbCr renk düzlemi dünya çapında sayısal video standardı geliştirilmesi çalışmalarının (ITU-R BT.601 çalışması) bir parçası olarak oluşturulmuştur. 8-bit için, "Y" 16 – 235 değerleri aralığında, "Cb" ve "CR" ise 16-240 değerleri aralığındadır. Birkaç farklı YCbCr örnekleme formatı bulunmaktadır. Bunlar: 4:4:4, 4:2:2, 4:1:1 ve 4:2:0'dir (Keith, 2001).



Şekil 3-4 YCbCr Örnekleme (Wang, 2004)
(a) YCbCr 4:4:4 örnekleme, (b) YCbCr 4:2:2 örnekleme, (c) YCbCr 4:1:1 örnekleme, (d) YCbCr 4:2:0 örnekleme,

3.2.1.1 YCbCr 4:4:4 örnekleme

Şekil 3-4 (a) YCbCr 4:4:4 formatını gösterir. Her örnekleme "Y", "Cb" ve "Cr" değeri içerir. Her örnek 8-bit veya 10-bit olabilir. Dolayısı ile sinyal 24-bit veya 30-bit veriden oluşur (Keith, 2001). Tez çalışmasında 24-bit YCbCr 4:4:4 örnekleme yöntemi kullanılmıştır.

3.2.1.2 YCbCr 4:2:2 örnekleme

Şekil 3-4 (b) YCbCr 4:2:2 örnekleme yöntemini göstermektedir. Her iki adet yatay "Y" örneklemesine karşın, bir adet "Cb" ve "Cr" örneklemesi bulunur. Her örnek 8-bit veya 10-bit olabilir. Dolayısı ile sinyal 16-bit veya 20-bit veriden oluşur. YCbCr verisini görüntülemek için 16 veya 20 bit olarak gelen veri 24 veya 30 bitlik 4:4:4 YCbCr verisine dönüştürülür. Eksik olan "Cb" ve "Cr" verileri interpolasyon yöntemi ile oluşturulur (Keith, 2001). Tez çalışmasında 24-bit YCbCr 4:2:2 örnekleme yöntemi kullanılmıştır.

3.2.1.3 YCbCr 4:1:1 örnekleme

Şekil 3-4 (c) YCbCr 4:1:1 örnekleme yöntemini (YUV12 olarak da adlandırılır) göstermektedir. Her dört adet yatay "Y" örneklemesine karşın, bir adet "Cb" ve "Cr" örneklemesi bulunur. Her örnek 8-bit olabilir. Dolayısı ile sinyal 12-bit veriden oluşur. YCbCr verisini görüntülemek için 12 bit olarak gelen veri 24 bitlik 4:4:4 YCbCr verisine dönüştürülür. Eksik olan "Cb" ve "Cr" verileri interpolasyon yöntemi ile oluşturulur (Keith, 2001).

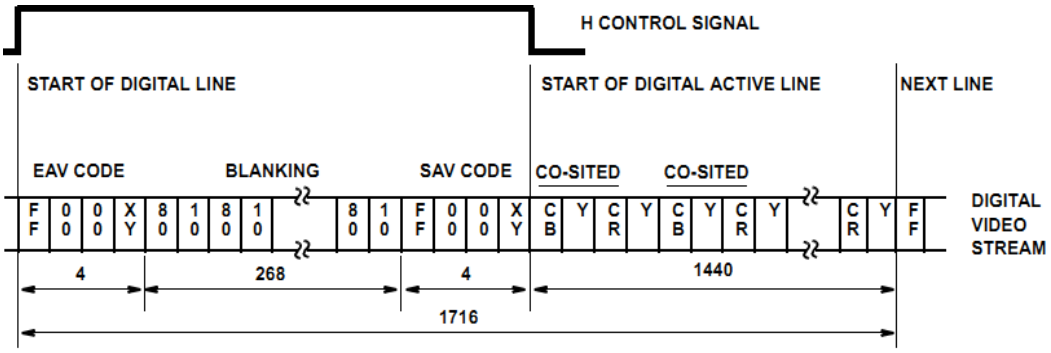
3.2.1.4 YCbCr 4:2:0 örnekleme

Yatay eksende yapılan 2:1'lik "Cb" ve "Cr" azaltmasını kullanan 4:2:2 örnekleme yönteminden farklı olarak 4:2:0 örneklemesi hem yatay hem de dikey eksenlerde "Cb" ve "Cr" azaltmasını kullanır (Şekil 3-4 (d)). Sıklıkla video sıkıştırma amacı için tercih edilmektedir. YCbCr verisini görüntülemek için gelen veri 4:4:4 YCbCr verisine dönüştürülür. Eksik olan "Cb" ve "Cr" verileri interpolasyon yöntemi ile oluşturulur. Ancak tüm veri geri getirilemediğinden ekranda renk hataları oluşmaktadır (Keith, 2001).

3.2.2 ITU-R BT 656 YCbCr video formatı

ITU656 sayısal video formatı NTSC ve PAL televizyon standartlarını sayısallaştırmak için geliştirilmiştir. ITU656 4:2:2 YCbCr örneklemesini kullanır. ITU656 paralel ara yüzü 8 veya 10 bit YCbCr veris kullanır. Çalışma frekansı 27MHz'dir.

4:2:2 YCbCr verisi "Cb0Y0Cr0Y1Cb2Y2Cr2…" şeklinde karıştırılarak Şekil 3-5'deki gibi gönderilir.



Şekil 3-5 Tek bir satır için ITU-R BT.656 veri akışı (Intersil, 2002)

EAV (aktif videonun sonu) kodu gönderildikten sonra bir satır tamamlanır ve yeni satır SAV (aktif videonun başlangıcı) kodu ile başlar. Bu şekildeki sayısal kodlama tekniği ile video senkronizasyon sinyalleri (Dikey ve yatay senkronizasyon) sayısal veri akışının içine gömülü biçimde gönderilir. 4-bitlik EAV ve SAV kodlarının ilk üç biti sabit değer iken dördüncü bit gönderilen alan bilgisini ve yatay ve dikey bekleme süresini belirtir.

Her başarılı CbYCr grubu bir piksele karşılık gelir ve grup dışında kalan "Y" bir sonraki piksele aittir. Bu nedenle bu standart YCbCr 4:2:2 olarak adlandırılmaktadır. YCbCr verisini görüntülemek için gelen 4:2:2 verisi 4:4:4

YCbCr verisine dönüştürülür. Eksik olan “Cb” ve “Cr” verileri bir sonraki “Y” sinyaline kopyalanarak piksel bilgisi oluşturulur. (Intersil, 2002).

Sayısal YCbCr - RGB verileri için renk düzlemi dönüştürülmesi işlemleri aşağıda verilen denklemdeki ifadeler doğrultusunda yapılır (Wang, 2004).

$$Y_d = 0.257 R_d + 0.504 G_d + 0.098 B_d + 16,$$

$$C_b = -0.148 R_d - 0.291 G_d + 0.439 B_d + 128,$$

$$C_r = 0.439 R_d -0.368 G_d - 0.071 B_d + 128,$$

$$R_d = 1.164 Y_d' + 0.0 C_b' + 1.596 C_r',$$

$$G_d = 1.164 Y_d' - 0.392 C_b' -0.813 C_r',$$

$$B_d = 1.164 Y_d' + 2.017 C_b' + 0.0 C_r',$$

$$Y_d' = Y_d - 16$$

$$C_b' = C_b - 128$$

$$C_r' = C_r - 128$$

Tez çalışmasında analog NTSC kompozit video kaynağı ITU-R BT.656 formatında YCbCr 4:2:2 olarak sayısallaştırılmış, sayısallaştırılan bu veri önce interpolasyon yöntemiyle YCbCr 4:4:4 olarak düzenlenmiş, daha sonra RGB formatına çevrilerek Bölüm 5’da bahsedilen algoritmalar uygulanmıştır. Elde edilen işlenmiş resim analog RGB formatına çevrilerek harici monitöre gönderilmiştir.

4. TEZ ÇALIŞMASINDA KULLANILAN SAYISAL GÖRÜNTÜ İŞLEME TEKNİKLERİ

Sayısal görüntü sayılardan oluşan bir matris olarak ifade edilebilir. Sayısal görüntü oluşturulması iki işleme dayanır: uzamsal örnekleme ve ölçeklendirme. İlk işlem olan uzamsal örnekleme analog görüntünün taranmasına benzer. İkinci süreç, ölçeklendirme, analog-sayısal çevrime dayanır. Görüntüdeki her uzamsal noktaya, noktadaki enerji miktarı oranında bir değer atanır. Bu iki sürece sayısallaştırma denir (Dorf, 2000). Şekil 4-1 sayısallaştırma sürecini açıklamaktadır.

Şekil 4-1 Sayısallaştırma süreci.

Sayısal görüntü işleme tekniklerinin tümü oluşturulan matris üzerinde işlemlerin yapılmasına dayanmaktadır.

Görüntü işleme teknikleri kullanılarak resimdeki belirli tipteki bozulmaların düzeltilmesine görüntü iyileştirme denir. Görüntü iyileştirme gürültünün azaltılmasıyla, kenarları daha belirgin hale getirerek veya resme daha güzel görünmesi için başka bir operasyonun uygulanması ile gerçekleştirilir.

İyileştirme için kullanılan algoritmaların çoğu "Pencere Operasyonları" olarak tanımlanan piksel bazlı fonksiyonlardır. Bir pencere operasyonu, görüntü matrisinin herhangi bir bölgesinde bir grup piksel üzerine (pencere içinde kalan kısma) uygulanan matematiksel işlemlerdir. Pencere boyutu 3x3, 9x9, 13x13 veya 21x21 olabilir. Buradaki yaklaşım küçük bir pencere kullanarak bölgesel veriyi iyileştirmek ve tüm resmi bu şekilde tarayarak ana görüntünün iyileştirilmesini sağlamaktır (Dorf, 2000).

4.1 Gürültü Azaltma

Gürültü azaltma sayısal görüntü matrisinde pürüzsüzleştirme (smoothing) işleminin uygulanması ile yapılabilir. Resimden 3x3 piksel boyutunda bir pencere

çıkardığımızı varsayalım. Bu 3x3 lük pencere içerisindeki değerlerin (a_1, a_2… a_9) ortalamasını alır ve bu ortalama değeri a_5 yerine pencereye yazarsak ve bu işlemi tüm resmi tarayarak, bütün pikseller için uygularsak resmin bütünlüğünü bozmadan resimdeki uç değerleri (gürültüleri) azaltmış oluruz. Bu algoritmanın yan etkisi olarak resimdeki keskin geçişlerde (kenarlarda) bulanıklaşma meydana gelir (Dorf, 2000).

3x3 lük pencere kullanıldığı için resmin kenarlarında (bir piksel boyutunda) iyileştirme yapılamamaktadır. Kenarlarda pürüzsüzleştirme işlemi komşu pikselin aynısını kenar piksel değeri olarak atayarak veya kenar değerini "0" vererek yapılmaktadır.

Bu operasyon kamera sensor hatası ve veri iletim kanallarındaki kayıptan kaynaklanan gürültüleri elimine etmekte kullanılır. Bu iki tip gürültü resimde parlak noktalar şeklinde ortaya çıkar ve uzamsal olarak birbirinden ayrık bölgelerde bulunur.

4.1.1 Konvolusyon operatörü

Bir önceki başlıkta bahsedilen pencere operasyonu piksel değerlerinin doğrusal ağırlıklı toplamını almaktır. a_5 yeni piksel değerimiz, b_n ise orijinal piksel değerlerimiz olsun. Bu durumda denklemimiz α_n matrisindeki değerler herhangi bir reel sayı olmak kaydıyla

$$a_5 = \sum_{n=1}^{9} (b_n \alpha_n)$$

olacaktır.

Dolayısı ile bir önceki bölümde anlatılan pürüzsüzleştirme uygulaması için kullanılabilecek α_n matrisi

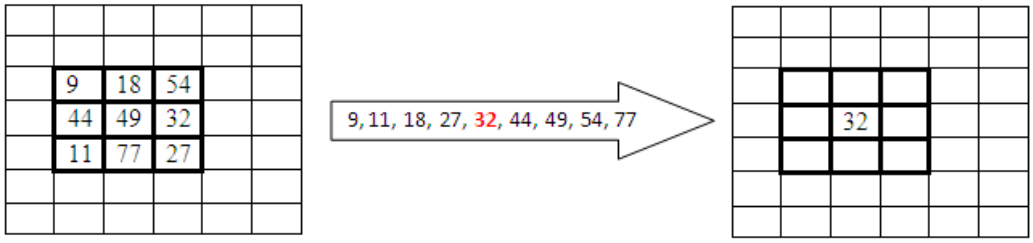
$$\alpha_n = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

şeklinde olacaktır.

α_n operatörünün ağırlıkları değiştirilerek farklı tipte iyileştirmeler yapılabilir. Bu yöntem ile yapılan işlemlere doğrusal pencere işlemleri veya konvolusyon operatör işlemleri denmektedir (Dorf, 2000).

4.1.2 Medyan filtre

Gürültü azaltma kısmında anlatıldığı gibi yine resimden 3x3 piksel boyutunda bir pencere çıkardığımızı varsayalım. Bu 3x3 lük pencere içerisindeki değerleri (a_1, a_2… a_9) büyükten küçüğe olacak şekilde sıralar ve ortadaki değeri (medyanı) a_5 yerine pencereye yazarsak ve bu işlemi tüm resmi tarayarak, bütün pikseller için uygularsak resmin bütünlüğünü bozmadan resimdeki uç değerleri (gürültüleri) azaltmış oluruz. Bu algoritma pürüzsüzleştirme işleminde olduğu kadar resimdeki keskin geçişlerde (kenarlarda) bulanıklaşma meydana getirmemektedir (Dorf, 2000). Şekil 4-2 medyan filtreleme tekniğini bir örnekle açıklamaktadır.



Şekil 4-2 Medyan filtre örneği

Bu tez çalışmasında gürültü azaltma algoritması olarak medyan filtre tekniği tercih edilmiştir.

4.2 Kenar Tespit Etme

Parlaklık seviyesine göre resimdeki nesnelerin kenar çerçevesinin tespit edilmesi sık başvurulan bir görüntü işleme uygulamasıdır. Kenar, sayısallaştırılmış bir resimde, göreceli olarak küçük bir bölgede parlaklık seviyesindeki büyük bir değişim olan alanlar olarak tanımlanabilir. Bu şekilde tanımlanan alanların tespit edilmesi için kullanılan algoritmaların çoğu öncelikle bir pencere operatörü ve konvolusyon matrisi kullanarak resimdeki büyük parlaklık seviyesi değişimi olan alanları belirginleştirir, sonrasında belirli bir eşik değerinden büyük olan bölgeleri kenar olarak atar. Bu konuda pek çok çalışma yapılmaktadır. Çalışmalar en uygun eşik değerinin bulunması, kullanılan

pencerenin boyutu, konvolusyon matrisinde kullanılan ağılıkların değeri gibi noktalarda yoğunlaşmıştır (Dorf, 2000).

Literatürde kullanılan çeşitli kenar tespit etme algoritmaları vardır. Canny, Sobel, Roberts, Prewitt bu algoritmalar arasında öne çıkmışlardır (Oskoei and Hu, 2010).

Bu tez çalışmasında kenarların tespit edilmesi için Prewitt kenar detektörü kullanılmıştır.

4.2.1 Prewitt kenar dedektörü

Prewitt kenar detektörü resimdeki eğimlerin (gradyant) büyüklüğünü bulmak için konvolusyon operatörü kullanır. "K" konvolusyon matrisinin "p" resmine uygulanması aşağıdaki gibi ifade edilir:

$$N(x,y) = \sum_{k=-1}^{1} \sum_{j=-1}^{1} K(j,k)p(x-j, y-k)$$

(Matthews, J. 2002)

Prewitt kenar detektörü dikey ve yatay eksenlerde olmak üzere iki farklı konvolusyon matrisi kullanır. "h_x" Dikey karşıtlığı, "h_y" ise yatay karşıtlığı belirler (Matthews, J. 2002).

$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad h_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

(Oskoei and Hu, 2010)

Bu maskeler (h_x ve h_y) kullanılarak veri bir vektör olarak ifade edilir. Bu iki bileşen vektörün "x" ve "y" bileşenleri olarak düşünülürse vektörün büyüklüğü ve yönü

$$\mathbf{g} = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

(Maini and Aggarwal, 2010)

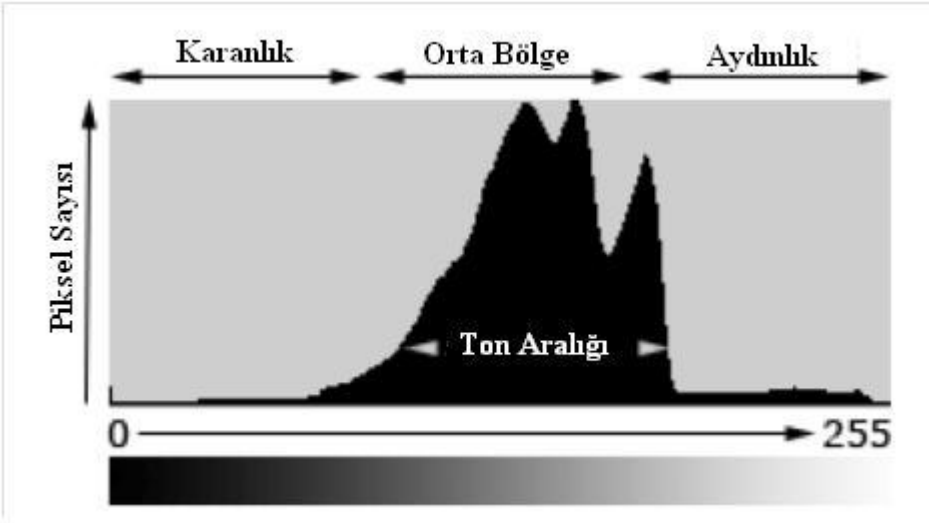
formülleri ile ifade edilebilir. Burada "**g**" gradyant vektörü, "g" gradyant büyüklüğünü ve "θ" da gradyant yönünü ifade eder.

4.3 Karşıtlık ve Parlaklık İyileştirme

Parlaklık (brightness) bir resimdeki piksellerin aydınlık seviyesini verir. Karşıtlık (kontrast) ise bir videoda siyah ve beyazın parlaklık seviyesine bağlıdır. Bir resmin karşıtlık değeri, beyaz parlaklığının, siyah parlaklığına oranıyla bulunur. Karşıtlığı iyileştirmek resimdeki karanlık kısımları daha karanlık, aydınlık kısımları daha aydınlık yaparak orta seviye parlak olan kısımların daha görünür hale gelmesini sağlayarak mümkün olur. Bunun için resmin aydınlık seviyesinin hesaplanmasına ihtiyaç duyulur. Bu hesaplama "histogram hesaplama" olarak adlandırılır.

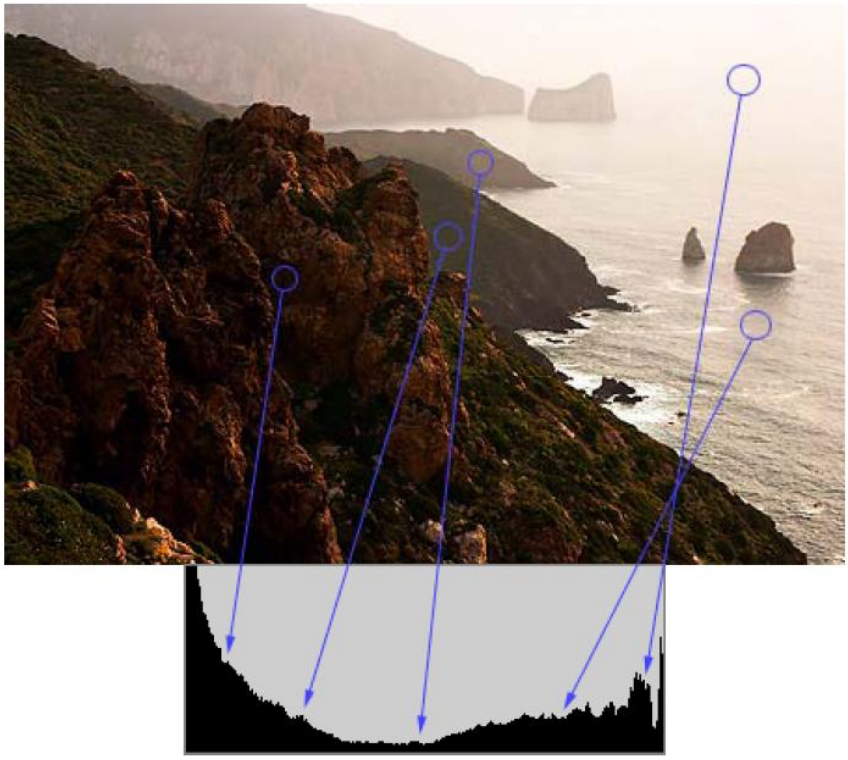
4.3.1 Histogram hesaplama

Histogram, bir resimdeki bütün piksellerin sahip oldukları gri-seviyelere göre grafiklendirilmiş gösterimidir. Mesela, 8-bit kullanılarak sayısallaştırılmış bir videoda, piksellerin sahip olabilecekleri gri-seviyeler 0-255 arasıdır.



Şekil 4-3 Histogram ton aralığı

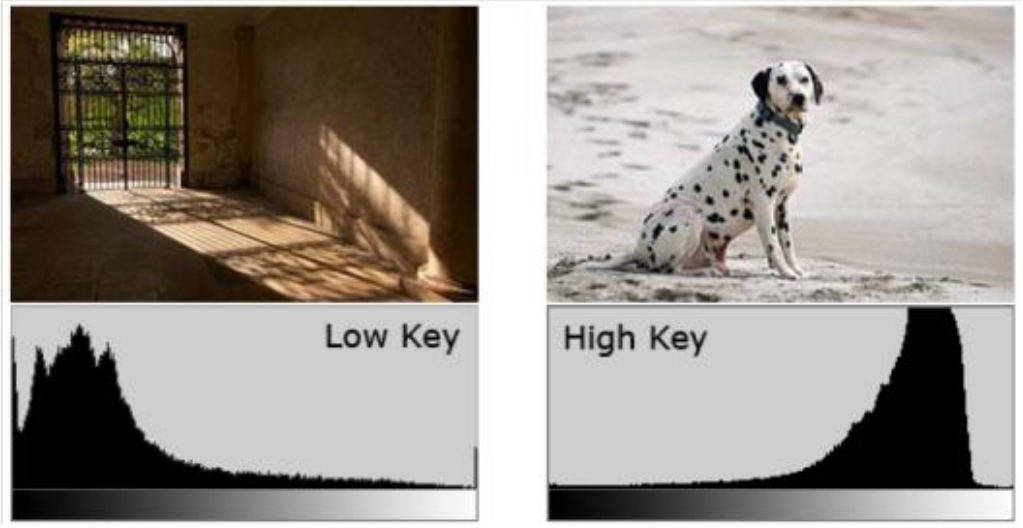
Ton Aralığı: Piksellerin yoğun halde bulunduğu, resmin genel aydınlık seviyesini belirleyen bölgeye verilen addır (Şekil 4-3).



Şekil 4-4 Karanlık ve aydınlık sahne içeren resim örneği

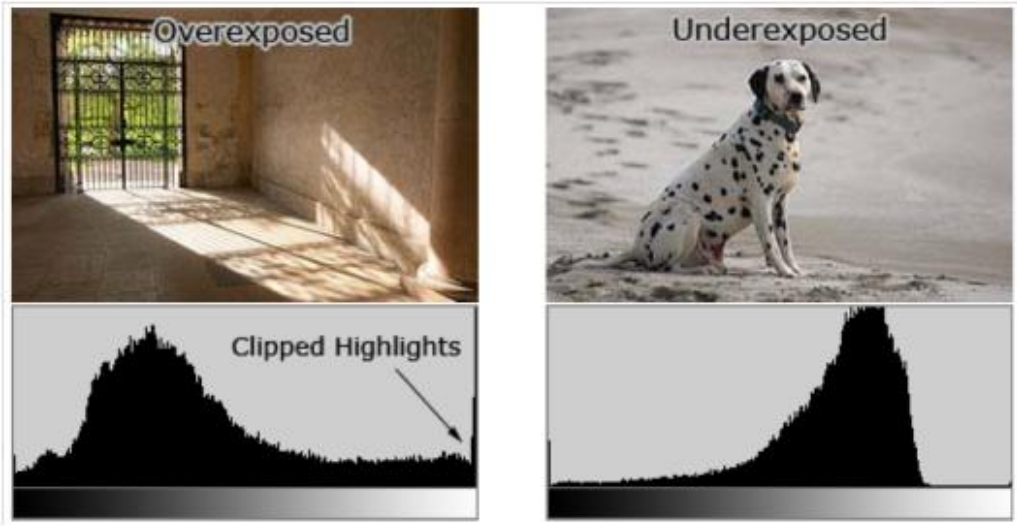
Çoğu kamera histogramı otomatik ayarlayıp çekim yapsa da, karşımıza sık sık çıkan karanlık veya aydınlık sahneler bu grup dışındadır (Şekil 4-4) .

Ton aralığı bölgesinin karanlık tarafta yer aldığı resimler “Düşük Cetvel (Low Key)”, aydınlık bölgede yer aldığı resimler ise “Yüksek Cetvel (High Key)” olarak adlandırılır (Şekil 4-5).



Şekil 4-5 Yüksek ve düşük cetvel gösterimi

Böyle resimlerde karşıtlık ayarı düzgün yapılmazsa, gölge kısımlarda yer alan ayrıntılar siyahın, parlak kısımdaki ayrıntılarsa beyazın içine gömülür. Bunu önlemek için kamera kısmında önlemler alınmıştır. Histogram otomatik ayarlanarak, filmin ayrıntıları kaçırması engellenir. Şekil 4-6'da solda parlatılmış (Overexposed) resim, sağda karartılmış (Underexposed) resim görülmektedir.

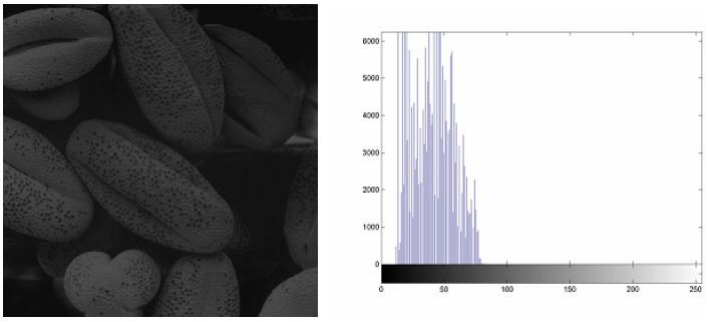


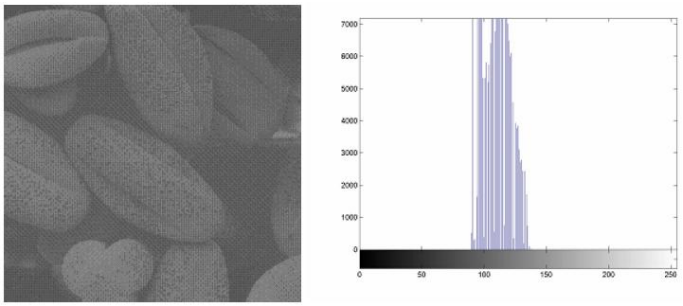
Şekil 4-6 Parlatılmış ve karartılmış resim

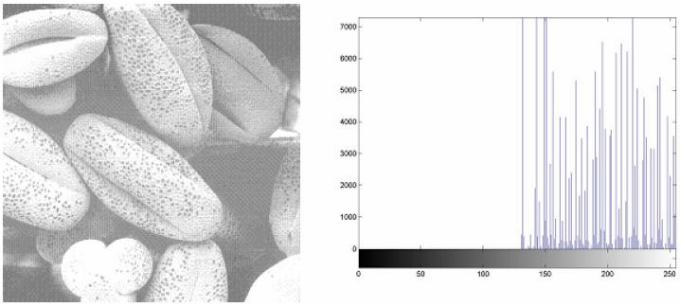
Böylece resmin histogramı gri-seviye eksenine yayılarak, daha iyi bir karşıtlık elde edilir (Demiryürekli, 2009).

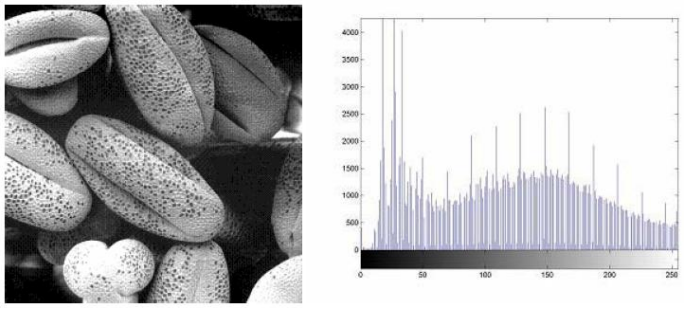
4.3.1.1 Histogram dağılımı:

İyi bir karşıtlık, piksel gri-seviye değerlerinin histogram eksenine eşit bir dağılım yapmasıyla sağlanabilir. Şekil 4-7'de karşıtlığı, histogram değerleriyle oynanarak arttırılan bir resim gösterilmiştir.


(a)


(b)


(c)


(d)

Şekil 4-7 Karşıtlık – histogram ilişkisi

(a) Çoğunluk karanlık bölgede, (b) Çoğunluk ortada, (c) Çoğunluk parlak bölgede, (d) Eşit dağılımla, kontrast arttırılmış(Gonzalez and Woods, 2002)

4.4 Morfolojik Operasyonlar

Morfolojik operasyonlar boolean fonksiyonları ile yapılır. Üzerinde çalışılan resim "f", boolean fonksiyonu "h" ve konvolusyon penceresi "B" için morfolojik operasyon uygulanmış $g = \boldsymbol{h(}$ f) olarak kabul edilirse;

G(n) = h[Bf(n)]

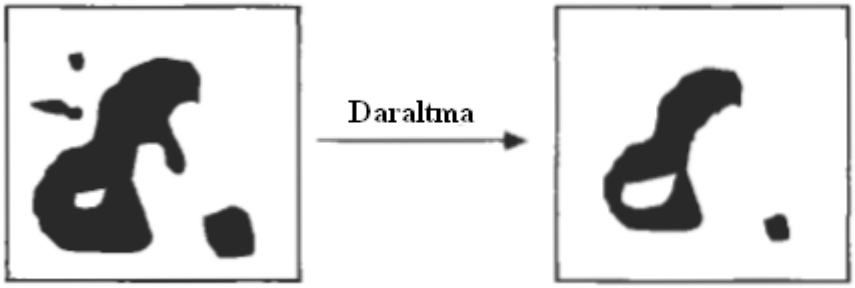
olarak tanımlanır. “n” tüm resim taranana kadar konvolusyon penceresinin adım sayısını belirtir. Morfolojik operasyonlar resmi genleştirmek, daraltmak, pürüzsüzleştirmek veya çok küçük imgeleri elemek amacı ile kullanılır (Bovik, 2000).

4.4.1 Daraltma

Daraltma (erosion) operasyonu “AND (ve)” mantık operatörü kullanılarak yapılır. g(n) = AND[Bf(n)] olarak tanımlanır.

Daraltma operasyonu resimdeki objeleri daha belirgin hale getirmek ve hareketli resimde oluşan gölgeleri yok etmek amacı ile kullanılır (Bovik, 2000).

Daraltma operasyonu Şekil 4-8’de grafiksel olarak açıklanmıştır.



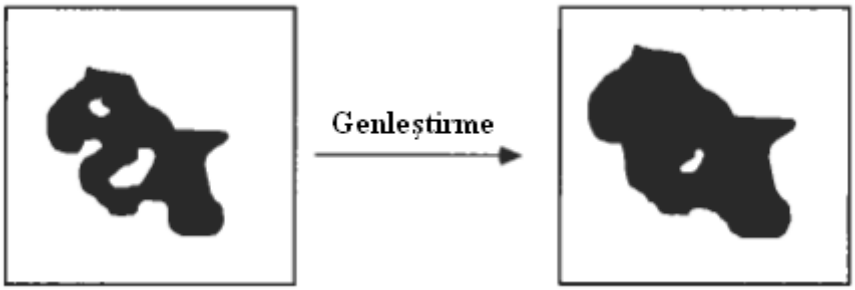
Şekil 4-8 Daraltma işlemi (Bovik, 2000).

4.4.2 Genleştirme

Genleştirme (dilation) operasyonu “OR (veya)” mantık operatörü kullanılarak yapılır. g(n) = OR [Bf(n)] olarak tanımlanır.

Genleştirme operasyonu resimde ön planda olan objeleri ki objeleri daha belirgin hale getirmek ve nesnelerin sınırlarını yuvarlatmak için kullanılır (Bovik, 2000).

Genleştirme operasyonu Şekil 4-9’da grafiksel olarak açıklanmıştır.



Şekil 4-9 Genleştirme işlemi (Bovik, 2000).

5. GÖRÜNTÜ İŞLEME UYGULAMALARI

FPGA üzerinde gerçeklenen görüntü iyileştirme ve işleme uygulamaları bu bölümde detaylı olarak anlatılmıştır. Kullanılan algoritmalar kaba kod (pseudo code) olarak verilmiştir. Algoritma çıktıları gösterilmiş ve sonuçlar kısaca özetlenmiştir.

5.1 Gürültü Azaltma

4.1 kısmında detayları anlatılan "Medyan Filtreleme" tekniği tez çalışmasında gürültü azaltma metodu olarak kullanılmıştır.

5.1.1 Medyan filtreleme tekniği kullanarak gürültü azaltma

Medyan filtre kullanarak gürültü azaltma uygulamasının DE2-70 kartında gerçeklenmesi algoritması Şekil 5-1 ve Şekil 5-2'de verilmiştir.

```
1   MEDIAN [iData, iPb, iPr, iCLOCK, RESET,
2           iKEY, oData, oPb, oPb]
3       //3x3'lük matris için 3 satır YPbPr
4       //bilgisi hafızaya alınır
5       LINEBUFFER_3 u0 (iCLOCK, iData, X0, X1, X2)
6       LINEBUFFER_3 u1 (iCLOCK, iPb, Pb0, Pb1, Pb2)
7       LINEBUFFER_3 u2 (iCLOCK, iPr, Pr0, Pr1, Pr2)
8
9       if (iCLOCK yükselen kenarda ise)
10      begin
11      //medyan filtre için parametre atamaları yapılır
12          Y0=X0; Z0=Y0; Y1=X1; Z1=Y1; Y2=X2; Z2=Y2;
13      //renk bilgisi işlemler yapılırken 3 işlemci saati
14      //süresince geciktirilir
15          oPb1 = Pb1 ; oPb2 = oPb1 ; oPb = oPb2 ;
16          oPr1 = Pr1 ; oPr2 = oPr1 ; oPr = oPr2 ;
17      end
18      //3x3'lük matrisin medyanını bulmak için karşılaştırma
19      //modülü çağırılır
20      COMPARE u10 (X0,X1,X2, oX0,oX1,oX2,iKEY,iCLOCK,RESET)
21      COMPARE u11 (oX0,oX1,oX2,ooX0,ooX1,ooX2,iKEY,iCLOCK,RESET)
22      COMPARE u12 (Y0,Y1,Y2, oY0,oY1,oY2,iKEY,iCLOCK,RESET)
23      COMPARE u13 (oY0,oY1,oY2, ooY0,ooY1,ooY2,iKEY,iCLOCK,RESET)
24      COMPARE u14 (Z0,Z1,Z2, oZ0,oZ1,oZ2,iKEY,iCLOCK,RESET)
25      COMPARE u15 (oZ0,oZ1,oZ2,ooZ0,ooZ1,ooZ2,iKEY,iCLOCK,RESET)
26      //bulunan medyan değeri "oData" paremetresi ile
27      //çağıran modüle gönderilir
28      COMPARE u16 (ooZ0,ooY1,ooX2, min,oData,max,iKEY,iCLOCK,RESET)
```

Şekil 5-1 Medyan filtreleme algoritması ana modülü

```
COMPARE [iX0,iX1,iX2,oX0,oX1,oX2,iKEY,iCLOCK,RESET]

    //çıkış parametre atamaları yapılır
    oX0 = Min;
    oX1 = Mid;
    oX2 = Max;

    if (iCLOCK yükselen kenarda VEYA RESET düşen kenarda ise)
        if(RESET = 0)
            {Min, Mid, Max}=0;
        else if (iKEY)
            if(iX0<iX1)
            begin
                if(iX0<iX2)
                    if(iX1<iX2)
                        {Min, Mid, Max} = {iX0, iX1, iX2}    //123
                    else
                        {Min, Mid, Max} = {iX0, iX2, iX1}    //132
                else
                    {Min, Mid, Max} = {iX2, iX0, iX1}        //231
            end
            else
                if(iX0<iX2)
                    {Min, Mid, Max} = {iX1, iX0, iX2}        //213
                else if ( iX1<iX2)
                    {Min, Mid, Max} = {iX1, iX2, iX0}        //312
                else
                    {Min, Mid, Max} = {iX2, iX1, iX0}        //321
        else
            Mid=iX1;
```

Şekil 5-2 Medyan filtreleme algoritması karşılaştırma (COMPARE) alt modülü

Medyan filtreleme parlaklık bilgisi (Y[7:0]) üzerinde uygulanmıştır. Renk bilgisi (CbCr[15:0]) işlemler süresince geçen zamanı kompanze edecek şekilde bir gecikme uygulanarak çıkış modülüne yönlendirilmiştir.

Tüm ekran 3x3 lük bir matris tarafından taranarak bu matris içinde kalan dokuz pikselin medyanı bulunmaktadır. Bunun için dokuz piksel değeri küçükten büyüğe doğru sıralanmakta, ortadaki değer medyan olarak seçilerek 3x3'lük matrisin ortadaki piksel değeri ile değiştirilerek resimde pürüzsüzleştirme işlemi tamamlanmaktadır. Bu işlem yapılırken satır satır gelen resim verisinde üç satırın hafızada tutularak işlemlerin yapılması gerekmektedir. Bu işlem için kullanılan satır bilgisini hafızaya alma algoritması olarak DE2-70 kartı ile birlikte gelen standart "LineBuffer_4" modülü kullanılmıştır. Her yeni satırda hafızada tutulan satırlar bir kaydırılarak tüm resmin taranması sağlanmıştır. Bu tekniğin handikapı ilk satır, son satır, ilk sütun ve son sütun için medyan filtrelemenin uygulanamaz oluşudur. Ancak bu dört satır ve sütunun tüm resimdeki görünürlüğü ihmal

edilebilecek düzeyde olduğundan izleyici tarafından fark edilmesi mümkün değildir.

Medyan filtre DE2-70 kartındaki iSW[0] tuşu ile aktifleştirilebilir. Şekil 5-3'deki orijinal resim üzerine medyan filtre uygulanarak elde edilen sonuç Şekil 5-4'de gösterilmiştir.

Şekil 5-3 Orijinal resim.

Şekil 5-4 Medyan filtreleme uygulanmış resim.

Yapılan çalışmanın daha net bir şekilde değerlendirilebilmesi için ekranın yarısını orijinal resim, diğer yarısını ise medyan filtre uygulanmış kısım olarak gösterecek şekilde bir algoritma geliştirilmiştir. DE2-70 kartındaki iSW[1] tuşu ekranı yatay eksende ikiye bölerek ekranın sol tarafında medyan filtrelemeyi uygulayıp, sağ tarafına orijinal resmi gösterecek şekilde monitör çıkışını konfigüre etmektedir. Şekil 5-5'de bu durum gösterilmiştir.

Şekil 5-5 Ekranı yatay eksende ikiye bölerek medyan filtreleme uygulamasının etkisinin gösterilmesi.

Ekranı ikiye bölme algoritması Şekil 5-6'da verilmiştir.

```
yarimEKRAN [iCLOCK,iKEY,iY,iiY,iCb,iiCb,
        iCr,iiCr,oY,oCb,oCr,iX_Count]
    //parametre atamaları yapılır
    oY  = Y;
    oCb = Cb;
    oCr = Cr;

    if (iCLOCK yükselen kenarda ise)
        if (iKEY = 1 yapılarak yarım ekran modülü aktif edilir ise)
            if ( iX_Count<640/2)      //Sol Ekran:medyan filtrelenmiş video
                Y  = iY;
                Cb = iCb;
                Cr = iCr;
            else if (iX_Count>=640/2) //Sağ Ekran:orijinal video
                Y  = iiY;
                Cb = iiCb;
                Cr = iiCr;
        else if (iKEY = 0 yapılarak yarım ekran modülü kapatılır ise)
                Cb<=iCb;
                Cr<=iCr;              //Tüm Ekran:medyan filtrelenmiş video
                Y<=iY;
```

Şekil 5-6 Ekranı ikiye bölerek sol tarafa iyileştirilmiş resim, sağ tarafa orijinal resim basma algoritması.

Şekil 5-4 ve Şekil 5-5'den da anlaşılacağı gibi medyan filtrenin uygulanması resimdeki gürültüleri büyük oranda azaltmakla beraber resmin keskinliğini de bir miktar bozmaktadır.

5.1.2 Dinamik adaptif medyan filtreleme

5.1.1 adımında sonuçları verilen medyan filtreleme tekniğinin resim keskinliğini azalttığı yapılan çalışma sonucunda görülmüştür. Medyan filtreleme statik bir şekilde gelen resme uygulanırsa, resimde gürültü olsun olmasın tüm resmin keskinliği azalacaktır. Ancak resimdeki gürültüler tespit edilebilirse, bu koşut durumuna uygun olarak medyan filtreleme uygulama kararı verilebilir. Bunun sonucu olarak elde edilecek algoritma dinamik ve adaptif bir şekilde resimdeki varsa gürültüyü temizleyecek, yok ise girişi çıkışa verecektir. Bunun için tasarlanan algoritma Şekil 5-7'e verilmiştir.

Bu algoritmada yapılan işlem öncelikle resimdeki gürültüyü tespit etmektir. Gürültü tüm resme oranla küçük bir yer kaplayan ani renk veya parlaklık değişimleri olarak ifade edilebilir. Bu yaklaşımı referans alarak; yine tüm ekran 3x3 lük bir matris tarafından taranarak bu matris içinde kalan dokuz pikselin ortalama değeri ve bu matrisin medyanı bulunmuştur. Bulunan bu iki değer birbirinden çıkarılarak 3x3'lük matris için sapma değeri belirlenmiştir. Sapma eğer medyan değerinin %50'si veya daha büyük ise 3x3'lük matrise medyan

filtreleme tekniği uygulanmıştır. Eğer sapma değeri medyan değerinin %50'sinden daha küçük ise matriste değişiklik yapılmadan algoritma işletilmeye devam edilmiştir. Bu şekilde sapması yüksek olan, gürültü olarak ifade edilebilecek, pikseller ayıklanmıştır. 3x3'lük matris tüm ekran boyunca taranarak resmin tamamının dinamik filtrelenmesi sağlanmıştır.

Adaptif dinamik medyan filtre DE2-70 kartındaki iSW[0] tuşu ile aktifleştirilebilir. iSW[1] tuşu statik medyan filtreleme algoritmasını çağırmaktadır. iSW[2] tuşu yine karşılaştırma yapılabilmesi için ekranı ikiye bölerek ekranın sol tarafına statik veya dinamik medyan filtrelemeyi uygulayıp, sağ tarafına orijinal resmi gösterecek şekilde videoyu konfigüre etmektedir.

```
adaptiveMEDIAN  [iData, iPb, iPr,iCLOCK,
                 RESET,iKEY,oData,oPb,oPr]
    //3x3'lük matris için 3 satır YPbPr
    //bilgisi hafızaya alınır
    LINEBUFFER_3 u0 (iCLOCK, iData, X0, X1, X2)
    LINEBUFFER_3 u1 (iCLOCK, iPb, Pb0, Pb1, Pb2)
    LINEBUFFER_3 u2 (iCLOCK, iPr, Pr0, Pr1, Pr2)
    //parametre atamaları yapılır
    oData = çıkış_video;
    if (iCLOCK yükselen kenarda ise)
        //medyan filtre için parametre atamaları yapılır
        Y0=X0; Z0=Y0; Y1=X1; Z1=Y1; Y2=X2; Z2=Y2;
        //renk bilgisi işlemler yapılırken 3 işlemci
        //saati süresince geciktirilir
        oPb1 = Pb1 ; oPb2 = oPb1 ; oPb = oPb2 ;
        oPr1 = Pr1 ; oPr2 = oPr1 ; oPr = oPr2 ;
        //orijinal parlaklık bilgisi işlemler yapılırken
        //4 işlemci saati süresince geciktirilir
        iData1 = X1; iData2 = iData1;
        iData3 = iData2; orijinal_video = iData3;
        ortalama_parlaklık_bilgisi = (X0 + X1 + X2) / 3;
        if (ortalama_parlaklık_bilgisi > Medyan)
            standart_sapma = ortalama_parlaklık_bilgisi - Medyan;
        else if (ortalama değer < Medyan)
            standart_sapma = Medyan - ortalama_parlaklık_bilgisi;
        if (çıkış konfigürasyon tuşları konumu iKEY[1:0] = "10" ise)
            çıkış_video = Medyan;
        else if (çıkış konfigürasyon tuşları konumu iKEY[1:0] = "01" ise)
                if (standart_sapma > (Medyan / 2))
                    çıkış_video = Medyan;
                else
                    çıkış_video = orijinal_video;
        else
            çıkış_video = iData;
    //3x3'lük matrisin medyanını bulmak için
    //karşılaştırma modülü çağrılır
    COMPARE u10 (X0,X1,X2, oX0,oX1,oX2,iKEY,iCLOCK,RESET)
    COMPARE u11 (oX0,oX1,oX2,ooX0,ooX1,ooX2,iKEY,iCLOCK,RESET)
    COMPARE u12 (Y0,Y1,Y2, oY0,oY1,oY2,iKEY,iCLOCK,RESET)
    COMPARE u13 (oY0,oY1,oY2, ooY0,ooY1,ooY2,iKEY,iCLOCK,RESET)
    COMPARE u14 (Z0,Z1,Z2, oZ0,oZ1,oZ2,iKEY,iCLOCK,RESET)
    COMPARE u15 (oZ0,oZ1,oZ2,ooZ0,ooZ1,ooZ2,iKEY,iCLOCK,RESET)
    //bulunan medyan değeri "Medyan" paremetresi
    //ile çağıran modüle gönderilir
    COMPARE u16 (ooZ0,ooY1,ooX2, min,Medyan,max,iKEY,iCLOCK,RESET)
```

Şekil 5-7 Adaptif dinamik medyan filtreleme algoritması

Elde edilen sonuç Şekil 5-8, Şekil 5-9 ve Şekil 5-10'de gösterilmiştir. Şekil 5-8 orijinal resmi, Şekil 5-9 adaptif dinamik medyan filtreleme algoritması sonucunu, Şekil 5-10 ise aynı resim için statik medyan filtreleme algoritmasını göstermektedir. Dikkat edilirse Şekil 5-10'da gürültü azaltılmıştır ancak resim detayları da kaybolmuştur. Sol kenara yakın dal üzerindeki detayların orijinal resme göre kaybolduğu net bir şekilde görülmektedir. Adaptif dinamik medyan filtreleme uygulanan Şekil 5-9'da ise bu detaylar halen gözlenebilmektedir.

Şekil 5-8 Orijinal resim

Şekil 5-9 Dinamik adaptif medyan filtreleme

Şekil 5-10 Statik medyan filtreleme

5.2 Kenar Tespit Etme

Tez çalışmasında kenar tespit etme algoritması olarak Prewitt operatörü kullanılmıştır.

5.2.1 Prewitt operatörü kullanılarak kenar tespiti

Prewitt operatörü 4.2.1 bölümünde anlatıldığı gibi kenar tespiti için yatay ve dikeyde farklı olmak üzere iki konvolusyon matrisi kullanır. Konvolusyon matrisleri 4.2.1 bölümünde verilmiştir. Matrislerin işlenmesi için bir satırın hafızada tutulması gerekmektedir. Bu işlem ayrı bir modül yardımıyla yapılmıştır. Elde edilecek sonuç renk bilgisinden bağımsız olacağı için konvolusyon matrislerinde girdi olarak parlaklık bilgisi (siyah-beyaz resim) kullanılmıştır. Konvolusyon matrisleri kullanılarak resimdeki yatay ve dikey kenarlar ayrı ayrı belirlenir.

Klasik Prewitt kenar tespit etme yönteminde yatay ve dikey kenar bilgilerinin önce karesi, ardından karekökü alınıp toplanarak resmin kenarları belirlenir. Ancak çalışma sırasında yapılan denemelerde belirlenen yatay ve dikey kenarların önce toplanması, ardından bu değerin iki katının alınmasının klasik Prewitt metoduna göre resimdeki kenarları çok daha belirgin şekilde tespit ettiği görülmüştür. Bunun neticesinde algoritma klasik Prewitt uygulamasından farklılık

göstermektedir. Tüm bu işlemler hareketli resimde ve gerçek zamanlı olarak yapılmaktadır.

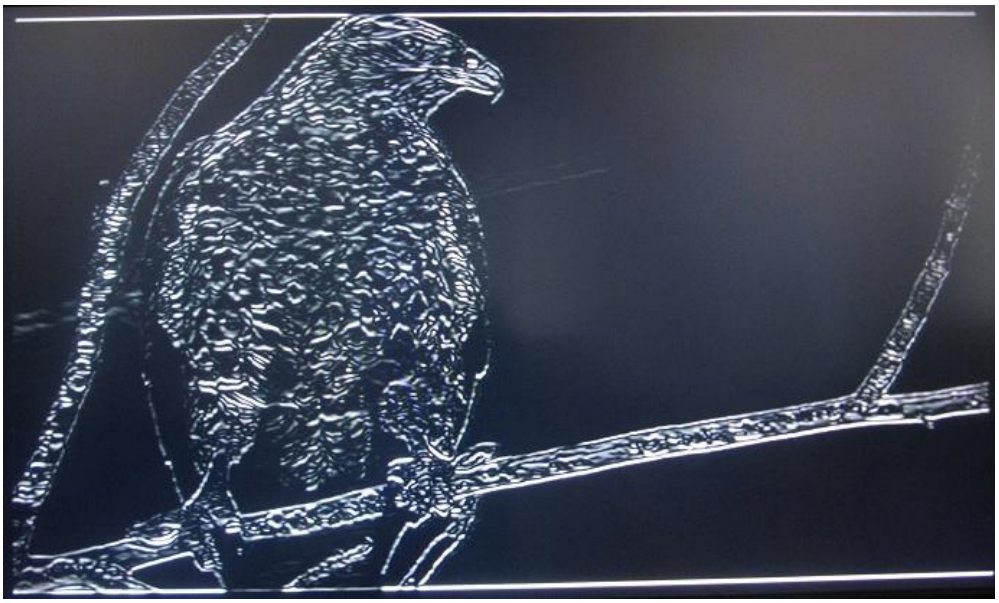
Gerçek zamanlı kenar tespiti algoritması Şekil 5-11'de verilmiştir.

```
prewittEDGE [iR,iG,iB,RESET, iCLOCK, iDVAL, iKEY, oR, oG, oB]
    //1 satır Y bilgisi hafızaya alınır
    LINEBUFFER u0   (iDVAL,iCLOCK,Y,line11,line21,line31)
    //parametre atamaları yapılır
    Y   =   iR + iB + iG;
    oR  =   prewitt_out;
    oG  =   prewitt_out;
    oB  =   prewitt_out;

    if (iCLOCK yükselen kenarda ise)
        if(RESET = 0)
                {H_col1, H_col2, V_col1, V_col2, H_edge, V_edge,
                prewitt_out, H+V_edge;} = 0;
        else
            if (her piksel için iDVAL = 1 ise)
                line32<=line31; line33<=line32; line34<=line33;
                line22<=line21; line23<=line22; line24<=line23;
                line12<=line11; line13<=line12; line14<=line13;

                //Yatay Kenar Hesaplanır
                H_col1 = (line12+line13+line14);
                H_col2 = (line32+line33+line34);
                if (H_col1 > H_col2)
                    H_edge= H_col1 - H_col2;
                else
                    H_edge = H_col2 - H_col1;

                //Dikey Kenar Hesaplanır
                V_col1 = (line14+line24+line34);
                V_col2 = (line12+line22+line32);
                if (V_col1 > V_col2)
                    V_edge=V_col1 - V_col2;
                else
                    V_edge=V_col2 - V_col1;

                //Yatay + Dikey Kenar Hesaplanır
                H+V_edge = H_edge + V_edge;

                if(iSel[0]=1 ise Çıkış video = Yatay Kenar)
                    prewitt_out=H_edge*2;
                else if(iSel[1]=1 ise Çıkış video = Dikey Kenar)
                    prewitt_out=V_edge*2;
                else if(iSel[2]=1 ise Çıkış video = Yatay+Dikey Kenar)
                    prewitt_out=H+V_edge*2;
            else
                {H_col1, H_col2, V_col1, V_col2, H_edge, V_edge,
                prewitt_out, H+V_edge;} = 0;
```

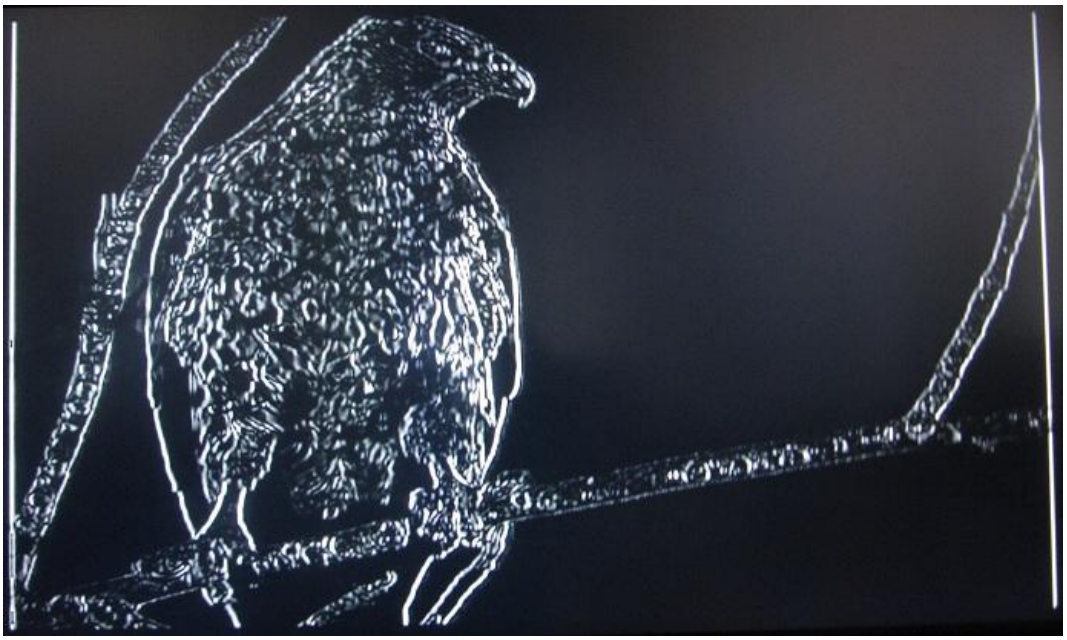
Şekil 5-11 Gerçek zamanlı kenar tespiti algoritması

DE2-70 kartının iSW[0] tuşu yatay kenarların tespit edilmesi algoritmasını kontrol etmektedir. Benzer şekilde iSW[1] tuşu dikey kenarların tespit edilmesi algoritmasını kontrol eder. iSW[2] tuşu ise yatay ve dikey kenarların toplanmış hali olan ana kenar tespiti algoritmasını aktif hale getirmektedir. Şekil 5-12 orijinal resmi göstermektedir. Şekil 5-13 yatay konvolusyon operatörü çıktısını, Şekil 5-14 dikey konvolusyon operatörü çıktısını ve Şekil 5-15 resmin kenar tespit algoritması çıktısını göstermektedir.

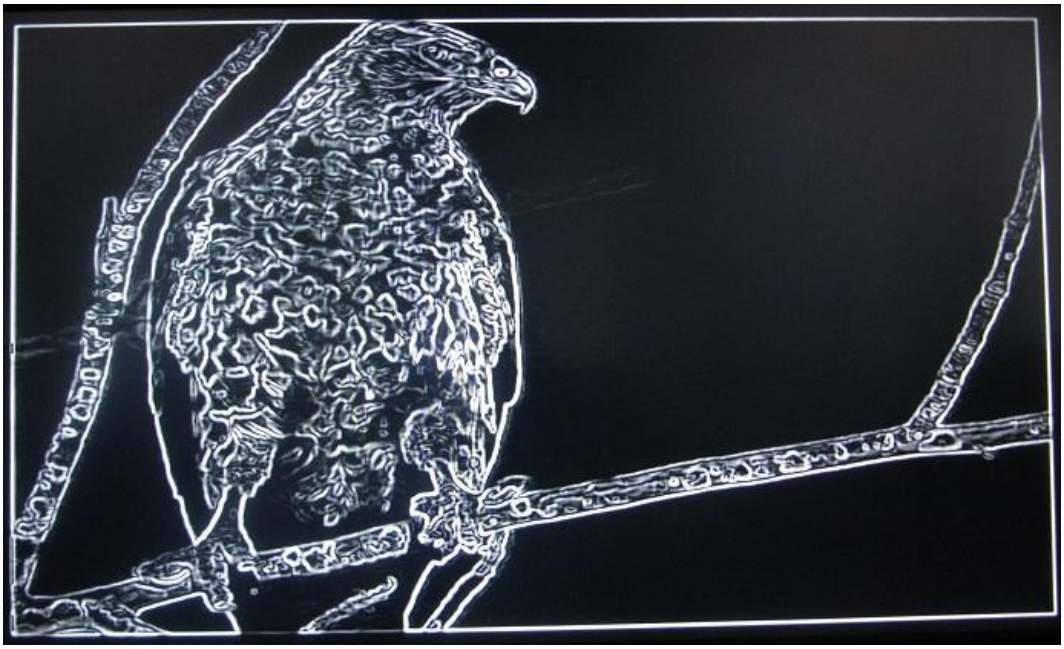
Şekil 5-12 Orijinal resim



Şekil 5-13 Yatay konvolusyon operatörü sonucu



Şekil 5-14 Dikey konvolusyon operatörü sonucu



Şekil 5-15 Kenar tespiti

5.3 Karşıtlık İyileştirme

4.3 bölümünde anlatıldığı gibi karşıtlık (kontrast) iyileştirme için gelen resmin histogramının hesaplanması ve çıkan sonuca göre resmin parlaklık seviyesinin ayarlanması gerekmektedir.

5.3.1 Gerçek zamanlı histogram hesaplama

Histogram modülü diğer tüm modüller gibi bağımsız bir modül olarak yazılmış olup bir üst modül tarafından ilgili parametre atamalarının yapılarak çağrılması suretiyle mimaride kullanılmaktadır. Histogram grafik ara yüzü 7-bit olacak şekilde tasarlanmıştır. Modüle gelen 10-bitlik parlaklık bilgisi (Y) 7-bitlik histogram tablosuna eşlenmiş ve hareketli resmin bir çerçevesine ait tüm parlaklık değerlerinin tutulduğu "histogramCount[127:0]" parametresi elde edilmiştir. Bu değişkenin elde edilmesinin ardından histogram grafiği 640 x 480 çözünürlüğündeki ekranda 258 x 150 boyutunda sütün grafik formatında ekrana bastırılmıştır. Tüm bu işlemler süresince hareketli resimdeki bir çerçevenin tamamlanması, yeni çerçeve için değişkenlerin sıfırlanması ve tampon belleğin temizlenmesi gibi operasyonlar çeşitli koşut ifadeleri yardımıyla gerçekleştirilmiştir. Gerçek zamanlı histogram hesaplama algoritması Şekil 5-16 ve Şekil 5-17'de verilmiştir.

```
1  HISTOGRAM [ iDVAL,iCLOCK,RESET,iKEY,iRed, iGreen,iBlue,iX_Count,
2              iY_Count,iY, oBW, oY, oRed, oGreen, oBlue,oDVAL]
3      //parametre atamaları yapılır
4      iBW     =   iY;
5      oBW     =   iRed + iBlue + iGreen;
6      oY      =   iBW;
7      oRed    =   regRed;
8      oGreen  =   regGreen;
9      oBlue   =   regBlue;
10     oDVAL   =   iDVAL;
11
12     if (iCLOCK yükselen kenarda VEYA RESET düşen kenarında ise)
13     begin
14         if(RESET = 0)
15             //değişkenlerin ilk değerleri atanır
16             {regRed, regGreen, regBlue} = 0;
17             for (i = 127; i >= 0; i = i-1)
18                 count[i] = 0;
19             for (j = 127; j >= 0; j = j-1)
20                 pre_count[j] = 0;
21             for (x = 0; x <= 127; x = x+1)
22                 temp1[127-x] <= 1023 - x*8;
23             for (y = 0; y <= 127; y = y+1)
24                 temp2[127-y] <= 1016 - y*8;
25             for (z = 0; z <= 127; z = z+1)
26                 temp3[127-z] <= 297 - z*2;
27             for (w = 0; w <= 127; w = w+1)
28                 temp4[127-w] <= 296 - w*2;
29         else if (iKEY = 1 ise histogram çizim bloğunu çalıştır)
30         begin
31             if ((iX_Count >= 0) VE (iX_Count = 640) VE
32                 (iY_Count >= 0) VE (iY_Count = 480))
33                 for (k = 127; k>=0; k = k-1)
34                     if ((iBW <= temp1[k])VE(iBW >=temp2[k])VE(count[k] <= 102400))
35                         count[k] <= count[k] + 1;
36             //640x480 boyutlu ekrandaki 10-bitlik veri 7-bitlik veriye indirgenir
```

Şekil 5-16 Gerçek zamanlı histogram hesaplama algoritması-1

```
37              if ((iX_Count = 640) VE (iY_Count = 480))
38                  //eğer çerçeve sonuna gelindiyse veriler aktarılır ve
39                  //değişkenler yeni çerçeve için ilk değerlerine atanır
40                  for (t = 127; t >= 0; t = t-1)
41                      pre_count[t] = count[t];
42                  for (m = 127; m >= 0; m = m-1)
43                      count[m] = 0;
44                  for (x = 0; x <= 127; x = x+1)
45                      temp1[127-x] <= 1023 - x*8;
46                  for (y = 0; y <= 127; y = y+1)
47                      temp2[127-y] <= 1016 - y*8;
48                  for (z = 0; z <= 127; z = z+1)
49                      temp3[127-z] <= 297 - z*2;
50                  for (w = 0; w <= 127; w = w+1)
51                      temp4[127-w] <= 296 - w*2;
52              ////////HISTOGRAM PENCERESİ ÇİZİMİNE BAŞLANIR//////////
53              //pencere koordinatları belirlenir: X[40;298], Y[310;450]
54              if ((iX_Count = 298) VE (iX_Count >= 40)
55                  VE (iY_Count >= 310) VE (iY_Count = 450))
56                  //histogram penceresi çerçevesi çizimine başlanır
57                  if  (iX_Count = 40)
58                      {regRed, regGreen, regBlue} = 0000(hex);
59                  else if (iY_Count > 445)
60                      {regRed, regGreen, regBlue} = 0000(hex);
61                  else if ((iX_Count = 298) VE (iY_Count > 310))
62                      {regRed, regGreen, regBlue} = 0000(hex);
63                  else if ((iX_Count >= 40) VE (iX_Count = 298) VE (iY_Count = 310))
64                      {regRed, regGreen, regBlue} = 0000(hex);
65                  //histogram penceresi çerçevesi çizimi bitirilir
66                  else
67                      //pencere iç kısmı beyaz yapılır
68                      {regRed, regGreen, regBlue} = FF70(hex);
69              //histogram penceresi içine sütun grafik çizimine başlanır
70                  for (n = 127; n>=0; n = n-1)
71                      if ((iX_Cont <= temp3[n]) VE (iX_Cont >= temp4[n])
72                          VE (iY_Cont >= 465 - (pre_count[n])) VE (iY_Cont <= 465))
73                              {regRed, regGreen, regBlue} = 0000(hex);
74              ////////HISTOGRAM PENCERESİ ÇİZİMİ BİTİRİLİR//////////
75              /////////// REFERANS TON ARALIĞI ÇİZİLİR ///////////////
76              else if ((  iX_Cont <= 298) VE (iX_Cont >= 40)
77                          VE (iY_Cont > 450) VE (iY_Cont <= 475))
78                      if ((iX_Cont >= 11'd40) VE (iX_Cont <= 11'd42))
79                          {regRed, regGreen, regBlue} = 0000(hex);
80                          for (u = 0; u<=255; u = u+1)
81                              if ((iX_Cont == u+41))
82                              {regRed, regGreen, regBlue} = u*4;
83              //histogram penceresi dışında kalan alan için giriş çıkışa yönlendirilir
84              else
85                  {regRed, regGreen, regBlue} =   {iRed, iGreen, iBlue};
86          else if (iKEY 0 ise histogram bloğunu çalıştırma)
87                  //giriş videosunu çıkışa yönlendirilir
88                  {regRed, regGreen, regBlue} =   {iRed, iGreen, iBlue};
```

Şekil 5-17 Gerçek zamanlı histogram hesaplama algoritması-2

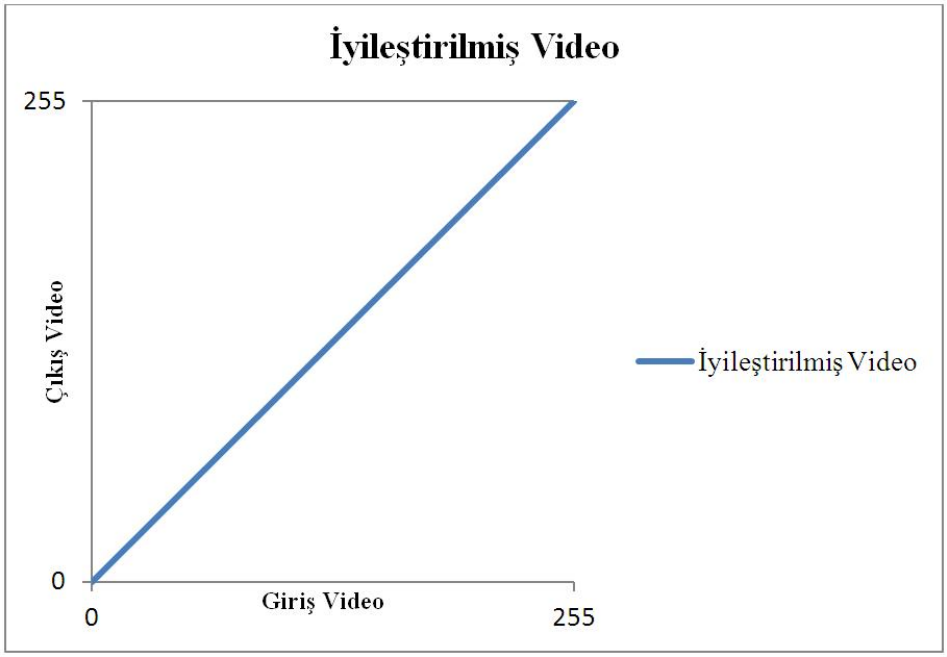
Histogram grafiğinin gösterilmesi için ekran pozisyonu olarak sol-alt tercih edilmiştir. DE2-70 çalışma kartı üzerindeki iSW[0] tuşu histogram modülünü kontrol etmektedir. Bu tuşun ‘1’ yapılması histogram grafiğini ekrana basmakta, ‘0’ yapılması ekrandan kaldırmaktadır. Şekil 5-18 gerçek zamanlı histogram hesaplama algoritmasının ekran çıktısını göstermektedir.

Şekil 5-18 Gerçek zamanlı histogram hesaplama

5.3.2 Renkli Resimde Karşıtlık İyileştirme

Herhangi bir işleme tabi tutulmamış bir resim için karşıtlık eğrisi grafiği Şekil 5-19'de verilmiştir. Grafikte x-ekseni 8-bit giriş piksellerinin değerini, y-ekseni ise 8-bit çıkış piksellerinin değerini göstermektedir. Bu grafikte katsayı olarak '1' kullanıldığı için giriş ve çıkış birbirine eşit olmaktadır.

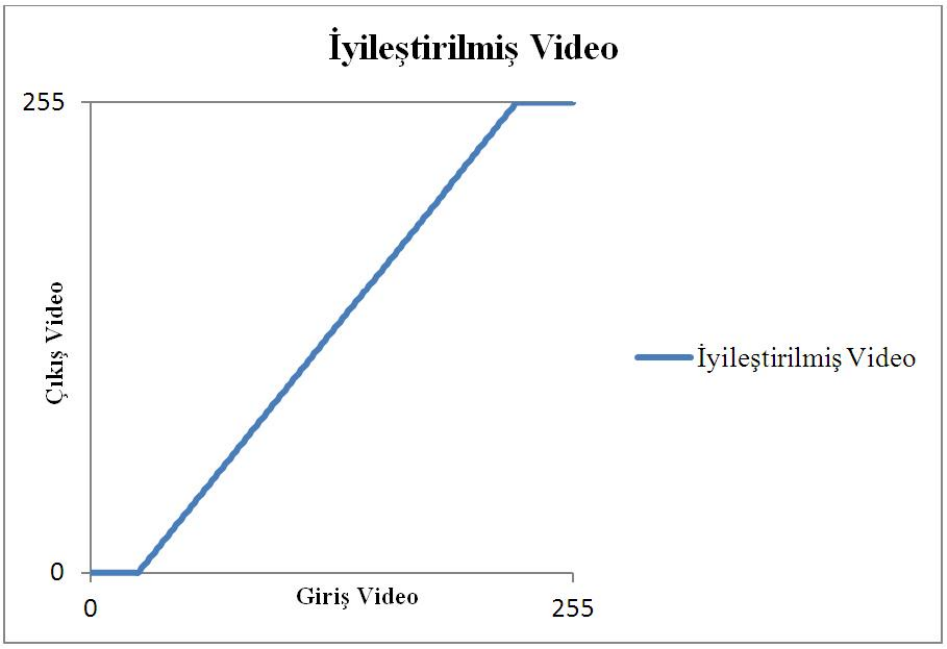
Karşıtlık iyileştirme modülünde iki farklı uygulama gerçekleştirilmiştir. Bunlar: Kontrastın uzatılması (stretching) ve yüksek kontrast uygulamalarıdır. Bu iyileştirmeler resimdeki 8-bitlik parlaklık bilgisinin çeşitli katsayılar ile çarpılarak değiştirilmesi suretiyle yapılmıştır. Katsayılar elde edilirken Matlab programından faydalanılmıştır. Öncelikle istenen 8-bitlik karşıtlık grafiği çizilmiş, ardından bu grafiğe uygun olacak şekilde 256 değer (8-bit) içeren katsayı dizisi Matlab programı yardımıyla oluşturulmuştur. Matlab kullanılarak elde edilen grafiğe ait sayısal değerler tablosu her grafik için ilgili iyileştirme modülüne gömülmüştür.



Şekil 5-19 Doğrusal(lineer) kontrast

5.3.2.1 Karşıtlık arttırılması

Şekil 5-19'deki lineer kontrast eğrisi eğimi artıracak şekilde Şekil 5-20'de gösterildiği gibi yeniden çizilirse resmin kontrastı arttırılmış olacaktır. Bu durum Şekil 5-21'de gösterilmiştir.



Şekil 5-20 Karşıtlığın arttırılması eğrisi

Şekil 5-21 Karşıtlığın arttırılması ve histogram gösterimi

DE2-70 kartındaki iSW[2] tuşu bu görev için atanmıştır. Tuş yardımıyla kontrastın arttırılma işlemi gerçek zamanlı olarak açılıp kapatılabilmektedir. iSW[0] tuşu kullanılarak histogram modülü aktif hale getirilip kontrastın arttırılmasının etkisi grafiksel olarak da görülebilmektedir. Ek olarak iSW[1] tuşu ekranı yatay eksende ikiye bölerek sol tarafta orijinal resmi, sağ tarafta da iyileştirilmiş resmi gösterecek şekilde monitör çıkışını konfigüre etmektedir. Bu tuş yardımıyla da kontrast uzatılması işleminin etkisi orijinal resim ile aynı anda karşılaştırma yapılarak görülebilmektedir. Bu durum Şekil 5-23'de gösterilmiştir. Karşıtlık arttırma algoritması Şekil 5-22'de verilmiştir.

```
STRETCH [iCLOCK, iKEY, iY,iCb,iCr,
        stretchLUT,oY, oCb, oCr, ooY]
    //parametre atamaları yapılır
    oY  = Y;
    oCb = Cb;
    oCr = Cr;
    ooY = originalY;
    if (iCLOCK yükselen kenarda ise)
    begin
        originalY = iY;
        if (iKEY = 1 ile kontrast arttırma aktif edilir ise)
            Cb<=iCb;
            Cr<=iCr;
            for (j=0;j<=255;j=j+1)
                if (iY = j)
                    Y = stretchLUT[j];
            //iY'nin 8-bitlik (256 farklı olası değeri)
            //stretchLUT tablosundaki yeni değerler ile
            //değiştirilerek kontrast arttırılır.
        else if (iKEY = 0 ile kontrast arttırma aktif edilmez ise)
            Y<=iY;
            Cb<=iCb;
            Cr<=iCr;
```

Şekil 5-22 Karşıtlık arttırma algoritması

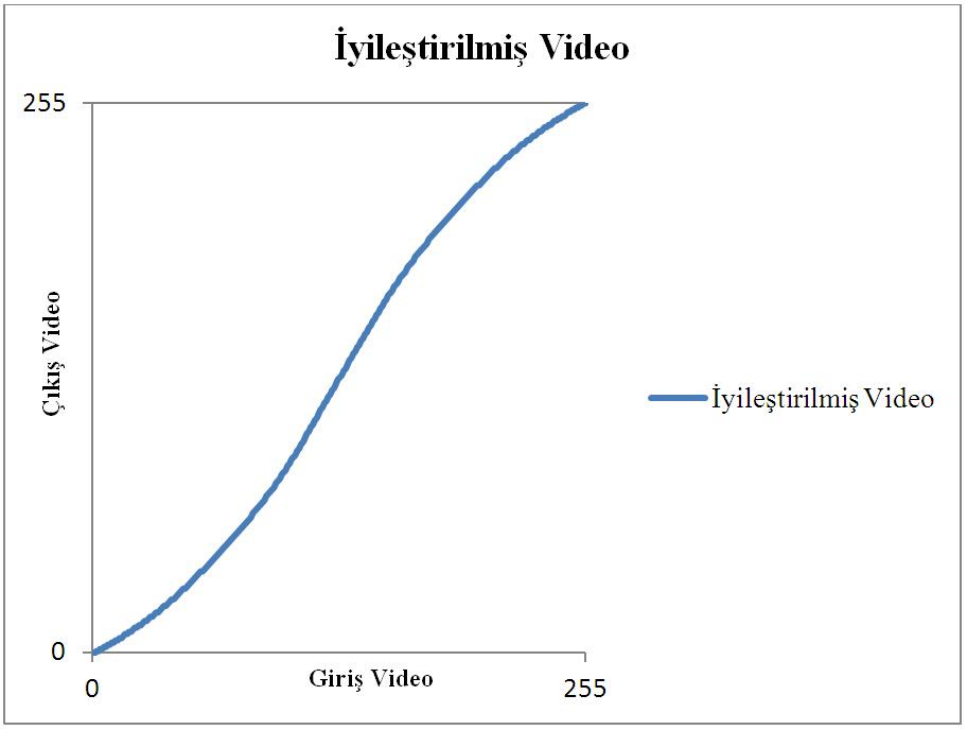
Şekil 5-21'teki histogram grafiği ile orijinal resimdeki (Şekil 5-18) histogram grafiği karşılaştırıldığında algoritmadan geçen resmin kontrastının arttığı gözlenmektedir.

Şekil 5-23 Karşıtlığın arttırılması. Orijinal resim solda, iyileştirilmiş resim sağda

Şekil 5-23 dikkatle incelenirse sağ taraftaki iyileştirilmiş kısımda sol taraftaki orijinal resme göre siyah bölgeler daha siyah, beyaz bölgeler daha beyaz hale getirilmiştir.

5.3.2.2 Yüksek karşıtlık

Şekil 5-19'deki lineer kontrast eğrisi Şekil 5-24'de gösterildiği gibi parlak kısımları daha parlak ve karanlık kısımları daha karanlık yapacak şekilde yeniden çizilirse resim yüksek kontrasta ulaşacaktır. DE2-70 kartındaki iSW[3] tuşu bu görev için atanmıştır. Tuş yardımıyla kontrastın yükseltilmesi işlemi gerçek zamanlı olarak açılıp kapatılabilmektedir. Yine iSW[0] ve iSW[1] tuşları kullanılarak histogram modülü ve/veya ekranı ikiye bölme modülü aktif hale getirilip kontrastın arttırılmasının etkisi ekranda daha kolay fark edilir hale getirilebilmektedir. Bu durumlar Şekil 5-25 ve Şekil 5-26'de gösterilmiştir.



Şekil 5-24 Doğrusal olmayan yüksek karşıtlık eğrisi

Şekil 5-25 Yüksek karşıtlık ve histogram gösterimi.

Şekil 5-25'deki histogram grafiği ile orijinal resimdeki (Şekil 5-18) histogram grafiği karşılaştırıldığında algoritmadan geçen resmin kontrastının arttığı gözlenmektedir.

Şekil 5-26 dikkatle incelenirse sağ taraftaki iyileştirilmiş kısımda sol taraftaki orijinal resme göre siyah bölgeler daha siyah, beyaz bölgeler daha beyaz hale getirilmiştir.

Şekil 5-26 Yüksek karşıtlık. Orijinal resim solda, iyileştirilmiş resim sağda

5.3.2.3 Gerçek zamanlı dinamik adaptif karşıtlık iyileştirme

5.3.2 kısmında anlatılan karşıtlık iyileştirme algoritmaları farklı girdilerde birbirlerine üstünlük sağlayacaktır. Örneğin gelen resimde parlak bileşenler çoğunlukta ise kontrastın uzatılması parlaklığı daha da arttıracağı için resimdeki detayların kaybolmasına yol açar. Bu nedenle gerçek zamanlı dinamik bir şekilde karşıtlık iyileştirme işlemi yapmak için gelen hareketli resmin parlaklık seviyesinin bilinmesi gerekmektedir. Gerçek zamanlı dinamik karşıtlık iyileştirme işlemi iSW[4] tuşu ile kontrol edilmektedir. Gelen resmin parlaklık seviyesine göre algoritma resmin kontrast seviyesini arttırmakta, parlaklığını azaltmakta, parlaklığını, arttırmakta veya resim üzerinde herhangi bir değişiklik yapmadan çıkış modülüne yönlendirmektedir.

Yine iSW[0] ve iSW[1] tuşları kullanılarak histogram modülü ve/veya ekranı ikiye bölme modülü aktif hale getirilip gerçek zamanlı dinamik karşıtlık iyileştirme etkisi ekranda daha kolay fark edilir hale getirilebilmektedir. Bu durumlar Şekil 5-27 ve Şekil 5-28'da gösterilmiştir.

Şekil 5-27 Gerçek zamanlı dinamik karşıtlık iyileştirme ve histogram gösterimi.

Şekil 5-28 Gerçek zamanlı dinamik karşıtlık iyileştirme. Orijinal resim solda, iyileştirilmiş resim sağda.

Yüksek karşıtlık, ve gerçek zamanlı dinamik karşıtlık iyileştirme algoritmalarını içeren kontrast iyileştirme algoritması Şekil 5-29 ve Şekil 5-30'da verilmiştir.

```
colorCONTRAST [iCLOCK, iKEY, iY,iCb,iCr, iX_Count,
               iY_Count, highCountrastLUT, brighterLUT,
               darkerLUT, oY, oCb, oCr, ooY]

    //parlaklık değerinin hesaplanması için
    //resim Y_CAL modülüne gönderilir
    Y_CAL   u7  (.iX_Count(iX_Count), .iY_Count(iY_Count), .iY(iY),
                .oY_Cal(oY_Cal), .iCLOCK(iCLOCK),);

    //parametre atamaları yapılır
    oY  = Y;
    oCb = Cb;
    oCr = Cr;
    ooY = originalY;

    if (iCLOCK yükselen kenarda ise)
        originalY = iY;
        if (iKEY[0] = 1 ile yüksek kontrast aktif edilir ise)
            Cb<=iCb; Cr<=iCr;
            for (j=0;j<=255;j=j+1)
                if (iY = j)
                    Y = highCountrastLUT[j];
        else if (iKEY[1] = 1 ile dinamik kontrast
                 iyileştirme aktif edilir ise)
            Cb<=iCb; Cr<=iCr;
            for (n=0;n<=255;n=n+1)
                if (iY = n)
                    if (oY_Cal<80)
                        Y = brighterLUT[n];
                    else if (oY_Cal>180)
                        Y = darkerLUT[n];
                    else if ((oY_Cal>110) AND (oY_Cal<150))
                        Y = n;
                    else
                        Y = highCountrastLUT[n];
        else (herhangi bir iyileştirme seçilmez ise)
            Y<=iY;
            Cb<=iCb;
            Cr<=iCr;
```

Şekil 5-29 Yüksek karşıtlık ve gerçek zamanlı dinamik karşıtlık iyileştirme algoritmaları ana modülü

```
1   Y_CAL [iCLOCK, iY, iY_Count, iX_Count,oY_Cal]
2
3       //parametre atamaları yapılır
4       oY_Cal  = Cal;
5
6       if (iCLOCK yükselen kenarda ise)
7           if ((iX_Count >= 0)
8               AND (iX_Count <= 640)
9               AND (iY_Count >= 0)
10              AND (iY_Count < 480))
11                  Y = iY + Y;
12
13          if  (iY_Count = 479)
14              Cal = Y;
15
16          if  (iY_Count = 480)
17              Y = 0;
```

Şekil 5-30 Karşıtlık arttırma algoritması Y_Cal alt modülü

5.4 Gerçek Zamanlı Hareket Tespit Etme

Canlı videoda gerçek zamanlı hareket tespit etme tez çalışmasında gerçekleştirilen en kapsamlı uygulamadır. Bu algoritma birden çok görüntü işleme yönteminin birleştirilmesi ile oluşturulmuştur. Ana modül algoritması olarak çerçeve fark yöntemi benimsenmiştir. Bu yöntemin detayları bölüm 5.4.1'de anlatılmıştır. Yardımcı algoritmalar konvolusyon operatörleri ve morfolojik operasyonlardan oluşmaktadır. Morfolojik operatörler olarak daraltma (Bkz. 4.4.1) ve genleştirme (Bkz. 4.4.2) algoritmaları kullanılmıştır. Konvolusyon matrisi kullanılarak tasarlanan diğer algoritma konvolusyon matrisindeki dokuz değeri toplayarak çıkan değerin belirlenen bir eşik değerine göre karşılaştırarak piksel değerlerini "siyah" veya "beyaz" olarak belirlemektedir. Şekil 5-31'de bu yakınsama algoritmasının detayları verilmiştir. Daraltma operasyonu algoritması Şekil 5-32'de, genleştirme operasyonu algoritması Şekil 5-33'de verilmiştir.

```
CONVERGE [iData, iCLOCK, iDVAL, iKEY, oData]

    //1 satır Y bilgisi hafızaya alınır
    LINEBUFFER u0   (iDVAL,iCLK,iData,a0,b0,c0)

    if (iCLOCK yükselen kenarda ise)
        if(her piksel için iDVAL = 1 ise)
            c1<=c0; c2<=c1; c3<=c2;
            b1<=b0; b2<=b1; b3<=b2;
            a1<=a0; a2<=a1; a3<=a2;
            Toplam = (a1+a2+a3+b1+b2+b3+c1+c2+c3);
            if(iKEY = 0 ile algoritma aktif edilmez ise)
                oData = iData;
            else if(iKEY = 1 ile algoritma aktif edilir ise)
                if (Toplam >= 3000)
                    oData = 1023;
                else
                    oData = 0;
```

Şekil 5-31 Yakınsama algoritması.

```
EROSION [iData, iCLOCK, iDVAL, iKEY, oData]

    //1 satır Y bilgisi hafızaya alınır
    LINEBUFFER u0   (iDVAL,iCLOCK,iData,a0,b0,c0)

    if (iCLOCK yükselen kenarda ise)
        if(her piksel için iDVAL = 1 ise)
            c1<=c0; c2<=c1; c3<=c2;
            b1<=b0; b2<=b1; b3<=b2;
            a1<=a0; a2<=a1; a3<=a2;
            Toplam = (a1+a2+a3+b1+b2+b3+c1+c2+c3);
            if(iKEY = 0 ile algoritma aktif edilmez ise)
                oData = iData;
            else if(iKEY = 1 ile algoritma aktif edilir ise)
                //Toplam 8x1023'ten büyük ise tüm komşu piksellerde
                //veri vardır (AND operatörü). Aktif piksel "1" yapılır.
                if (Toplam >= 8200)
                    oData = 1023;
                //Toplam 8x1023'den küçük ise komşu piksellerden en az biri
                //boştur. o nedenle buradaki veri silinir.
                else
                    oData = 0;
```

Şekil 5-32 Daraltma algoritması.

```
DILATION [iData, iCLOCK, iDVAL, iKEY, oData]

    //1 satır Y bilgisi hafızaya alınır
    LINEBUFFER u0   (iDVAL,iCLOCK,iData,a0,b0,c0)

    if (iCLOCK yükselen kenarda ise)
        if(her piksel için iDVAL = 1 ise)
            c1<=c0; c2<=c1; c3<=c2;
            b1<=b0; b2<=b1; b3<=b2;
            a1<=a0; a2<=a1; a3<=a2;
            Toplam = (a1+a2+a3+b1+b2+b3+c1+c2+c3);
            if(iKEY = 0 ile algoritma aktif edilmez ise)
                oData = iData;
            else if(iKEY = 1 ile algoritma aktif edilir ise)
                //Toplam 1023'ten büyük ise en az bir piksellerde
                //veri vardır (OR operatörü). Aktif piksel "1" yapılır.
                if (Toplam >= 1023)
                    oData = 1023;
                //Toplam 1023'den küçük ise komşu piksellerden hiçbirinde
                //veri yoktur. o nedenle buradaki veri silinir.
                else
                    oData = 0;
```

Şekil 5-33 Genleştirme algoritması.

5.4.1 Çerçeve fark yöntemi

Çerçeve fark yönteminin temel prensibi video bilgisindeki ardışık çerçevelerin birbirinden çıkarıldığında elde edilen farkın hareketi vereceği varsayımına dayanır. Tez çalışmasında giriş video işareti olarak NTSC formatında analog video kullanıldığından dolayı bir saniyede altmış çerçeve gelmektedir. İlk çerçeveden itibaren ardışık çerçeveler birbirlerinden çıkarılarak ve elde edilen fark ekrana bastırılarak hareket tespit edilmiş olur. Bu işlemlerin yapılabilmesi için ardışık iki çerçevenin hafızaya alınması gerekmektedir. Hareket tespiti esnasında işlenen veri boyutunu azaltmak, dolayısıyla performansı arttırmak için renk bilgisi kullanılmamıştır. Siyah-beyaz resim üzerinden işlemler gerçekleştirilmiştir.

Hafızaya alınan çerçevelerin birbirinden çıkarılması sonucu elde edilen değer tek bir pikseldeki en küçük farkı dahi içermektedir. Tek bir piksel 10-bitlik veriden (1024 farklı olası değer) oluşmaktadır. Sabit resimde bile kamera alıcısından başlayıp optik verinin analog sinyale dönüştürülmesi, iletilmesi, analog verinin sayısallaştırılması ve video işleme modülüne gelmesine kadar verinin geçtiği tüm ideal olmayan sistemlerden sonra ardışık iki çerçevede 1024 olası değerden aynı değerin gelmesi çoğunlukla mümkün olmamaktadır. Dolayısıyla çerçeve fark işlemi sonrasında elde edilen veri ilk aşamada anlamlı bir veri değildir, hareket tespiti için gereğinden fazla detayı içermektedir.

Şekil 5-34 orijinal videoyu, Şekil 5-35 iki ardışık çerçevenin birbirinden çıkarılması sonucu elde edilen fark bilgisini göstermektedir.

Şekil 5-34 Orjinal video

Şekil 5-35 İki ardışık çerçevenin birbirinden çıkarılması sonucu elde edilen video

DE2-70 kartındaki iSW[0] tuşu çerçeve fark yöntemi algoritmasını aktifleştirmek için kullanılmıştır.

Çerçeve fark yöntemi algoritması Şekil 5-36'de verilmiştir.

```
frameSUBTRACTION iCLOCK, iKEY, iData1, iData2, oData]

    if (iCLOCK yükselen kenarda ise)
            if (iKEY[0] = 0 ile algoritma aktif edilmez ise)
                oData = iData1;
            else if (iKEY[0] = 1 ile algoritma aktif edilir ise)
                if(iData2>iData1)
                    fark = iData2-iData1;
                else if (iData1>iData2)
                    fark = iData1-iData2;
                else
                    fark = 0;

                if (iKEY[1] = 1 ile eşik değeri kontrolü aktif edilir ise)
                    if(fark > (3C(hex)))
                        oData = 1023;
                    else
                        oData = 0;
                else if (iKEY[1] = 0 ile eşik değeri kontrolü aktif edilmez ise)
                    if (fark > 0)
                        oData = 1023;
                    else
                        oData = 0;
```

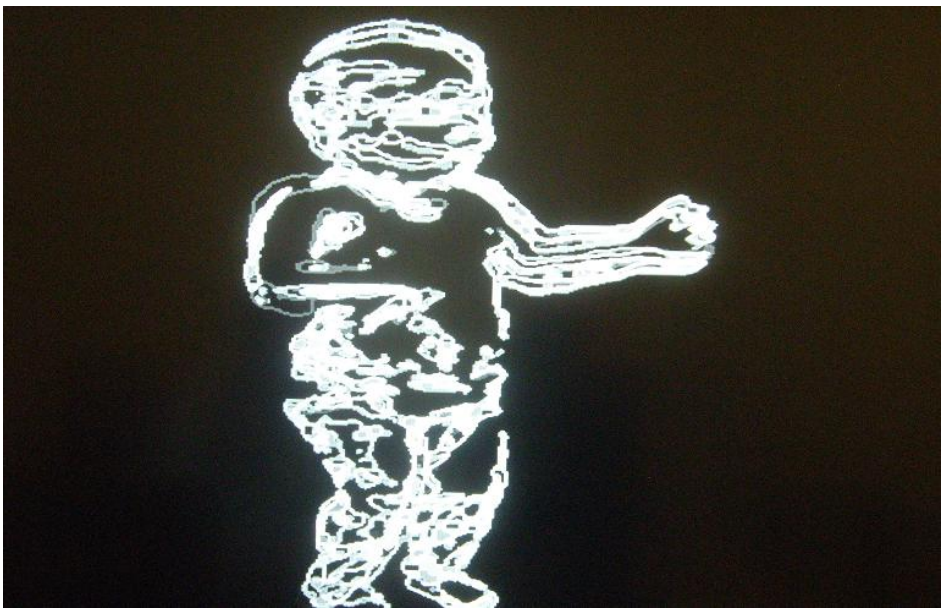
Şekil 5-36 Çerçeve fark yöntemi algoritması

Şekil 5-35'de gösterilen fark bilgisindeki detayın azaltılması adına 5.4 kısmında bahsedilen algoritmalar (yakınsama, genleştirme ve daraltma operasyonları) sırasıyla hareketli resme uygulanmıştır.

DE2-70 kartındaki iSW[4:2] tuşları bu algoritmaları aktifleştirmek için kullanılmıştır. Elde edilen video Şekil 5-37'de gösterilmiştir.

Şekil 5-37 Konvolusyon ve morfolojik operatörlerin sonucu

Şekil 5-37'den de görüldüğü gibi anlamlı veri bu işlemlerin ardından da elde edilememiştir. Son aşama olarak çerçeveler arasındaki fark bilgisinin "3C(hex)" den büyük olması koşutu çerçeve fark algoritmasına eklenmiştir. Bu değer deneme sınama metoduyla bulunmuş normal bir video görüntüsü için optimize edilmiş değerdir. DE2-70 kartındaki iSW[1] tuşu bu özelliği aktifleştirmek için kullanılmıştır. Elde edilen sonuç Şekil 5-38'de gösterilmiştir.



Şekil 5-38 Canlı videoda gerçek zamanlı hareket tespiti

5.4.2 Çerçeve fark yönteminin kaynak video ile bütünleştirilmesi

Hareketin tespit edilmesinin ardından bu bilginin kaynak video üzerinde ön plana çıkarılarak kullanıcıya daha anlamlı bir resim gösterilmesi amaçlanmıştır.

Bu algoritmada öncelikle 640x480 boyutundaki ekran için hareketin tespit edildiği alan (veya alanlar) belirlenmiştir. Bu alan içinde kalan hareketli resim kırmızı olacak şekilde renklendirilerek ve bu alanlar çerçeve içine alınarak hareketin olduğu bölge ön plana çıkarılmıştır. Bunun için geliştirilen algoritma Şekil 5-39, Şekil 5-40 ve Şekil 5-41'de verilmiştir.

```
drawMOTIONframe [iDVAL, iKEY, iCLOCK, RESET,
                 iRed, iGreen, iBlue,
                 iX_Count, iY_Count, oRed,
                 oGreen, oBlue, oDVAL]
    //Hareket tespit edilen siyah beyaz resimde veri
    //olan piksel tespit edilir
    if ((iRed>200)VE(iBlue>200)VE(iGreen>200))
        validData = 1;
    else
        validData = 0;
    //parametre atamaları yapılır
    oRed    = regRed;
    oGreen  = regGreen;
    oBlue   = regBlue;
    oDVAL   = iDVAL;

    if (iCLOCK yükselen kenarda VEYA RESET düşen kenarında ise)
        if(RESET = 0)
        //değişkenlerin ilk değerleri atanır
            regRed      =   0;
            regGreen    =   0;
            regBlue     =   0;
            leftX       =   FFFF;
            rightX      =   0;
            topY        =   FFFF;
            botY        =   0;
            leftX1      =   FFFF;
            rightX1     =   0;
            topY1       =   FFFF;
            botY1       =   0;
            newLeftX    =   FFFF;
            newRightX   =   0;
            newTopY     =   FFFF;
            newBotY     =   0;
            newLeftX1   =   FFFF;
            newRightX1  =   0;
            newTopY1    =   FFFF;
            newBotY1    =   0;
        else
            if(iKEY=0 ile algoritma aktif edilmez ise
               çıkış video = giriş video)
                {regRed, regGreen, regBlue} = {iRed, iGreen, iBlue};
            if(iKEY=1 ile algoritma aktif edilir ise)
                if (validData)
        //her yeni piksel için hareketli resim bilgisi
        //sınırları güncellenir
                    if ((newBotY - iY_Count)<=1 VEYA newBotY==0)
                        regRed      =   FFFF;
                        regGreen    =   0000;
                        regBlue     =   0000;
                        if (iX_Count <= newLeftX)
                            if (iX_Count >= 6)
                                newLeftX = iX_Count - 1;
                        if (iX_Count >= newRightX)
                            if (iX_Count <= 634)
                                newRightX = iX_Count + 1;
```

Şekil 5-39 Tespit edilen hareketin renklendirilmesi ve çerçeve içine alınması algoritması-1

```
56                                  newRightX = iX_Count + 1;
57                              if (iY_Count <= newTopY)
58                                  if (iY_Count >= 6)
59                                      newTopY = iY_Count - 1;
60                              if (iY_Count >= newBotY)
61                                  if (iY_Count <= 474)
62                                      newBotY  =  iY_Count + 1;
63                                      newBotY1 =  1;
64                          else if((newBotY1 - iY_Count)<=1
65                                   VEYA (newBotY1 == 1))
66                              regRed      =   FFFF;
67                              regGreen    =   0000;
68                              regBlue     =   0000;
69                              if (iX_Count <= newLeftX1)
70                                  if (iX_Count >= 6)
71                                      newLeftX1 = iX_Count - 1;
72                              if (iX_Count >= newRightX1)
73                                  if (iX_Count <= 634)
74                                      newRightX1 = iX_Count + 1;
75                              if (iY_Count <= newTopY1)
76                                  if (iY_Count >= 6)
77                                      newTopY1 = iY_Count - 1;
78                              if (iY_Count >= newBotY1)
79                                  if (iY_Count <= 474)
80                                      newBotY1 = iY_Count + 1;
81                              else
82                                  {regRed, regGreen, regBlue}
83                                      = {iRed, iGreen, iBlue};
84                          else
85                              {regRed, regGreen, regBlue}
86                                      = {iRed, iGreen, iBlue};
87
88                  //belirlenen sınır değerlerine göre beyaz renkli
89                  //çerçeve hareketi gösterecek şekilde ekrana çizilir
90                          if ((iY_Count = botY) VE
91                             ((iX_Count >= leftX) VE (iX_Count <= rightX)))
92                                  {regRed, regGreen, regBlue} = FF;
93                          else if ((iY_Count = topY) VE
94                                  ((iX_Count >=leftX) VE (iX_Count <= rightX)))
95                                      {regRed, regGreen, regBlue} = FF;
96                          else if ((iX_Count == leftX) VE
97                                  ((iY_Count >= topY) VE (iY_Count <= botY)))
98                                      {regRed, regGreen, regBlue} = FF;
99                          else if ((iX_Count == rightX) VE
100                                 ((iY_Count >= topY) VE (iY_Count <= botY)))
101                                     {regRed, regGreen, regBlue} = FF;
102                         else if ((iY_Count > topY) VE (iY_Count < botY) VE
103                                  (iX_Count > leftX) VE  (iX_Count < rightX))
104                                     {regRed, regGreen, regBlue}
105                                         = {iRed, iGreen, iBlue};
106
107                         if ((iY_Count = botY1) VE
108                            ((iX_Count >= leftX1) VE (iX_Count <= rightX1)))
109                                 {regRed, regGreen, regBlue} = FF;
110                         else if ((iY_Count == topY1) VE
111                                 ((iX_Count >=leftX1) VE (iX_Count <= rightX1)))
112                                     {regRed, regGreen, regBlue} = FF;
113                         else if ((iX_Count == leftX1) VE
114                                 ((iY_Count >= topY1) VE (iY_Count <= botY1)))
115                                     {regRed, regGreen, regBlue} = FF;
```

Şekil 5-40 Tespit edilen hareketin renklendirilmesi ve çerçeve içine alınması algoritması-2

```
116                 else if ((iX_Count == rightX1) VE
117                     ((iY_Count >= topY1) VE (iY_Count <= botY1)))
118                         {regRed, regGreen, regBlue} = FF;
119                 else if ((iY_Count > topY1) VE (iY_Count < botY1) VE
120                     (iX_Count > leftX1) VE  (iX_Count < rightX1))
121                         {regRed, regGreen, regBlue}
122                             = {iRed, iGreen, iBlue};
123             else (Hareketli resim kırmızı yapılır)
124                 regRed   =  FF;
125                 regGreen =  0;
126                 regBlue  =  0;
127         //yeni frame için hareket sınırı değişkenleri sıfırlanır
128             if ((iX_Count = 639) VE (iY_Count = 479))
129                 leftX       =   newLeftX;
130                 rightX      =   newRightX;
131                 topY        =   newTopY;
132                 botY        =   newBotY;
133                 leftX1      =   newLeftX1;
134                 rightX1     =   newRightX1;
135                 topY1       =   newTopY1;
136                 botY1       =   newBotY1;
137                 newLeftX    =   FFFF;
138                 newRightX   =   0;
139                 newTopY     =   FFFF;
140                 newBotY     =   0;
141                 newLeftX1   =   FFFF;
142                 newRightX1  =   0;
143                 newTopY1    =   FFFF;
144                 newBotY1    =   0;
```

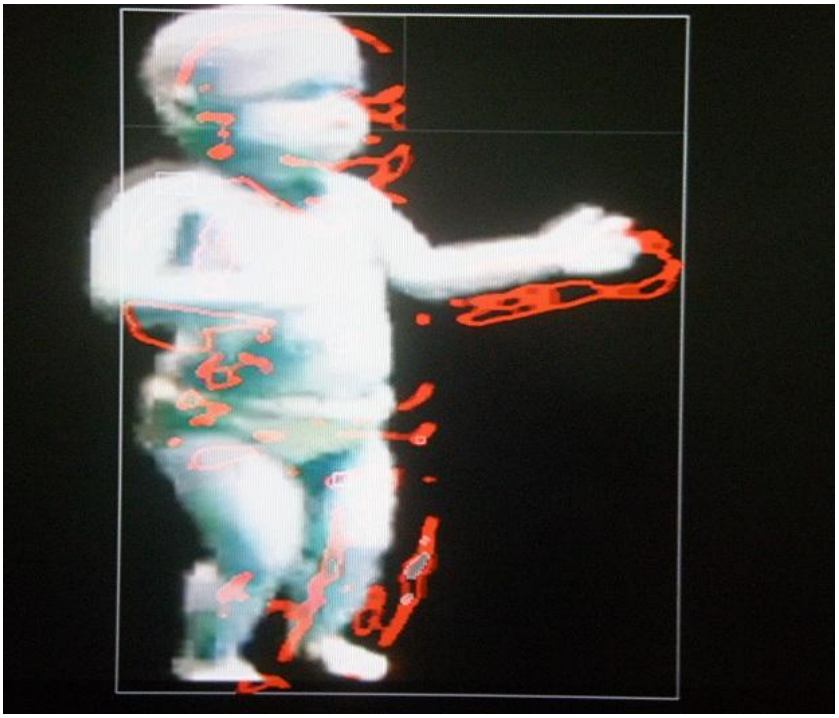
Şekil 5-41 Tespit edilen hareketin renklendirilmesi ve çerçeve içine alınması algoritması-3

DE2-70 kartındaki iSW[5] tuşu bu algoritmayı aktifleştirmek için kullanılmıştır. Algoritma çıktısını ekranda göstermek için DE2-70 kartındaki iSW[7] tuşu kullanılmıştır. Bu tuş ‘1’ yapılarak elde edilen sonuç ekrana yönlendirilmektedir.

Şekil 5-42 bu algoritma çıktısını göstermektedir.

Şekil 5-42 Tespit edilen hareketin renklendirilerek çerçeve içine alınması

Bu işlemden sonra elde edilen veri bit bazında (24bit, 640x480 piksel için) kaynak video ile mantıksal VEYA operatörüne sokularak iki resim üst üste bindirilmiş ve hedeflenen sonuca kısmen ulaşılmıştır. Elde edilen nihai sonuç Şekil 5-43'da verilmiştir.



Şekil 5-43 Kaynak video üzerinde tespit edilen hareketin gösterilmesi

Elde edilen videoda iki resmin üst üste gösterilmesi sırasında zaman düzleminde bir gecikme gözlenmektedir. Bunun nedeni hareket tespit edilirken yapılan hesaplamalardan dolayı geçen zaman süresince kaynak videonun gelmeye devam etmesidir. Gelen kaynak video herhangi bir birimde depolanıp işlemler tamamlandığında hareket tespit edilen video görüntüsü ile aynı anda ekrana basılamadığından dolayı bu durum oluşmaktadır. Mevcut donanım mimarisinde kaynak videoyu depolayacak yeterli hafıza alanı bulunmamaktadır.

DE2-70 kartındaki iSW[6] tuşu iki resmin üst üste gösterilmesi algoritmasını aktifleştirmektedir.

Bu algoritma iki farklı videoyu girdi olarak almakta, iSW[7] ve iSW[6] tuşlarının konumlarına göre bu videoları veya bu iki video toplamını ekrana basmaktadır.

Girdi olarak alınan ilk video orijinal videodur. İkinci video iSW[5] tuşu konumuna göre sadece hareket tespiti veya renklendirilmiş ve çerçeve içine alınmış hareket tespiti olabilmektedir. Algoritma Şekil 5-44'de verilmiştir.

```
1   COMBINE [iR, iG, iB, imR, imG, imB,RESET, iCLOCK, iDVAL,iKEY,oR, oG, oB]
2
3       //parametre atamaları yapılır
4       oR  =   ooR;
5       oG  =   ooG;
6       oB  =   ooB;
7
8       if (iCLOCK yükselen kenarda ise)
9       begin
10          if(RESET = 0)
11                  ooR = 0; ooG = 0; ooB = 0;
12          else
13              if(iher piksel için iDVAL = 1 ise)
14                  if( tuş seçimi iKEY = "00" ise
15                      Çıkış video = girişVideo1)
16                          ooR = iR;
17                          ooG = iG;
18                          ooB = iB;
19                  else if(tuş seçimi iKEY = "01" ise
20                          Çıkış video = girişVideo1 + girişVideo2)
21                          ooR = iR OR imR;
22                          ooG = iG OR imG;
23                          ooB = iB OR imB;
24                  else if(tuş seçimi iKEY = "10" ise
25                          Çıkış video = girişVideo2)
26                          ooR = imR;
27                          ooG = imG;
28                          ooB = imB;
```

Şekil 5-44 Kaynak video ve hareket tespitinin çerçeveye alınması videolarının birleştirilmesi algoritması.

6. SONUÇ VE TARTIŞMA

Tez çalışmasında gömülü bir sistemde sayısal görüntü işleme ve iyileştirme algoritmalarının geliştirilmesi gerçek zamanlı olarak hareketli resim üzerinde uygulanması hedeflenmiştir. Yapılan çalışmalar sonucunda bu hedefe büyük ölçüde ulaşılmıştır.

Görüntü iyileştirme algoritmaları olarak gürültü azaltma ve karşıtlık iyileştirme konuları ele alınmıştır. Görüntü işleme alanında ise gerçek zamanlı video bilgisi üzerinde kenar tespiti ve yine gerçek zamanlı video bilgisi üzerinde hareket tespiti konularına çalışılmıştır. Bu konularda geliştirilen algoritmalar gömülü sistem donanımı olarak seçilen "Terasic DE2-70" çalışma ve geliştirme kartı üzerinde "Verilog" dili kullanılarak gerçeklenmiştir.

Görüntü iyileştirme algoritması olarak ilk gerçeklenen konu gürültü azaltma algoritmalarının geliştirilmesidir. Statik medyan filtreleme algoritması gerçeklenmiş, gürültülü resimde pürüzsüzleştirme işlemi uygulanmıştır. Tespit edilen keskinliğin kaybolması problemine çözüm olarak adaptif dinamik medyan filtreleme algoritması geliştirilmiştir. Bu çözümde gelen kaynak resim incelenerek gürültü tespit edilmekte ve gürültünün miktarına bağlı olarak medyan filtre uygulanmakta veya kaynak resim değiştirilmeden çıkışa yönlendirilmektedir. İlgili algoritmalar Şekil 5-1, Şekil 5-2, Şekil 5-6 ve Şekil 5-7'de, sonuçlar 5.1 bölümünde verilmiştir. Şekil 5-5, Şekil 5-9 ve Şekil 5-10 bu iki yöntemin etkilerini ortaya koymaktadır.

Kimi durumlarda, resimde gürültü miktarı çok fazla ise tek bir medyan filtre taraması yeterli gelmeyebilir. Bu durumlarda medyan filtreleme algoritmasının kaynak resim üzerinde birden fazla iterasyon ile uygulanması gürültünün azaltılmasında etkili olacaktır. İleriki çalışmalarda bu durum irdelenecektir.

Başarıyla çalıştırılan diğer görüntü iyileştirme uygulaması karşıtlık iyileştirmedir. Bu algoritmada çeşitli farklı durumlar için doğrusal veya doğrusal olmayan karşıtlık iyileştirme teknikleri gerçeklenmiştir. Bu işlemler için öncelikli olarak gelen kaynak resmin gerçek zamanlı histogramı bulunmuş ve sütun grafik şeklinde ekrana bastırılmıştır. Histogram bulunması algoritması Şekil 5-16 ve Şekil 5-17'de verilmiştir. Doğrusal karşıtlık iyileştirme yöntemi olarak karşıtlık arttırılması (Şekil 5-22) algoritması yazılmıştır. Sonuçlar Şekil 5-21 ve Şekil 5-23'de verilmiştir. Doğrusal olmayan iyileştirme olarak ise yüksek karşıtlık

algoritması (Şekil 5-29 ve Şekil 5-30) gerçeklenmiştir. Burada Şekil 5-24'da verilen eğriye uygun olarak piksel değerleri daha parlak veya daha karanlık olacak şekilde limite yaklaştırılmıştır. Elde edilen sonuçlar Şekil 5-25 ve Şekil 5-26'da verilmiştir.

Bahsedilen iyileştirme algoritmalarının hangisinin uygulanmasının resim üzerinde olumlu etki yaratacağı gelen kaynak resmin içeriği ile doğrudan alakalı olduğu görülmüştür. Gelen kaynak resme dayalı karşıtlık iyileştirme algoritması olarak adaptif dinamik karşıtlık iyileştirme tekniği geliştirilmiştir. Bu algoritmada kaynak resmin parlaklık değeri hesaplanmıştır. Belirlenen dört eşik değeri ile karşılaştırılan histogram verisi resmin çok karanlık, karanlık, çok aydınlık, aydınlık veya normal sınırlarda olma durumunu ortaya koymuş, buna istinaden resme gerekli karşıtlık iyileştirme algoritması uygulanmıştır. Geliştirilen bu algoritma Şekil 5-29 ve Şekil 5-30'da, sonuç Şekil 5-27'de verilmiştir.

Gerçeklenen bu algoritmalar sayısal görüntü iyileştirme gerektiren gerçek zamanlı gömülü sistemlerde (askeri, medikal, sivil güvenlik vb.) kullanılabilir.

Görüntü iyileştirme algoritmalarının yanında görüntü işleme konusunda da algoritmalar geliştirilmiş ve gerçeklenmiştir. Bunlar gerçek zamanlı kenar tespiti ve gerçek zamanlı hareket tespiti uygulamalarıdır.

Kenar tespiti için "Prewitt Operatörü" kullanılmıştır. Hareketli resimde gerçek zamanlı kenar tespiti algoritması Şekil 5-11'de verilmiştir. Bu algoritmada resim üzerinde 3x3 piksel boyutunda yatay ve dikey kenarlar için ayrı olmak suretiyle iki farklı matris gezdirilerek yatay ve dikey kenarlar belirlenmekte, bu çıktılar toplanarak resimdeki kenarlar ortaya çıkarılmaktadır. Sonuçlar Şekil 5-13, Şekil 5-14 ve Şekil 5-15'de verilmiştir. Bu algoritma daha da geliştirilerek karakter (plaka) tanıma, nesne tanıma gibi uygulamalarda kullanılabilir. Bunun için bir veri tabanı tanımlanarak ve kenar tespit etme algoritması bu veri tabanı ile ilişkilendirilerek bahsedilen uygulamalar gerçeklenebilir.

Gerçek zamanlı hareket tespit uygulamasında iki farklı modül geliştirilmiştir. İlk uygulamada hareket tespit edilerek kullanıcıya sadece gerçek zamanlı videodaki hareket bilgisi gösterilmiştir. Sonuç Şekil 5-38'de gösterilmiştir. Geliştirilen bu algoritma iki ardışık çerçevenin birbirinden çıkarılarak elde edilen farkın işlenmesi sonucunda hareketin algılanmasına dayanmaktadır. Fark bilgisi işlenirken yakınsama algoritması (Şekil 5-31),

daraltma tekniği (Şekil 5-32) ve genleştirme yöntemi (Şekil 5-33) kullanılmıştır. Bu işlemlerden geçirilen fark bilgisi deneme sınama metoduyla belirlenen "3C(hex)"eşik değeri ile karşılaştırılarak hareketin tespit edilmesi gerçeklenmiştir. Çerçeve fark yöntemi ile hareket tespiti algoritması Şekil 5-36'da verilmiştir.

Gerçeklenen ikinci uygulama tez çalışmasındaki en kapsamlı algoritmadır. Bu algoritmada hareket tespit edilerek canlı resim üzerinde ayrı bir renk ile ve bir çerçeve içine alınarak belirginleştirmek suretiyle kullanıcıya gösterilmiştir. Bu algoritmada bir önceki paragrafta anlatılan hareket tespit edilmesi algoritması ayrı bir modüle sokularak burada ekranda hareket olan kısım renklendirilmiş ve hareket olan kısmın piksel koordinatları belirlenerek bu sınırlar arası bir dikdörtgen ekrana çizdirilmiştir. Ardından bu video ve orijinal video birleştirilerek canlı videoda hareket olan kısmın çerçeve içinde gösterimi gerçeklenmiştir. Bu algoritma çıktısı Şekil 5-43'de gösterilmiştir. Algoritma detayları Şekil 5-39, Şekil 5-40, Şekil 5-41 ve Şekil 5-44'de verilmiştir.

Tasarlanan bu algoritmanın iki handikabı vardır. Bunlardan ilki çerçeve çizdirilmesi ile ilgilidir. Çerçeve çizilirken sadece satır bilgisi kullanılmıştır. Bu nedenle aynı satırda iki farklı nesne hareket halindeyse algoritma bu satırlardan büyük indeks numarasına sahip olanını çerçeve sınırı olarak atamaktadır. Dolayısı ile iki farklı hareket eden obje birbirinden ayrı oldukları halde aynı satırda kesiştikleri için aynı büyük çerçeve içinde gösterilmektedir. Farklı satırlarda hareket olması durumunda algoritma istenildiği gibi farklı çerçeveler çizmektedir. Bu durum ileriki çalışmalarda sütun bilgisinin de algoritmaya eklenmesi ile çözülebilir. Ancak bu durumda tüm çerçevenin hafızada tutulması gerekecektir. Bu da yüksek hafıza içeren bir donanım ihtiyacını doğurmaktadır.

İkinci problem algoritmanın ekrana orijinal video ile birlikte hareketi çizmesi esnasında zaman düzleminde gecikmeyle çalışmasıdır. Bunun nedeni DE2-70 kartındaki hafıza donanımının yetersizliğidir. DE2-70 kartında 2 adet SDRAM bulunmaktadır. Bu hafıza birimlerinden birincisi analog videonun sayısallaştırılması, de-interlacing işleminin uygulanması gibi modüllerce kullanılmaktadır. İkinci hafıza birimi hareket tespiti uygulamasında ardışık çerçevelerin saklanması ve işlemlerin yapılması için kullanılmaktadır. Bu işlemlerin yapılması esnasında orijinal video gelmeye devam etmektedir. Gelen orijinal video herhangi bir birimde depolanıp işlemler tamamlandığında hareket tespit edilen video görüntüsü ile aynı anda ekrana basılamadığından dolayı bu durum oluşmaktadır. Mevcut donanım mimarisinde kaynak videoyu depolayacak

yeterli hafıza alanı bulunmamaktadır. İleriki çalışmalarda daha yüksek hafızaya sahip bir donanım ile algoritmadaki bu problem giderilebilir.

Geliştirilen hareket tespiti algoritması daha da iyileştirmeye açıktır. Hareket eden nesne tespit edilerek ve çerçevesi belirlenerek hedef takibi ve hedefe kilitlenme gibi uygulamalar üzerinde çalışılabilir.

Hareket tespiti algoritmaları güvenlik uygulamalarında etkin bir şekilde kullanılabilir. Hareket tespitinin ardından başka sistemleri deveye alan veya uyaran uygulamalar geliştirilebilir.

Yapılan çalışmalarda görülmüştür ki FPGA kullanımı gerçek zamanlı görüntü işleme uygulamaları açısından (analog giriş ve 640x480@60Hz çıkış veri boyutu için) gömülü sistemlerde performans problemi yaratmamaktadır. İşlemler yeterli hızda yapılabilmektedir.

Çalışmaların, özellikle gerçek zamanlı hareket tespiti uygulamalarının, geliştirilebilmeleri için daha yüksek hafıza içeren bir donanıma ihtiyaç duyulmaktadır.

Tez çalışması süresince yapılan çalışmalar neticesinde verilog, mikroişlemciler, RAM operasyonları, gömülü sistemler, gerçek zamanlı sistemler, görüntü iyileştirme ve işleme konularında kişisel kazanım sağlanmıştır.

KAYNAKLAR DİZİNİ

Altera, 2007, "Video and image processing design using FPGAs", http://www.altera.com/literature/wp/wp-video0306.pdf (Erişim Tarihi: 21 Mart 2012).

Angelopoulou, M. E., Bouganis, C., Cheung, P. and Constantinides, G. A., 2009, "Robust Real-Time Super-Resolution on FPGA and an Application to Video Enhancement", ACM Transactions on Reconfigurable Technology and Systems Journal Volume (2), Issue (4), Article (22), 27p.

Bezen, Ö., 1999, Implementation of image processing functions using FPGAs by VHDL methedology, MSc Thesis, Middle East Technical University, 112p.

Bovik, A., 2000, Handbook of image and video processing. Academic Press, New York, 974p.

Cho, J., Mirzaei, S., Oberg, J., and Kastner, R., 2009, "FPGA based face detection system using haar classifiers", http://cseweb.ucsd.edu/~kastner/papers/fpga09-face_detection.pdf (Erişim Tarihi:26 Mart 2012).

Demiryürekli, O., 2009, "Vestel video grubu video iyileştirme araçları", VESTEL, 18s, (yayımlanmamış).

Dorf, R. C., 2000, The Electrical Engineering Handbook, CRC Press LLC, Boca Raton, 3011p.

Doulos, 1995, VHDL Golden Reference Guide, 136p.

Draper, B.A., Beveridge, J.R., Böhm, A.P.W, Ross, J. and Chawathe, M., 2003, Accelerated Image Processing on FPGAs, IEEE Transactions on Image Processing, 12(12)-1543-1551.

Gacar, A., 2009, FPGA Tabanlı Görüntü İşleme Arabirimi, Yüksek Lisans Tezi, Ege Üniversitesi, 76s, (yayımlanmamış).

Gonzalez R. C. and Woods R. E., 2002, Digital Image Processing, Prentice Hall, New Jersey, 797p.

KAYNAKLAR DİZİNİ (devam)

Gunay, H., 2010, Efficient FPGA Implementation of Image Enhancement using Video Streams, M.Sc. Thesis, Middle East Technical University, 73p (unpublished).

Hong, S.N. and Asher, H., 2005, Adaptive Edge Detection for Real-Time Video Processing using FPGAs, International Signal Processing Conference (ISPC), Santa-Clara, 7p.

Intersil, 2002, "Application note 9728.2, BT.656 Video interface for ICs", http://www.intersil.com/data/an/an9728.pdf (Erişim Tarihi:26 Mart 2012)

Karabıyık, A., 2005 Dalgacık sinir ağının alan programlanabilir kapı dizisi ile donanımsal geçekleştirilmesi, Y.Lisans tezi, Ege Üniversitesi, 111s.

Karthigaikumar, P. and Baskaran, K., 2011, "FPGA Implementation of High Speed Low Area DWT Based Invisible Image Watermarking Algorithm", Procedia Engineering Journal Volume (30), 266–273.

Keith, J., 2001, Video demystified, a handbook for the digital engineer, LLH Publishing, Eagle Rock, 783p.

Kelmelis E.J., Ortiz, F. And Curt, P., 2010, "Comparing FPGAs and GPUs for High-performance Image Processing Applications", Visual Information Processing XIX, Proc. SPIE 7701, 77010C, 9p.

Kokufata, K. and Tsutomu, M., 2009, "Real-time Processing of Local Contrast Enhancement on FPGA", International Conference on Field Programmable Logic and Applications Journal, 288-293.

Kumar, K., Jain, A. And Srivastava, A.K., 2009, "FPGA Implementation of Image Enhancement Techniques", Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2009, Proc. of SPIE Vol. 7502 750208-8, 8p.

KAYNAKLAR DİZİNİ (devam)

Matthews, J., 2002, "An Introduction to Edge Detection: The Sobel Edge Detector", www.generation5.org/content/2002/im01.asp (Erişim Tarihi:26 Mart 2012).

Maini, R. and Aggarwal, H., 2010, "Study and Comparison of Various Image Edge Detection Techniques", International Journal of Image Processing (IJIP), Volume (3), Issue (1), 12p.

Maxim, 2001, "AN734: Video basics", http://www.maxim-ic.com/an734 (Erişim Tarihi:22 Mart 2012).

National Instruments, 2012, NI-Tutorial-6984: Introduction to FPGA technology: top five benefits, 2p.

Nelson, A. E., 2000, Implementation of image processing algorithms on FPGA hardware, M.Sc. Thesis, Vanderbilt University, 86p.

Oskoei, M.A. and Hu, H., 2010, "CES-506: A Survey on Edge Detection Methods" www.essex.ac.uk/csee/research/publications/technicalreports/2010/CES-506.pdf (Erişim Tarihi:26 Mart 2012).

Özkan, İ., 2010, Traffic Sign Detection Using FPGA, MSc Thesis, Middle East Technical University, 91p (Unpublished).

Terasic, 2009, "DE2-70 User manual", http://www.terasic.com (Erişim Tarihi:21 Mart 2012).

Wang, Y., 2004, "Video Basics", http://eeweb.poly.edu/~yao/EL6123/Introduction.pdf (Erişim Tarihi:26 Mart 2012).

Wang, Y., 2006, "Image Filtering", http://eeweb.poly.edu/~yao/EE3414/image_filtering.pdf (Erişim Tarihi:26 Mart 2012).

Wikipedia, "Verilog", http://tr.wikipedia.org/wiki/Verilog (Erişim Tarihi: 21 Mart 2012).

KAYNAKLAR DİZİNİ (devam)

Zhengyang, G., Wenbo, X. and Zhilei, C., 2010, "Image Edge Detection Based on FPGA", Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science, DOI 10.1109/DCABES.2010.39, 169-171.

ÖZGEÇMİŞ

Ad Soyad	Ozan TEKDUR
Doğum Tarihi	20.06.1980
Doğum Yeri	Çanakkale
Uyruğu	Türkiye Cumhuriyeti
Eğitim	
Yüksek Lisans	2009-... Ege Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı
Lisans	1998-2002 Dokuz Eylül Üniversitesi Mühendislik Fakültesi Elektrik Elektronik Mühendisliği Bölümü
Çalışma Hayatı	
	Aselsan A.Ş. (Ankara 2010 - …) Sayısal Tasarım Mühendisi
	Vestel Elektronik A.Ş. (Manisa / 2004-2010) FPD TV Sistem Tasarım Mühendisi
	Ajan Elektronik Ltd.Şti (İzmir / 2003-2004) Elektronik Tasarım Mühendisi

Yayın Listesi

1. Ozan TEKDUR, Silvio BRAGIOTTI, Metin NİL, "Yaşlı ve Bakıma Muhtaç İnsanlar İçin Evde Bakım Sistemleri ve IPTV", Haberleşme Teknolojileri ve Uygulamaları Sempozyumu, 2007.

EKLER

Ek1: Türkçe İngilizce Terimler Sözlüğü

Ek1: Türkçe İngilizce Terimler Sözlüğü

Aktif Videonun Sonu	End of Active Video
Aktif Videonun Başlangıcı	Start of Active Video
Binişimsizleştirme	De-Interlacing
Binişimleştirme	Interlacing
Daraltma	Erosion
Değişebilen Faz Hattı	Phase Alternating Line
Dikey	Vertical
Düşük Cetvel	Low Key
Genleştirme	Dilation
Gömülü Sistem	Embedded System
Kaba Kod	Pseudocode
Karartılmış resim	Underexposed Picture
Karşıtlık	Contrast
Kenar tespiti	Edge Detection
Kırmızı	Red
Konfigüre etmek	Configure
Konvolusyon	Convolution
Mavi	Blue

Örnekleme	Sampling
Parlaklık	Luminance
Parlatılmış Resim	Overexposed Picture
Sahada Programlanabilir Kapı Dizileri	Field Programmable Gate Array
Sayısal Sinyal İşlemcisi	Digital Signal Processor
Sayısallaştırma	Quantization
Standart Çözünürlük	Standard Definition
Ton Aralığı	Tonal Range
Ulusal Televizyon Sistemleri Komitesi	National Television System Committee
Uygulamaya Özel Entegre Devre	Application Spesific Integrated Circuit
Uzatma	Stretch
Ve	AND
Veya	OR
Yatay	Horizontal
Yeşil	Green
Yüksek Cetvel	High Key
Yüksek Çözünürlük	High Definition