GÖMÜLÜ GRAFİK İŞLEMCİLERİ İÇİN OPENCL TABANLI GÖRÜNTÜ İŞLEME KÜTÜPHANESİ VE İNSAN YÜZÜ TESPİT ETME UYGULAMASI

OSMAN SEÇKİN ŞİMŞEK

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

TEMMUZ 2014

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Osman Eroğul

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU

Anabilim Dalı Başkanı

Osman Seçkin Şimşek tarafından hazırlanan GÖMÜLÜ GRAFİK İŞLEMCİLERİ İÇİN OPENCL TABANLI GÖRÜNTÜ İŞLEME KÜTÜPHANESİ VE İNSAN YÜZÜ TESPİT ETME UYGULAMASI adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Doç. Dr. Oğuz ERGİN

Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Doç. Dr. Ali Bozbey _____

Üye : Doç. Dr. Oğuz Ergin _____

Üye : Yrd. Doç. Dr. Mehmet Tan _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

……..……..

Osman Seçkin Şimşek

Üniversitesi : TOBB Ekonomi ve Teknoloji Üniversitesi

Enstitüsü : Fen Bilimleri

Anabilim Dalı : Bilgisayar Mühendisliği

Tez Danışmanı : Doç. Dr. Oğuz ERGİN

Tez Türü ve Tarihi : Yüksek Lisans – Ağustos 2014

Osman Seçkin ŞİMŞEK

GÖMÜLÜ GRAFİK İŞLEMCİLERİ İÇİN OPENCL TABANLI GÖRÜNTÜ İŞLEME KÜTÜPHANESİ VE ÖRNEK BİR İNSAN YÜZÜ TESPİT ETME UYGULAMASI

ÖZET

Grafik işlemciler genel amaçlı işlemler için kullanılmaya başladığından beri, kullanım yaygınlığı artmıştır. Yalnızca 3 boyutlu grafik oluşturmak, hareket ettirmek ve simüle etmek için kullanılan grafik işlemciler artık veri paralelliğinin olduğu her alanda kullanılabilmektedir. İçerdiği yüzlerce çekirdek sayesinde, aynı anda normal bir işlemcinin ulaşamadığı kadar iş parçacığını çalıştırabilecek kapasiteye sahip olmuştur. Veri paralel uygulama alanları arasında en başta görüntü işleme uygulamaları, matris ve vektör içerikli lineer cebir uygulamaları, akışkanlar mekaniği, biyoinformatik gibi alanlar gelmektedir. Bu alanlardaki uygulamaların hızlandırılması için grafik işlemcilerin kullanımı halihazırda söz konusudur.

Gelişen ve küçülen teknoloji sayesinde, bahsi geçen grafik işlemciler mobil platformlara uyumlu hale gelmiş ve cep telefonu, tablet bilgisayar başta olmak üzere gömülü birçok sistemde kullanılmaya başlanmıştır. Bu çalışmada, gömülü sistemlerde kullanılmak üzere bir görüntü işleme kütüphanesi geliştirilmiş ve gömülü platformlar için üretilen grafik işlemcisi için eniyilemesi yapılmıştır. Geliştirilen kütüphane, görüntü işleme konularında standart haline gelmiş OpenCV (Open Computer Vision) kütüphanesi ile karşılaştırılmıştır. Günlük hayatta sıkça kullanılan bir görüntü işleme uygulaması olan "İnsan Yüzü Tespit Etme" uygulamasının geliştirilen kütüphane kullanılarak gerçeklenmesi ve başarım sonuçları da bu çalışmanın kapsamındadır.

Anahtar Kelimeler: Genel Amaçlı Grafik İşlemciler, Gömülü Yazılım Geliştirme, Görüntü İşleme, Başarım Eniyilemesi, Paralel İşleme

University : **TOBB University of Economics and Technology**

Institute : **Institute of Natural and Applied Sciences**

Science Programme : **Computer Engineering**

Supervisor : **Assoc. Prof. Oğuz ERGİN**

Degree Awarded and Date : **Master of Science – August 2014**

Osman Seçkin ŞİMŞEK

OPENCL BASED IMAGE PROCESSING LIBRARY FOR EMBEDDED GPGPUs AND A SAMPLE FACE DETECTION APPLICATION

ABSTRACT

Since graphic processing units (GPUs) can be used for general purpose calculations, its spectrum of use increased. Not only GPUs can be used rendering, moving and simulating 3 dimensional graphics, but also can be used where data parallelism applies. GPUs include hundreds of cores, which enables them to process hundreds of threads simultaneously that standart processors can not achieve. Data parallel applications include image processing, linear algebra applications, fluid dynamics, bioinformatics and such. GPUs are used in these fields in order to accelerate common algorithms and applications.

With emerging and downscaling new technologies, these GPUs can be used in mobile phone, tablets and various embedded systems. In this work, an image processing library has been developed and optimized for embedded GPGPU platforms. The library have been compared to the standardized library, OpenCV. An application of "Human Face Detection" which is widely used and its performance results are also the content of this work.

Key Words: GPGPU, Embedded Software, Image Processing, Performance Optimization, Parallel Processing

TEŞEKKÜR

Yüksek lisans ve lisans çalışmalarım boyunca bana yardım eden, yönlendiren, destekleyen değerli hocam ve tez danışmanım Doç. Dr. Oğuz ERGİN'e, derslerde ve diğer konularda deneyimlerinden yararlandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği öğretim üyelerine, tez konusu ile ilgili çalışmalarım sırasında büyük katkıları olan Mikroişlemciler Laboratuvarında çalışan tüm lisans ve yüksek lisans öğrencisi arkadaşlarıma, bu günlere gelmemdeki en büyük yardımcım anneme ve beni asla yalnız bırakmayan, en büyük destekçim olan eşim Merve'ye teşekkürü bir borç bilirim.

İÇİNDEKİLER

ÇİZELGELERİN LİSTESİ

ŞEKİLLERİN LİSTESİ

KISALTMALAR

Kısaltmalar	Açıklama
API	Application Programming Interface
CPU	Central Processing Unit (Genel Amaçlı İşlemci)
CUDA	Common Unified Device Architecture
FPGA	Field Programmable Gate Array
FPS	Frame Per Second
GLSL	OpenGL Shading Language
GPGPU	General Purpose Graphics Processing Unit
GPU	Graphics Processing Unit (Grafik İşlemcisi)
OpenCL	Open Computing Language
OpenCV	Open Computer Vision
RGB	Red Green Blue
SIMD	Single Instruction Multiple Data
SIMT	Single Instruction Multiple Threads
SM	Streaming Multiprocessor
SP	Streaming Processor

1. GİRİŞ

Günümüzde 3 boyutlu grafikleri üretebilmek için, genel amaçlı işlemcilerin yerine, bu iş için özelleşmiş olan grafik işlemciler kullanılmaktadır. Grafik işlemciler, kayan noktalı sayıların aritmetiksel işlemlerini daha hızlı yapmak için özelleşmiş, matematiksel işlem debisi bakımından ve bellek bant genişliği açısından normal işlemcilerden daha yüksek başarımlı işlemcilerdir. Kayan nokta işlemlerini hızlandırmadaki amaç, hareket edecek noktaların ekranda gösterileceği bir sonraki adımdaki yerini hesaplamaktır. Yüksek başarımlı matematiksel işlem kabiliyeti, grafik işlemcilerin yoğun işlem gerektiren uygulamalarda kullanımını gündeme getirmiştir. Yapılan geliştirmeler sonucunda da günümüzde kullanılan grafik işlemlerinin yanında genel amaçlı işlemler için de programlanabilen grafik işlemciler ortaya çıkmıştır.

Grafik işlemcilerin yanı sıra, günümüzde çok revaçta olan diğer bir teknoloji de gömülü cihazlardır. Gömülü platformlar, belirli bir amaca yönelik; işlem kapasitesi, pil kapasitesi, bellek kapasitesi gibi konularda daha sınırlı; çoğunlukla gerçek zamanlı işlem yapma gereksinimi olan ve genellikle masaüstü veya dizüstü ortamlara göre boyut olarak daha küçük olan platformları kapsar. Otomotiv, telefon, modem, oyuncak, beyaz eşya gibi günlük hayatta sık kullanılan cihazların içerisinde gömülü sistemler bulunmaktadır. Gömülü sistemlerin örneklerinden olan cep telefonu ile başlayan mobil platformlar, tablet ile daha da büyümüş; tüm üreticilerin akıllı telefonları desteklemesi ile de günümüzde en yaygın kullanılan gömülü platformları haline gelmiştir. Gömülü sistemlerin ortak ve belki de en önemli özelliği gerçek zamanlı işlem gereksinimine ihtiyaç duymasıdır.

Güncel gömülü işlemciler ilerleyen teknoloji ile daha düşük güç tüketse ve daha yüksek başarım ile çalışsa da, uygulama gereksinimleri de benzer hızda büyüdüğü için yetersiz kalabilmektedir. Özellikle yüksek işlem debisi gerektiren uygulamalarda gömülü işlemciler yetersiz kalmaktadır. Görüntü işleme, video işleme, şifreleme bu tip uygulamalara örnek olarak verilebilir.

Bu yetersizlikleri gidermek için grafik işlemcilerin kullanılması son yıllarda üzerinde çalışılan bir konudur. Grafik işlemciler yapıları gereği, çok sayıda işlem birimi bulundurmaktadır. Aynı anda çok yüksek bant genişliğinde veriyi işleyebilecek işlem birimlerini desteklemek için de, yüksek miktarda veriyi okuma ve yazma kapasitesine sahiptir. Bu özellikleri sayesinde normal işlemcilerin yavaş kaldığı gerçek zamanlı veri işleme konularında fayda sağlayabilmektedir. Masaüstü ve sunucu işlemcilerine yardımcı işlem birimi olarak grafik işlemcileri eklenmektedir. Ancak gömülü sistemlerde bu kullanım daha yenidir ve uygulama açısından bazı handikapları vardır. Gömülü sistemler daha düşük güç tüketimi ile çalışmak durumunda olmalarından ötürü, masaüstü platformlardaki kadar çok sayıda işlem birimine sahip olamamaktadırlar. Çalışma saat sıklıkları da (frekans) daha düşük olmaktadır. Sınırlı bellek ve hafıza ürünleri ile çalışmak durumunda olmaları da yine başka bir zorlaştırıcı faktördür. Bu gibi engeller göz önüne alındığında gömülü sistemlerde genel amaçlı işlemci ile grafik işlemcisinin eşzamanlı kullanılması büyük fayda sağlamaktadır.

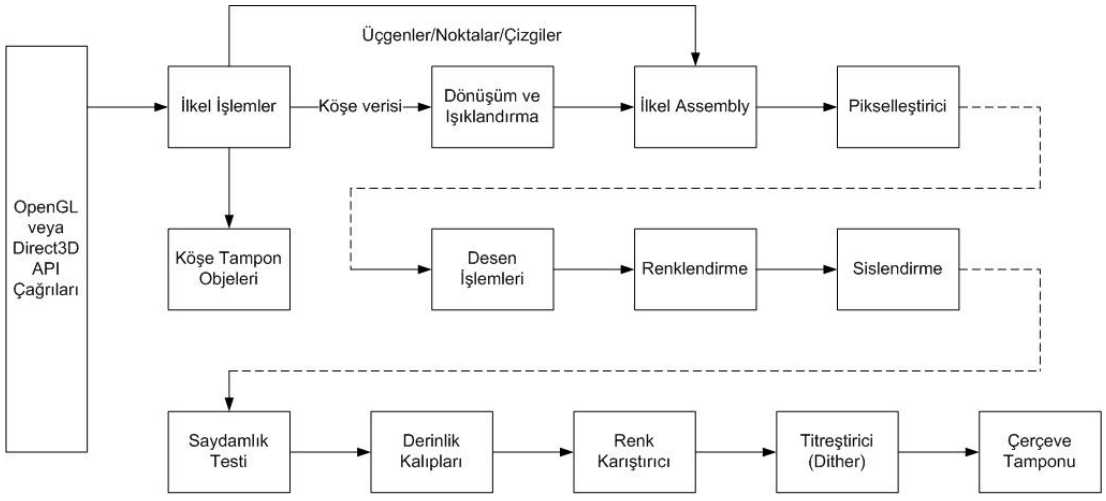
Bu çalışmada gömülü bir grafik işlemcisi üzerinde çalışması için optimize edilmiş, CPU ile GPU'yu beraber kullanabilen bir görüntü işleme kütüphanesi geliştirilmiştir. Temel görüntü işleme kabiliyetlerinden olan filtreleme, dönüşüm işlemlerinin yanı sıra, matris ve vektör işlemleri, renk dönüşümleri, içerik tespit algoritmaları gibi kabiliyetler bu kütüphanenin içeriğini oluşturmaktadır. Geliştirilen kütüphanenin başarım testleri için bir endüstri standardi haline gelmiş ve dünya çapında sıklıkla kullanılan OpenCV kütüphanesi kullanılmıştır. Masasaüstü platformlarda da çalışabilen kütüphanenin gömülü grafik işlemcisi üzerinde eniyilemesi yapılmıştır. Kütüphanenin kullanımının test edilebilmesi amacıyla ve sağladığı hızlanmayı göz önüne konulabilmesi için de örnek bir uygulama olan ve günümüzde sıklıkla kullanılan yüz tespit etme uygulaması da gerçeklenmiş ve sonuçlar açıklanmıştır.

2. TEMEL KONULAR

Bu kısım, grafik işlemcilerin yapısı, programlanabilirliği ve gömülü sistemlerde kullanılan grafik işlemcilerin farklılıklarını ele alarak sonraki kısımlarda bahsi geçen konuların anlaşılabilmesi açısından önemlidir.

2.1. GRAFİK İŞLEMCİLERİNİN YAPISI

Grafik işlemcilerini programlanabilmesi için öncelikle grafik işlemcilerin mimari yapısının bilinmesi gereklidir. Grafik işlemcilerin ilk çıkış noktası, CPU'ların hem normal işlemleri, hem de grafik işlemlerini beraber yaparken yeterli performans verememesidir. İlk üretilen grafik işlemciler sadece daha iyi grafik ortaya çıkarabilmek amacıyla üretilmiştir. İlk grafik işlemcilerde sadece belirli yöntemleri izleyen sabit bir grafik boru hattı bulunmaktaydı. Her çerçeve (frame) aynı boru hattından geçerek sonunda çerçeve tamponu (framebuffer) adı verilen birimde depolanmaktaydı. Buradan da çıkış aygıtı olan ekrana verilmekteydi. Grafik işlemcilerinin boru hattı Şekil 2.1' de gösterilmiştir.



Şekil 2.1. Grafik işlemcilerin sabit boru hattı şeması.

Grafik işlemcisinin kullanımı CPU tarafından uygulama programlama arayüzü (API) kullanılarak gönderilen çağrılarla yapılmaktadır. Program çalışması sırasında grafik işlemcisine gidecek giriş/çıkış buyrukları ile gerekli veri ve komutlar grafik işlemcisine gönderilir. Grafik boru hattının ilk aşamasında ilkel işlemler olan, gelen verileri köşe, üçgen, nokta, çizgi gibi bileşenlerine ayrıştırmaktır. İlk aşamadan sonra

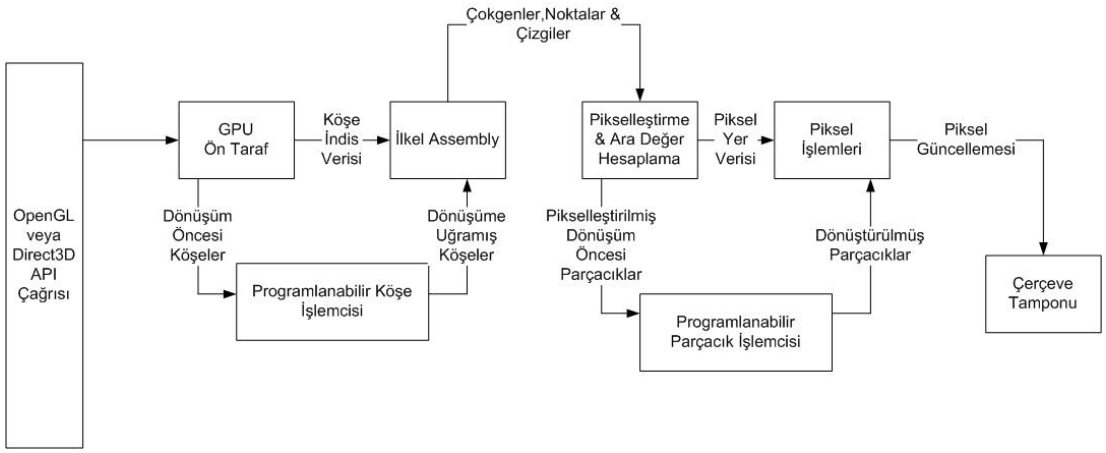
köşelere gerekiyorsa geometrik dönüşüm işlemleri ve ışıklandırma uygulanmaktadır. Işıklandırmadan sonra önceki aşamadan gelen üçgen ve nokta verileri ile birlikte birleştirme yapılır ve pikselleştiriciye gönderilir. Pikselleştiricide tüm bu toplanan veriler alınarak boşluklar doldurulur ve her bir cisim pikseller olarak ifade edilir. Boşluk doldurulması, köşelerinin renk ve koordinatları belli olan üçgenlerin içerisinde barındırdığı tüm piksellerin doldurularak içi dolu bir üçgen ortaya çıkarılmasıdır. Pikselleştirme aşamasından sonra desen işlemleri gelir. Eğer şekillerin üzerine bir resim dosyasından desen eklenecekse bu aşamada işlenmektedir. Desen aşamasını renklendirme ve sislendirme aşamaları izlemektedir. Desenli veya desensiz olan piksellerin renklendirmesi bu aşamada yapılır. Görüntüde sis olması durumunda ise sislendirme aşamasında eklemeler yapılır. Aynı şekilde saydamlık-opaklık düzenlemesi de cisimlere uygulanır.

Bu aşamalardan sonra görüntüdeki farklı cisimlerin yerlerine göre derinlik alıgısının yaratılabilmesi için derinlik işlemleri gelmektedir. Koordinat düzleminde daha önceden belli olan, görünümde hangi cismin önde veya arkada duracağının tayin edilmesi ve buna göre renk karıştırılması son aşamalarda gerçekleşir. En son, gerekli olduğu durumda titreştiricilik (dithering) işlemi de uygulanarak ortaya çıkan görüntü çerçevesi tampon bölgeye aktarılır ve buradan da ekranda gösterilir.

Bu denli fazla aşamalı bir boru hattının en büyük problemi bazı görünrülerde gerekli olmayan işlemleri de her görüntüye uygulamaktır. Örneğin, desen gerektirmeyen bir şekil için de yine desen işlemleri aşaması kullanılmaktadır. Her ne kadar desen işlemleri yapılmayacak olsa da yine de bu kısımda başarım kaybıyla sonuçlanan bir ek yük (overhead) oluşmaktadır.

Başarım kaybının önüne geçmek ve programcılara daha esnek bir modelleme ortamı sağlamak amacıyla grafik işlemcilerin mimarisi sabit boru hattından programlanabilir boru hattına geçirilmiştir. Programlanabilir boru hattının farkı, sabit boru hattındaki zorunlu işlemleri aynı şekilde yapmak, desen işlemleri gibi, sislendirme gibi, saydamlık gibi aşamaların programlanmasını ise programcının inisiyatifine bırakmaktır. Örnek vermek gerekirse, bir görüntüde sadece desen kullanılacaksa ve sislendirme kullanılmayacaksa, programlanabilir kısımlar ona göre ayarlanarak

başarım kaybı önlenebilmektedir. Programlanabilir grafik işlemcisinin boru hattı şeması Şekil 2.2'de gösterilmiştir.



Şekil 2.2. Programlanabilir grafik işlemcisi boru hattı.

Programlanabilir boru hattının kullanımında da yine sabit boru hattı mimarisindeki gibi CPU'dan gelen komut ve veri kümesi ile çalışma başlatılır. Gelen komutların içerisinde programlanabilir köşe işlemcisi ve programlanabilir parçacık işlemcisi için yazılmış kodlar da bulunmaktadır. Gelen kodlar programlanabilir işlem birimlerine ulaşır ve bu aşamalarda hangi işlemlerin yapılacağı belirlenir. Grafik işlemcisine gelen komutlar bir taraftan programlanmış köşe işlemcisinde gerekli işlemlere tabi tutulurken, öte yandan köşe indis ve verileri ile birleştirilir. Sonuçta ortaya çıkan çokgenler, üçgenler, noktalar pikselleştirilmek üzere pikselleştirme birimine aktarılır. Pikselleştirme biriminden çıkan parçacıklar da yine programlanmış parçacık işlem birimine gönderilir ve gerekli renk dönüşümleri uygulanarak yer bilgisi ile birleştirilir. Son olarak birleştirilen veriler güncellenerek ortaya çıkan görüntü çerçeve tamponuna aktarılır ve buradan da ekranda gösterilebilir.

Sabit boru hattına sahip grafik işlemcilerdeki dönüşüm ve ışıklandırma işlemleri, derinlik ayarlama işlemleri programlanabilir köşe işlemcisinde programlanabilmektedir. Renk ile ilgili işlemler, saydamlık, renk karıştırma işlemleri, desen işlemleri ise parçacık işlemcisi kısmında ele alınmaktadır. Bu birimlere genel olarak gölgelendirici (shader) denilmektedir.

Programlanabilir gölgelendiricilerin esneklik sağlaması da programlanabilme kapasitesinden kaynaklanmaktadır. Sabit boru hattına sahip grafik işlemcilerde programlama yapılırken, yalnızca önceden tanımlı dönüşümler cisimler üzerine uygulanabilmekteyken, programlanabilir boru hattı ile bu tamamen programcının inisiyatifine bırakılmıştır. Farklı bir dönüşüm modeli tanımlamak programcının elindedir.

Programlanabilir gölgelendirici (shader) sayesinde grafik işlemcinin yüksek başarımlı ve çok sayıdaki kayan nokta işlemcisine erişim mümkün olduktan sonra, bu işlem birimlerini yalnızca grafik amaçlı değil, genel amaçlı işler için kullanımı ortaya çıkmıştır. [1] çalışmasında grafik işlemcileri lineer cebir uygulamarını hızlandırmak için kullanılmıştır. Çalışmada programlanabilir gölgelendirici ünitelerine gerekli veriler aktarılmış, gölgelendirici kodunda istenilen işlemler yapılmış, çıkan sonuçlar da ekrana basılmak yerine tekrar CPU'ya aktarılmıştır. Bu şekildeki kullanım ile aslında genel amaçlı grafik işlemcilerinin temelleri atılmıştır.

Günümüzde kullanılan grafik işlemcilerin yapısı ise temelde programlanabilir model ile aynıdır. Farklılık ise, farklı programlanabilir birimler yerine birleşik gölgelendirici (unified shader) adı verilen birimler bulunmaktadır. Bu sayede ağır geometri işlemi gerektiren durumlarda köşe gölgelendirici, ağır piksel işlemi gerektiren durumlarda piksel gölgelendirici gibi davranışta bulunarak esnekliği sayesinde başarımdan ödün vermez [2].

Birleşik gölgelendiricilerin ortaya çıkması ile birlikte NVidia, genel amaçlı grafik işlemci mimarisini ve mimariye uygun programlama modeli olan CUDA (common unified device architecture) ile genel amaçlı grafik işlemcileri piyasaya sürmüştür. Genel amaçlı grafik işlemcilerin kullanımı sıklaştıktan sonra, Khronos Grubu programlama modelini standart haline getirmiş ve OpenCL (Open Computing Language) programlama modelini çıkarmıştır.

2.2. GENEL AMAÇLI GRAFİK İŞLEMCİ PROGRAMLAMA MODELİ

CUDA ve OpenCL grafik işlemcilerde genel amaçlı işlemler yapma imkanı sağlayan altyapılardan ilk ikisidir. CUDA 2006 yılında ve OpenCL de 2008 yılında piyasaya

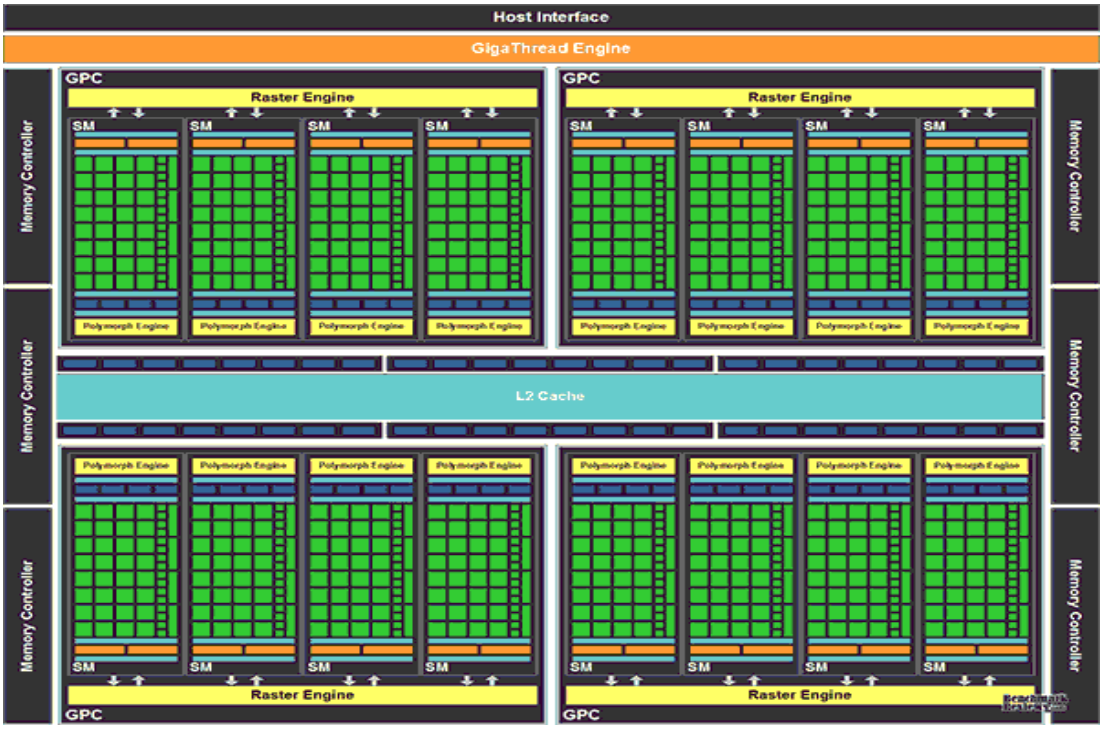
sürülmüştür. CUDA Nvidia firması tarafından geliştirilmiş ve yalnızca Nvidia grafik işlemcileri üzerinde çalışabilmektedir. Öte yandan Khronos Grup tarafından gereksinimleri ve platform tanımları yapılan OpenCL, telif hakkı kimseye ait olmayan, çapraz-platform uyumlu bir modeldir. Gereksinimleri gerçekleyen her üretici firma OpenCL desteği verebilmektedir. Bu sebeple OpenCL genel amaçlı işlemciler (CPU), grafik işlemciler (GPU), sinyal işleyiciler (DSP) ve alanda programlanabilir kapı dizileri (FPGA) gibi platformlarda destek verebilmektedir. Gömülü sistem platformları da OpenCL desteği bulunan platformlar arasına son yıllarda katılmıştır. OpenCL farklı konularda artıları olan platformları bir araya getirerek heterojen bir çalışma ortamı sağlayarak en yüksek başarıma ulaşabilir.

2.2.1. GPGPU Mimari Modeli

CUDA mimarisi ve platformu temel olarak tek buyrukta çok sayıda iş parçacığı (Single Instruction Multiple Threads) çalıştırma prensibine dayalıdır. OpenCL ile gerçeklenmiş işlemciler de temelde aynı mantığa dayalı olsa da bu şekilde isimlendirilmezler. Grafik işlemcilerin mimari yapısı Şekil 2.3'te gösterilmiştir. Normal işlemcilerde çok yer kaplayan ve karmaşık bir kontrol yapısı, büyük boyutlarda ve çok aşamalı önbellekler bulunurken, grafik işlemcilerde kontrol ve önbellek az yer kaplar ve basittir. Kalan silikon alanı ise işlem birimleri ile doldurulmuş olup, paralel işlem kapasitesi maksimize edilmiştir.

Şekil 2.3. CPU ve GPU mimarileri arasındaki farklar

Günümüzdeki genel amaçlı işlemciler gecikme zamanını küçültmeye yönelik birçok yöntemi içinde barındırır. İşlem gecikmelerini en aza indirgemek için boru hattı yöntemi kullanılır. Bellek ile veri alışverişi sırasındaki gecikmelerin azaltılabilmesi için de önbellek kullanılmaktadır. Aynı zamanda dallanma gecikmelerini azaltmak için dallanma öngörücüler, önbelleğe veriyi kullanımdan önce getirerek bellek erişimini en aza indirgemek için ön-yakalama (prefetch) yöntemleri de kullanılır. Genel amaçlı işlemciler görev paralelliği ve rastgele erişim yönünde başarılıdır. Görüldüğü gibi genel amaçlı işlemcilerde gecikme zamanını en aza indirgemek ön plandadır. Grafik işlemcilerde ise, asıl olarak grafik işleme amacıyla tasarlanması dolayısıyla, büyük önbellekler, ön-yakalayıcı, karmaşık bir dallanma öngörücüsü bulunmamaktadır. Bu birimlerin yerine daha fazla işlem birimi konularak paralel yapı daha da geliştirilmiş ve içerisine aynı anda binlerce iş parçacığını çalıştırabilecek çoklukta işlem birimi konulmuştur. Mimarideki farklılık seri işlemlerde veya dallanması çok olan işlemlerde grafik işlemcisinin aleyhine olmasına rağmen, aynı işlemleri büyük boyuttaki veriler üzerinde uygulamaya elverişlidir. Grafik işlemcilerin artısı da bu şekildeki uygulamalarda kendini göstermektedir.



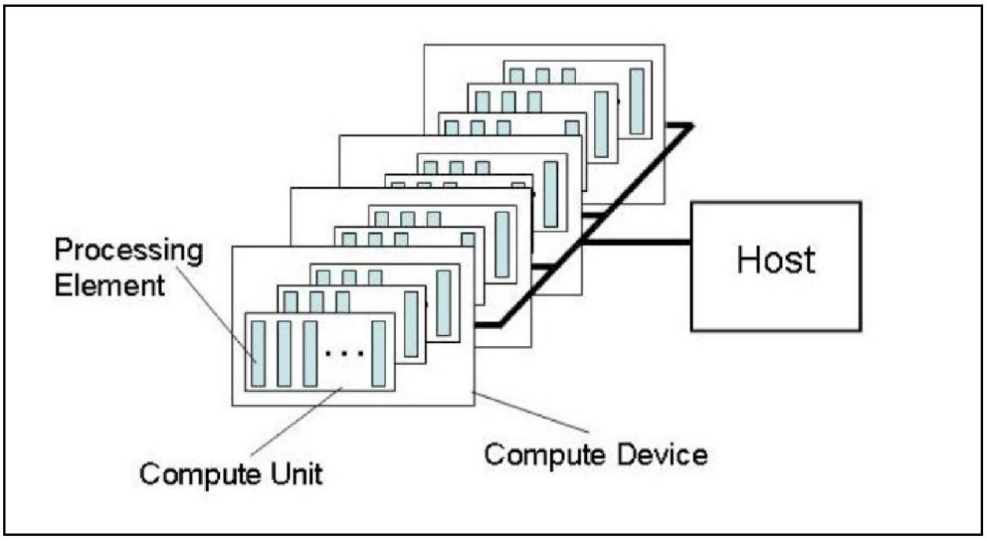
Şekil 2.4. Genel amaçlı grafik işlemcilerin iç yapısı

Şekil 2.4'te görülen grafik işlemcisi mimarisindeki akış çoklu-işlemcisi (SM), genel amaçlı işlemci mimarisindeki çekirdek kavramına benzemektedir. CPU'lardaki çekirdekler birden çok boru hattı ve çok sayıda işlem birimi içerebildiği gibi, SM'ler de çok sayıda akış işlemcisi (SP) içerir. Akış işlemcileri tek buyruk çok veri (SIMD) prensibi ile işlem yapabilme kapasitesine sahiptir. Tek bir buyruktaki işlemi aynı saat vuruşunda tasarımdan tasarıma değişmekle birlikte 2, 3, 4, 8 veya 16 kelimelik veri üzerine uygulayabilmektedir. GPGPU üzerinde yapılan işlemlerin çoğu basit aritmetik mantık işlemleri şeklindedir. Daha karmaşık ve donanımsal olarak gerçeklenmiş olan trigonometrik, logaritmik vb. fonksiyonlar için özelleşmiş işlem birimleri mevcuttur.

2.2.2. OpenCL Platform Modeli

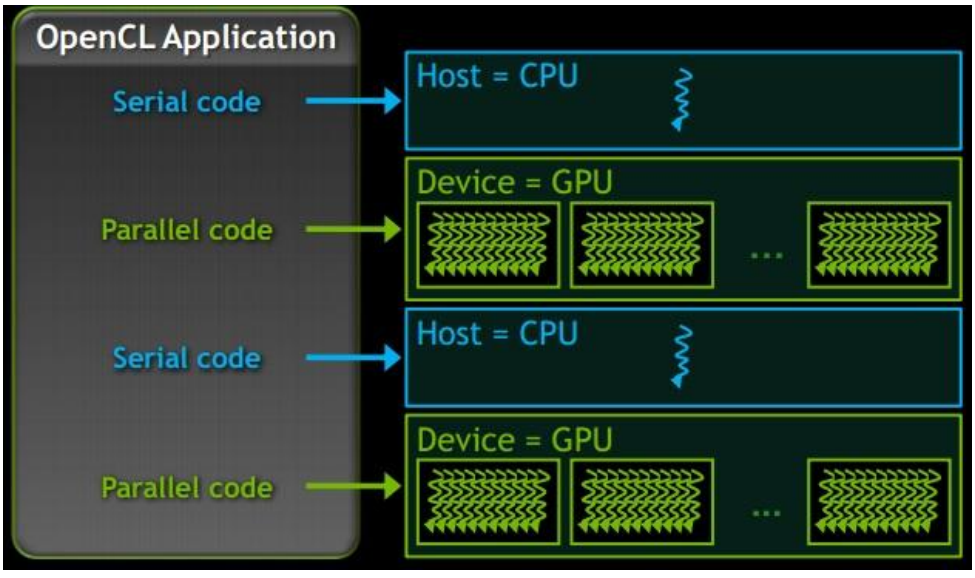
Genel amaçlı işlemler için kullanılabilen grafik işlemcileri yalnız başına çalışamamaktadır. Her zaman bir platformda genel amaçlı işlemci veya işlemcilerin yanında, bir ek işlemci olarak yer almak zorundadır. Gelişen teknoloji ile birlikte aynı yonga üzerinde hem grafik işlemci hem de genel amaçlı işlemciler bulunabilmektedir. Her halükarda, işletim sisteminin üzerinde çalışması gereken bir genel amaçlı işlemci, grafik işlemcisinin de bu CPU ile haberleşmesini sağlayan bir sürücü yazılımı bulunmalıdır.

Önceden bahsedildiği gibi OpenCL yalnızca grafik işlemcileri değil, sinyal işleyici veya FPGA gibi cihazları da desteklemektedir. OpenCL destekli cihazların her birine OpenCL modelinde işlem aygıtı (compute device) denilmektedir. İşlem aygıtlarının hepsi sunucu (host) adı verilen genel amaçlı işlemciye bağlıdır. OpenCL de CUDA da aslında çalışan programın yalnızca bir bölümünü oluşturmaktadır. İşletim sistemi ile ilgili çağrıları, verileri okuma-yazma ve giriş çıkış işlemlerinin hepsi sunucu kısmında yürütülmektedir. OpenCL'in çalışma modeli ile ilgili ayrıntılı bilgiler bölüm 2.2.3'te verilecektir. Şekil 2.5'te OpenCL'in platform modeli gösterilmiştir. İşlem aygıtları bir sunucuya bağlıdır. İşlem aygıtları işlem birimlerine ayrılmaktadır. İşlem birimlerinin mimari yapıdaki karşılığı akış çoklu-işlemcisidir. Aygıtlarda bir veya birden fazla işlem birimi olabilir. İşlem birimi de işlem elementlerine bölünmektedir. İşlem elementlerinin mimarideki karşılığı da akış işlemcisidir.



Şekil 2.5. OpenCL platform modeli

GPGPU platformlarında sunucu ve işlem aygıtlarının çalışma biçimi Şekil 2.6'da gösterilmiştir. Görüldüğü gibi genel amaçlı işlemci aslında grafik işlemcisine paralel hale getirilmiş iş parçacıkları gönderip sonuçları tekrar kendisinde toplamaktadır.



Şekil 2.6. GPGPU paralelleştirmesinin çalışma biçimi

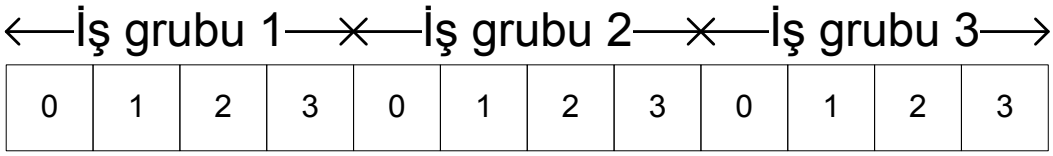
2.2.3. OpenCL Çalışma Modeli

OpenCL uygulaması 2 ana parçadan oluşmaktadır. Sunucu programı ve bir veya birden fazla çekirdek programı. Çekirdek, grafik işlemcisi üzerindeki çalışan ve OpenCL C dilinde yazılan program parçasıdır. Sunucu programın nasıl çalışacağını

OpenCL belirlemez. OpenCL yalnızca çekirdekler ile sunucu arası iletişimin nasıl olacağını belirler.

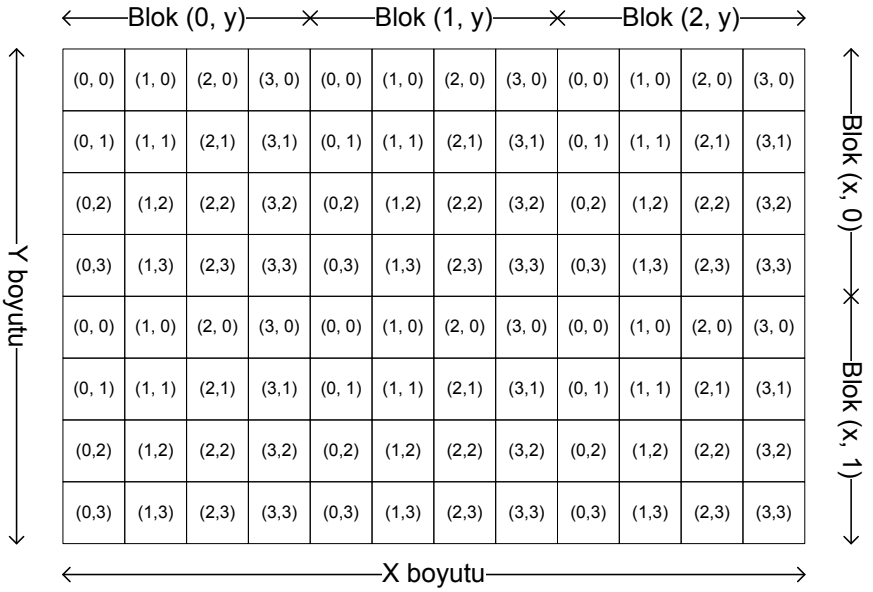
OpenCL'in çalışma modelini anlayabilmek için öncelikle OpenCL ile bir çekirdek programın nasıl çalıştığının anlaşılması gerekmektedir. Çekirdekler sunucu program içerisinde, yani sunucu üzerinde tanımlanırlar. Çekirdeğin çalışması için sunucu program aygıt üzerine OpenCL çekirdeğinin çalıştırılacağına dair komutu gönderir. Komutu alan aygıt, OpenCL için bir tam sayı indis uzayı oluşturur. Çekirdek, bu tam sayı uzayının her bir elemanı için çalıştırılır. Çalışan çekirdeğin her bir birimine iş-kalemi denilir. İş-kalemleri birleşerek bağımsız iş-grubunu oluşturur. Bir işlem aygıtı içerisindeki iş-grubunu oluşturan tüm iş-kalemleri donanımsal olarak aynı anda çalıştırılırlar. OpenCL iş-gruplarının veya çekirdeklerin çalışmasını seri olarak gerçekleştirebilir, ancak bir iş grubunu oluşturan iş-kalemlerinin her birisinin paralel olarak çalışacağını garanti eder.

Tam sayı indis uzayı N boyutlu bir uzaydır. N-boyutlu olmasından ötürü NB-menzil uzayı olarak adlandırılır. Günümüzdeki OpenCL sürümlerinde N; 1, 2 ve 3 olabilmektedir. Bir boyutlu olan vektörlerden üç boyutlu olan nesnelere kadar bütün boyutları indisleyebilmektedir. Çalışma sırasında N boyutlu indisler iş-kalemlerine atanarak donanım üzerinde hangi işlem birimine gideceği belirlenir. NB-menzil uzayı iş-gruplarına bölünür, iş grupları da iş-kalemlerine bölünerek tüm uzayın indislenmesi yapılır. Bölütlendirme ve bu boyutların belirlenmesi sunucu tarafında yapılır, komutlar yardımıyla OpenCL ortamına (context) gönderilir.



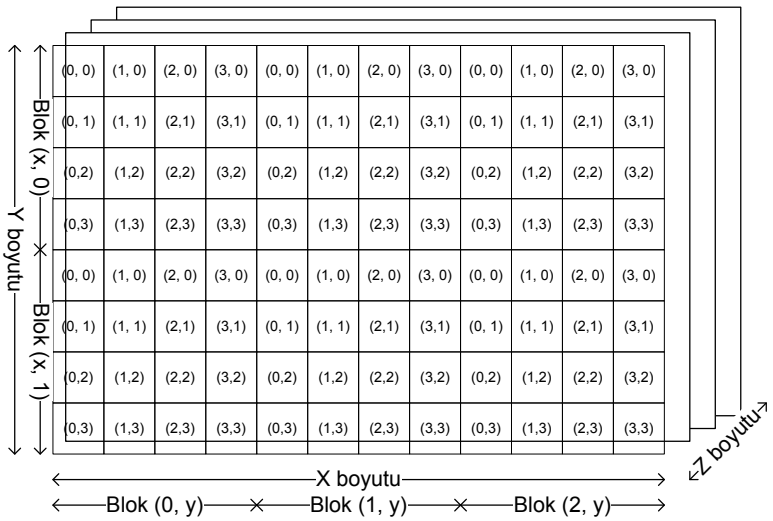
Şekil 2.7. Bir boyutlu vektörün iş-kalemlerine ayrılması

Şekil 2.7. Bir boyutlu vektörün iş-kalemlerine ayrılması gösterilmiştir. Her iş-kaleminin iş-grubu içerisindeki indisi ve genel olarak indis uzayındaki indisi yerel ve evrensel olarak hafızada tutulur. Çekirdek içerisinde iş-kalemlerinin indisleri çağrılırken her iki değer de kullanılabilmektedir.



Şekil 2.8. İki boyutlu matrisin iş-kalemlerine ayrılması

Şekil 2.8'de iki boyutlu bir indis uzayının şeması gösterilmiştir. Bu modelde ise hem x hem de y koordinat düzlemi üzerinde iş-kalemlerinin indisleri belirlenir. İndisler çağrılacağı zaman iki değer de kullanılır.



Şekil 2.9. Üç boyutlu matrisin iş-kalemlerine ayrılması

Şekil 2.9'da üç boyutlu indis uzayının şemasıdır. Bu modelde de x, y ve z koordinat düzlemlerindeki yerler dikkate alınarak iş-kalemlerine erişim sağlanabilir.

Bu modellerin hepsinde karşılaşılan sıkıntı aynıdır. OpenCL C dili kullanılarak yazılan programlarda verilere erişimde indislemeden kaynaklanan sorunların kaynağı ise bellek uzayının tek boyutlu olmasıdır. Yani programımızda 2 veya 3 boyutlu tanımlanmış diziler aslında bellekte yine tek boyut olarak birbiri ardına sıralanmış şekilde tutulmaktadır. Sorunun üstesinden gelebilmek için boyutlardaki indisleme kullanılarak tek boyutta bellekte nereye erişileceği hesap edilir.

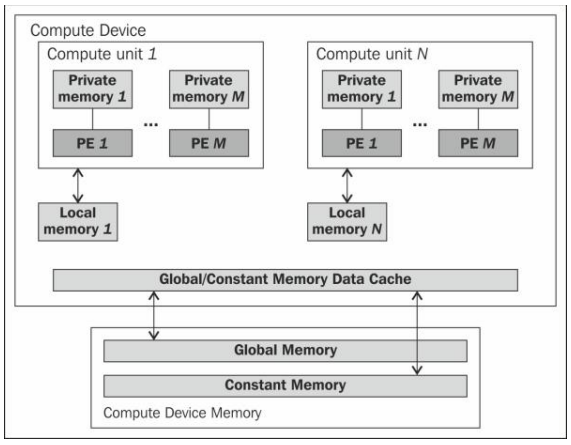
OpenCL ortamını oluşturan öğeler; OpenCL destekli aygıtlar, çekirdekler, program nesneleri ve bellek nesneleridir. OpenCL ortamı sunucu programı üzerinde yaratıldığında, çalışabilmek için bu dört öğeye ihtiyaç duymaktadır. Aygıtlar, OpenCL kodu çalıştırılmadan önce kontrol edilmelidir. OpenCL destekli bir aygıt olmaması durumunda programın çalışması mümkün değildir. Çekirdekler OpenCL C dilinde yazılmış ve aygıt üzerinde çalışacak olan program parçalarıdır. Program nesneleri, çekirdeğin veya çekirdeklerin gerekli konfigürasyon parametreleri ile birlikte aygıt üzerinde çalışabilecek hale getirilmiş halidir. Program nesnelerini elde etmenin iki yolu vardır. İlk yol, çalışma zamanı sırasında çekirdeklerin derlenerek ortaya program nesnesinin çıkmasıdır. İkinci yol ise, çevrimdışı bir derleyici yardımı ile program nesnesinin belirli bir aygıta göre derlenmesi ile oluşur. Her iki yöntemde de elde edilen program nesnesi aynı işlevdedir. İkinci yöntem, çalışma zamanı sırasında ek bir derleme yükü olmamasından ötürü ilk yöntemden daha yüksek başarıma sahiptir. Bellek nesneleri ise, çekirdeklerin içerisindeki fonksiyonlarda kullanılan parametreleri, değişkenleri ve yaratılan statik veya dinamik nesnelerin tümüdür. CPU'nun ve aygıtın farklı bellek adreslemesi olduğu için (gömülü platformlar bu örneğe dahil değildir.) bellek nesneleri sunucu tarafında oluşturularak aygıt üzerine API çağrısı ile yazılır.

OpenCL ortamı oluşturulduktan sonra programın çalışabilmesi için komutların aygıt üzerine gönderilmesi gerekmektedir. Sunucu ve aygıt arasındaki bu etkileşimi sağlayan yapıya komut-kuyruğu adı verilir. Komut-kuyruğu ortam oluşturulduktan sonra tanımlanır ve tek bir aygıt ile eşleştirme yapılır. Bir komut-kuyruğu birden fazla aygıtta çalışamaz. Eşleştirme yapıldıktan sonra komutlar komut-kuyruğu yardımı ile aygıta gönderilir ve aygıt üzerinde komutların zamanlaması (scheduling) yapılır. OpenCL'de çekirdek çalıştırma komutları, bellek komutları ve

senkronizasyon komutları olmak üzere üç adet komut vardır. Tipik bir OpenCL programında, ortam ve komut-kuyruğu tanımlanır, bellek ve program nesneleri tanımlanır ve komutlar komut-kuyruğuna gönderilir. Bellek ve program nesnesi sunucudan aygıt üzerine gelir ve çalışma zamanı başlar. Çalışma bittikten sonra sonuçların bulunduğu bellek nesneleri tekrar sunucu üzerine alınabilir. Birden çok çekirdeğin çalışması durumunda çekirdeklerin etkileşmesi ihtiyacı ortaya çıkabilir. Bu durumda ise, iki çekirdek arasında bellek işlemleri ile ilgili sorunlar ortaya çıkabilir. İlk çekirdek işini bitirmeden ikinci çekirdek çalışmaya başlarsa yazılmayan değerler okuyacağından program hatalı sonuçlar üretecek veya hiç sonuç üretemeyerek hata verecektir. Bunun önüne geçebilmek için de üçüncü tip komut olan senkronizasyon komutları kullanılmaktadır.

2.2.4. OpenCL Bellek Hiyerarşisi

OpenCL destekli aygıtların desteklemeleri gereken bir bellek hiyerarşisi vardır. OpenCL'i destekleyen her aygıt kendisine ait özelleştirmeler ve iyileştirmeler yapabilirler, ancak genel bellek hiyerarşisi değişmemektedir. OpenCL'de Şekil 2.10'da gösterildiği gibi CPU üzerinde bulunan sunucu belleği ve GPGPU üzerinde bulunan; evrensel bellek, sabit bellek, yerel bellek ve özel bellek olmak üzere 5 çeşit bellek alanı vardır.



Şekil 2.10. OpenCL bellek hiyerarşisi

- Sunucu belleği: CPU'ya ait bellek alanıdır. Aygıtın erişimi bulunmamaktadır. Yalnızca tanımlı OpenCL ortamı içerisinde, aygıt ile etkileşimi sağlamak amacıyla kullanılır. Sunucuda çalışan kod parçaları ve aygıta gönderilen, sonrasında da sonucu geri okumakta kullanılan bellek alanı burasıdır. Aygıtın bu bellek alanına erişimi yoktur, sunucunun yazma/okuma erişimi vardır.
- Evrensel bellek: aygıt üzerinde bulunan, boyut olarak en büyük bellektir. Tüm iş-gruplarına ve iş-kalemlerine yazma/okuma izni verir. Önbellekleme mekanizması opsiyoneldir, bazı aygıtlarda mevcut olmasına karşın bazı aygıtlarda ihmal edilebilir düzeydedir. Sunucu tarafından bu alana veri yazılıp okunabilmektedir.
- Sabit bellek: OpenCL ortamı oluşturulduktan sonra bu bellek alanına yazılan veriler, çekirdeğin çalışması süresince değişime uğramazlar. İş-kalemleri bu alanı yalnızca okuma amaçlı kullanabilirler. Sunucu tarafından bu alana veri yazılıp okunabilmektedir. Evrensel bellekte tutulan sabit bellek elemanları, hızlı erişim için önbelleğe aktarılmaktadır. Sabit bellekte tutulan verilere erişim bu yüzden evrensel bellekte tutulan verilere göre daha hızlıdır. Sabit bellekte tutulan verilerin ömrü, uygulamanın çalışma süresini kapsar.
- Yerel bellek: her iş-grubunun kendisine ait bir yerel belleği bulunmaktadır. İş-grubu içerisindeki tüm iş-kalemleri bu yerel belleği kullanır. İş-grubu dışından erişim gerçekleşemez. Yazma/okuma erişim iznine sahiptir. Yerel bellek yonga üzerinde bulunur bu yüzden erişim hızlıdır. Çekirdek çalışması sona erdiğinde yerel bellekteki verilerin varlığı sona erer.
- Özel bellek: her iş-kalemine ait bellek alanıdır. Yazmaç düzeyindedir. Bu bellek alanındaki veriler başka iş-kalemleri tarafından okunamaz.

Bellek modelinde hızlarına ve paralelliklerine göre birimleri sıralamak gerekirse, en yavaş ve en seri erişim evrensel belleğe yapılmaktadır. Sabit bellek önbellekleme mekanizması sayesinde evrensel bellekten daha başarılı ve daha paralel erişime açıktır. Yerel bellek, programcı tarafından kullanılması gereken, çok hızlı ve paralel bir yapıdır. Özel bellek ise en hızlı erişimin sağlandığı yapıdır. Her iş-kalemi için ayrı özel bellek alanı bulunduğu için parallelik söz konusu değildir. Belleğe hızlı

erişim sağlamak ve programın başarımını arttırmak için hızlı bellek birimlerini sık sık kullanmak gerekmektedir.

2.3. GÖMÜLÜ PLATFORM GRAFİK İŞLEMCİLERİ

Khronos grubunun diğer bazı standartlarının aksine, OpenCL gömülü sistemler için ayrı bir spesifikasyona sahip değildir. Ancak gömülü sistemler için bir profil oluşturulmuştur. Gömülü profil OpenCL spesifikasyonunun bir alt kümesidir. Zorunlu olarak gerçeklenmesi gerekenlerin yanı sıra bazı fonksiyonaliteler opsiyonel olarak bırakılmıştır. Bu fonksiyonalitelere örnek olarak, "long", "double", "half" gibi değişken türlerini destekleme zorunluluğunun olmaması, yuvarlama modlarının hata oranının daha esnek olması, 3 boyutlu "image" veri türünün desteklenmek zorunda olmaması gösterilebilir.

Gömülü platformların genel problemleri güç ve bellek kısıtlarıdır. Düşük güç tüketimini sağlamak için işlemcilerin saat sıklıkları düşürülür, daha özelleşmiş işler için tasarlanmış işlemciler kullanılır. Bellek uzayı da aynı şekilde daha kısıtlıdır. Gömülü grafik işlemcilerde de durum aynıdır ve şu kısıtlar gömülü grafik işlemcilerdeki başarımın masaüstü grafik işlemcilere göre düşük olmasının sebebidir:

- Bellek boyutunun düşük olması,
- Bellek bant-genişliğinin düşük olması,
- Bellek saat sıklığının düşük olması,
- Grafik işlemci saat sıklığının düşük olması,
- Alan kısıtı olduğu için daha az işlem birimine sahip olması,
- Donanımsal bir şekilde aynı anda işlenebilecek iş parçacığının az olması,
- Yonga üzeri bellek alanının ve yazmaç sayısının az olması.

3. OPENCL GÖRÜNTÜ İŞLEME KÜTÜPHANESİ

Bu çalışmada bahsedilen OpenCL kullanılarak geliştirilen görüntü işleme kütüphanesi, hem Windows hem de Linux ortamında geliştirilmiştir. Çalışma ortamı bir gömülü sistem ortamı olan Freescale i.Mx6q olup içerisinde QuadCore ARM Cortex A9 işlemci ve Vivante GC2000 grafik işlemcisi barındırmaktadır. Çalışma ortamının işletim sistemi Gömülü Linux dağıtımlarından olan Ubuntu 11.10'dur. Geliştirme ortamı masaüstü bilgisayarlar olduğu için, gömülü platforma kod üretme işlemi Freescale firmasının sağladığı çapraz-derleyici ile gerçekleşmiştir.

Kütüphanenin sağladığı faydanın boyutunu ölçmek için, kütüphaneyi karşılaştıracak bir referans seçilmesi gerekmektedir. Referans olarak görüntü işleme konusunda dünya çapında en sık kullanılan kütüphane olan OpenCV kütüphanesi seçilmiştir. Yaygın kullanımı ve gömülü platform desteği OpenCV'yi anlamlı bir seçenek haline getirmiştir. Karşılaştırmalar, gömülü platformda ve masaüstü platformda yapılmıştır. Her ikisinde de OpenCV, işlemci üzerinde ve geliştirilen kütüphane, grafik işlemcisi üzerinde koşturulmuştur.

3.1. FONKSİYONLAR

Kütüphane en temel görüntü işleme fonksiyonlarını içermektedir. Gömülü platformlarda çok karmaşık fonksiyonların işlenmesi kullanılmadığı için kütüphane gereksinimlerine dahil edilmemiştir. Kütüphane, içerik ve fonksiyonalite açısından 6 alt birime bölünmüştür. Bu alt birimler ve içerdiği fonksiyon sayıları Çizelge 3.1'de gösterilmiştir.

Çizelge 3.1. Kütüphanenin içerdiği fonksiyonlar

Kütüphane Modülü	**İçerdiği Fonksiyon Sayısı**
Filtreleme Fonksiyonları	9
Geometrik Dönüşüm Fonksiyonları	6
Matris İşlemleri Fonksiyonları	33
Renk Dönüşüm Fonksiyonları	10
İçerik Tespit Fonksiyonları	6
Morfoloji Fonksiyonları	5
TOPLAM	**69**

3.1.1. Matris İşlemleri Fonksiyonları

Matris fonksiyoları, içeriğinde en temel operasyonları barındırmaktadır. Matris operasyonları, matris-matris arasında gerçekleşen ve matris ile bir skalar değer arasında gerçekleşen operasyonlardan oluşmaktadır. Diğer bir tür de yalnız bir matris üzerinden, ortalama, standart sapma, minimum ve maksimum değer hesaplama gibi fonksiyonlardır. Matris işlemlerinin içerdiği fonksiyonlar ve yaptığı işlemler aşağıdaki listede açıklanmıştır.

- Matrisler arası işlem: iki matris arasında toplama, çıkarma ve çarpma işlemleri. Fonksiyonlar iki matris arasında çalışır. Çarpım işlemi matris çarpımını içermektedir. Bunların yanı sıra, iki matrisin elemanları arasındaki mutlak farkın hesaplanması da bu kategoridedir. Elemenların farkı hesaplanıp mutlak değer alınarak sonuca ulaşılır.
- Matris skalar arası işlemler: bir matris ve bir skalar sayı arasında gerçekleşen dört işlemi içermektedir. Matrisin her elemanı ile bir skalar sayı, çarpma, bölme, toplama, çıkarma işlemlerine tabi tutulur. Denklem 3.1'de çarpma ve 3.2'de ise bölme işlemi gösterilmiştir.

$$\begin{matrix} 3 & 14 & 22 \\ 1 & 36 & 4 \\ 51 & 3 & 12 \end{matrix} * 3 = \begin{matrix} 9 & 42 & 66 \\ 3 & 108 & 12 \\ 153 & 9 & 36 \end{matrix} \quad (3.1)$$

$$\begin{matrix} 26 & 11 & 24 \\ 4 & 12 & 5 \\ 3 & 1 & 48 \end{matrix} \div 2 = \begin{matrix} 13 & 5.5 & 12 \\ 2 & 6 & 2.5 \\ 1.5 & 0.5 & 24 \end{matrix} \quad (3.2)$$

- Kanal işlemleri: üç farklı kanaldan gelen verileri tek bir matris haline getirmek veya üç kanallı bir resim verisini içeren matrisi üç farklı renk matrisi şekline dönüştürmek için bu işlemler kullanılmıştır. Kullanım alanı, yalnızca bir renk üzerinde filtreleme veya tespit işlemleri gerçekleştirilmesi mantığına dayanır. Ayrılan kanallar daha sonra birleştirilebilir.

- Transpoz: transpoz işlemi bir matrisin satır ve sütunlarını yer değiştirmek amacıyla kullanılır. Lineer cebir işlemlerinde ve görüntü işleme algoritmalarında sıkça kullanılmaktadır.
- Ortalama, standart sapma, minimum ve maksimum işlemleri: bu işlem kümesi tek matris üzerinde hesaplama yapmaktadır. Minimum ve maksimum işlemleri matristeki en büyük ve en küçük değerleri bulur. Ortalama işlemi matristeki tüm elemanların toplamının eleman sayısına bölünmesidir. Denklem 3.3'te gösterilmiştir. Standart sapma ise, Denklem 3.4'te gösterildiği şekilde, her elemandan ortalamanın farkı alınarak karesi hesaplanır. Bu karelerin toplamı, toplam eleman sayısına bölünerek karekökü hesaplanarak bulunur.
-

$$\begin{bmatrix} 2 & 4 \\ 5 & 4 \\ 4 & 7 \\ 5 & 9 \end{bmatrix} \; matrisinin\ ortalaması \; \frac{2+4+5+4+4+7+5+9}{8} = 5 \tag{3.3}$$

$$\begin{bmatrix} (2-5)^2 & (4-5)^2 \\ (5-5)^2 & (4-5)^2 \\ (4-5)^2 & (7-5)^2 \\ (5-5)^2 & (9-5)^2 \end{bmatrix} \qquad \sqrt{\frac{9+1+0+1+1+4+0+16}{8}} = 2 \tag{3.4}$$

- Logaritma ve üssel: tüm matris elemanlarının logaritmasının hesaplanması ve verilen bir üssünün hesaplanması fonksiyonlarını içermektedir.

Matris işlemleri için kullanılan tüm fonksiyonlar vektörlere de uygulanabilmektedir. Matris işlemlerinin gerçeklemesi sırasında kullanılan yöntemler ve eniyileme metotları 3.2 Fonksiyonların Gerçeklenmesi ve Kullanılan Eniyileme Yöntemleri başlığın altında incelenmiştir.

3.1.2. Filtreleme Fonksiyonları

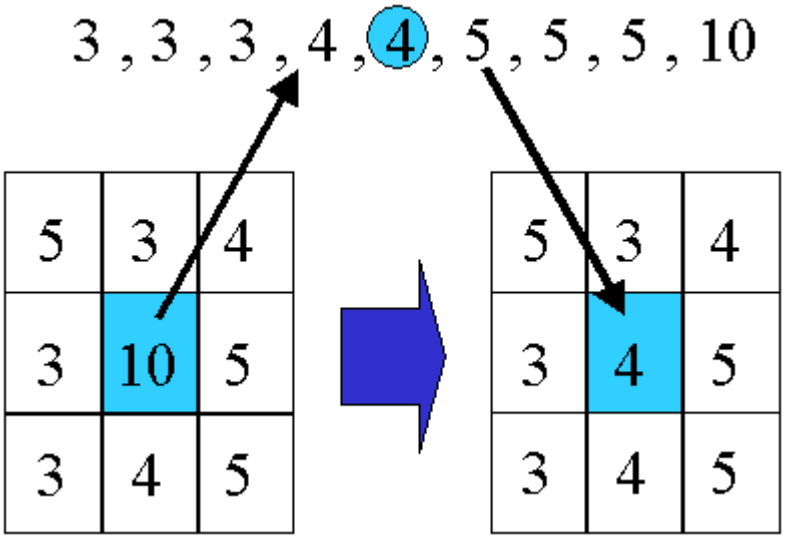
Filtreleme fonksiyonları en sık kullanılan, temel filtrelerden oluşmaktadır. Bunların yanı sıra, filtre oluşturmak ve bu filtreyi istenilen resim veya matris üzerinde

uygulamak için gereken konvolüsyon işlemi de bu fonksiyon kümesine dahildir. Filtreleme fonksiyonları ve yaptıkları işlemler aşağıdaki listede verilmiştir.

- Konvolüsyon: konvolüsyon işlemi bir filtreyi bir matris, vektör veya resim üzerinde, her eleman için, filtrenin her elemanı ile uygulanan veri yapısının o iterasyondaki elemanlarının çarpımlarını toplama yöntemi ile elde edilir. Tek boyutlu konvolüsyon işleminin tek iterasyonu Denklem 3.5'te gösterilmiştir.

$$B' = a * A + b * B + c * C \tag{3.5}$$

- Ortanca Filtresi: ortanca filtresi, uygulandığı alandaki sıralanmış değerlerden ortanca olanı bularak, uygulanan pikseldeki yeni değere bulunan ortanca değerin konulması ile uygulanır. Uygulanma yöntemi şekilde Şekil 3.1'de gösterilmiştir. Filtrenin uygulandığı alandaki değerler sıralanmış ve yeni değer olarak sıralamanın ortasındaki değer seçilmiştir.



Şekil 3.1. Median filtresinin uygulanması

- Laplacian Filtresi: görüntü işleme uygulamalarında kullanılan filtre, Laplacian operatörünün ayrık sürümünü kullanır. Denklem 3.6'da Laplacian filtresinin tek boyutlusu, Denklem 3.7'de ise iki boyutlusu gösterilmiştir.

Denklem 3.8'de ise yine iki boyutlu Laplacian operatörü vardır ancak bu sefer köşe değerler de hesaba dahil edilmiştir. Laplacian filtresinin uygulanması ise Laplacian operatörü ile matris üzerinde konvolüsyon işlemi gerçekleştirilmesi ile elde edilir.

1 Boyutlu Laplacian operatörü: $\begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$ (3.6)

2 Boyutlu Laplacian operatörü: $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ (3.7)

2 Boyutlu Laplacian operatörü, köşeler dahil: $\begin{bmatrix} 0.5 & 1 & 0.5 \\ 1 & -6 & 1 \\ 0.5 & 1 & 0.5 \end{bmatrix}$ (3.8)

- Gaussian Filtresi: görüntü işlemede kullanılan Gauss filtresi, bulanıklık amacıyla kullanıldığından Gaussian bulanıklığı olarak da adlandırılır. Gaussian operatörünün resim üzerine konvolüsyon ile uygulanır. Gaussian operatörünün 1 Boyutlu hesaplaması Denklem 3.9'da 2 Boyutlu hesaplaması ise Denklem 3.10'da gösterilmiştir.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}} \quad (3.9)$$

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2-y^2}{2\sigma^2}} \quad (3.10)$$

Filtrenin parametresinde verilen sigma değeri ve filtrenin boyutu ile filtre operatörünün değerleri hesaplanır.

- Sobel Filtresi: sobel operatörü, ayrık türev operatörüdür. Küçük ve ayrışabilen bir operatör olması sebebiyle çok tercih edilir. Resimlerde kenar

tespiti yapmak amacıyla uygulanır. Konvolüsyon ile bir resim üzerine uygulanır. Yatay, dikey ve sobel operatörleri Denklem 3.11'de gösterilmiştir. Denklemdeki * işlemi konvolüsyonu temsil etmektedir.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \tag{3.11}$$

- Kutu Filtresi: kutu filtresi de bir bulanıklaştırma filtresidir. Filtrenin boyutuna göre, uygulandığı piksellerin toplamını alarak, hedef pikselin yeni değeri olarak ataması yapılır. Konvolüsyon kullanımı da olabileceği gibi, daha basit ve hızlı algoritmalar da kullanılabilmektedir.
- Normalize Kutu Filtresi: normalize kutu filtresi, kutu filtresi ile temelde aynıdır. Farkı ise, toplam değil ortalama hesaplaması yapılmasıdır.
- Bulanıklık Filtresi: bulanıklık filtresi normalize kutu filtresi ile aynıdır.
- Çift Taraflı (Bilateral) Filtre: çift taraflı filtre, kenar muhafaza eden bir bulanıklık filtresidir. Resmin genelini bulanıklaştırır ancak kenar belirten kısımlara zarar vermez. Çift taraflı filtrenin gerçekleştirimi [3] çalışmasındaki gibi yapılmıştır.

3.1.3. Geometrik Dönüşüm Fonksiyonları

Geometrik dönüşüm fonksiyonları resmin şekilsel yapısı ile alakalı değişiklikleri gerçekleştirmek için kullanılır. Geometrik dönüşüm fonksiyonları ve yaptıkları işlemler aşağıdaki listede gösterilmiştir.

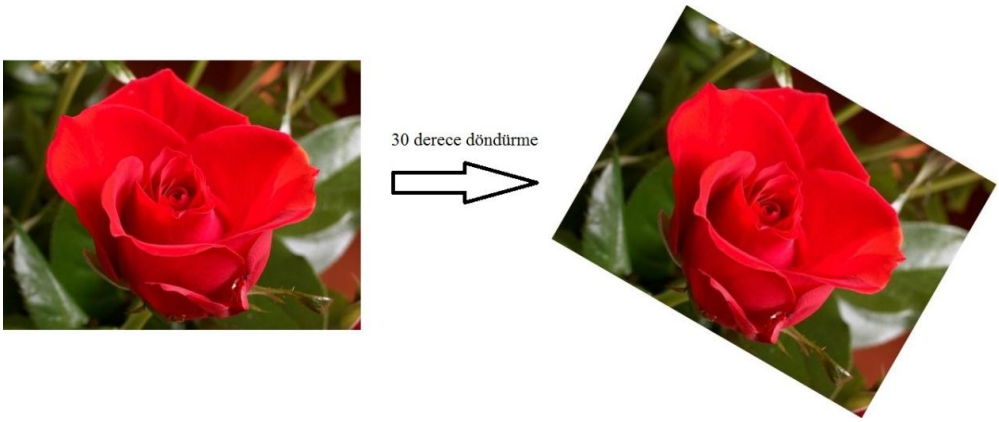
- Perspektif dönüşümü: görüntünün bakış açısını değiştirmeye yarayan fonksiyondur. Amaç her noktanın verilen dönüşüm matrisine göre yeni koordinatlarını hesaplamak, kaynak noktayı, yeni koordinatına yazmaktır. Perspektif dönüşümünde kullanılacak matrisin hesaplanması için 4 adet koordinat gereklidir. Dönüşüm matrisi hesaplandıktan sonra bu matris perspektif dönüşümünün hesaplanmasında kullanılır. Denklem 3.12'de hesaplama gösterilmiştir.

$$hedef\ (x,y) = kaynak\ (\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}) \quad (3.12)$$

- Afin dönüşümü: afin dönüşümü, döndürme, ölçekleme, yer değiştirme fonksiyonlarını bir arada yerine getirebilen, tek başına yapabilen fonksiyondur. Dönüşüm matrisinin elde edilmesi için 3 koordinat gereklidir. Hesaplanan dönüşüm matrisi Denklem 3.13'te gösterildiği şekilde uygulanarak afin dönüşümü yapılır.

$$hedef\ (x,y) = kaynak\ (M_{11}x + M_{12}y + M_{13}, M_{21}x + M_{22}y + M_{23}) \quad (3.13)$$

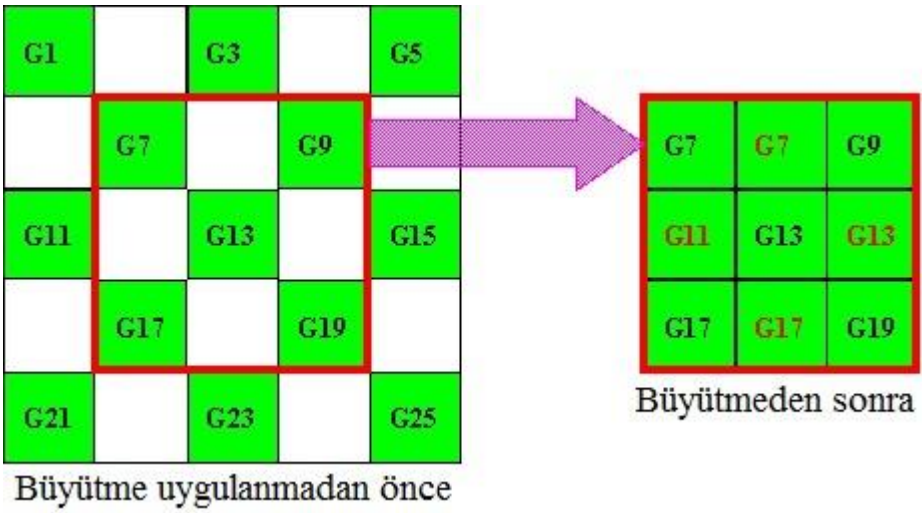
- Aynalama: aynalama fonksiyonu resmin x, y ve z koordinat düzlemlerine göre yansımasını hesaplar.
- Döndürme: bu fonksiyon resmi verilen açı kadar döndürür. Yeni resimde boş kalan noktalar beyaz renk ile doldurulur. Döndürme fonksiyonunun uygulanmış hali Şekil **3.2**'de gösterilmiştir.



Şekil 3.2. Döndürme işleminin uygulaması

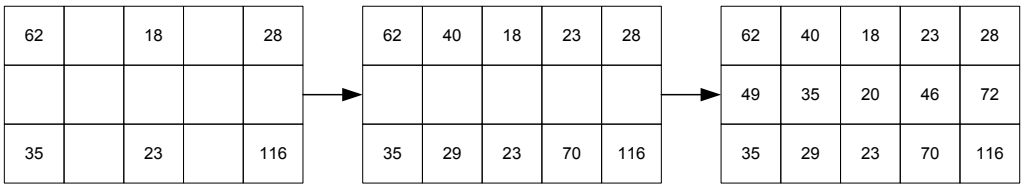
- Yeniden boyutlandırma: resmin boyutunu büyütmek veya küçültmek için kullanılır. Her iki modda da yeni eklenen veya çıkarılan piksellerin nasıl

doldurulacağı ile ilgili en yakın komşu veya çift doğrusal yöntemleri kullanılır. En yakın komşu algoritması en yakındaki komşu pikselin değerinin yeni oluşturulan piksele kopyalanması ile Şekil 3.3'te gösterildiği gibi yapılır.



Şekil 3.3. En yakın komşu yöntemiyle resim boyutunu büyütme

Çift doğrusal yönteminde ise yeni eklenen pikseller orijinal piksellerin ortalaması alınarak hesaplanır ve boyutu büyütülmüş resme Şekil 3.4'teki gibi önce yatay düzlemde sonra düşey düzlemde eklenir.



Şekil 3.4. Çift doğrusal resim büyütme

3.1.4. Renk Dönüşüm Fonksiyonları

Renk dönüşüm fonksiyonları, resimlerin temsil ediliş biçimlerinde yapılan değişiklikler ile ilgilenir. Resimlerin renk temsilleri değiştirilerek, ışıklandırma farklılıkları, renk hassasiyetleri gibi detaylar daha anlamlı hale getirilebilir. Kütüphanede kullanılan renk dönüşüm fonksiyonları aşağıdaki listede sıralanmıştır.

- Gri-tonlama dönüşümü: Gri-tonlamalı renk gösterimi yalnızca tek kanaldan oluşur. Siyah ve beyaz renk tonları arasındaki bir bayt ile gösterilen 256 adet rengi kapsar. Resim gösterimlerinde en sık kullanılan yapı olan kırmızı-yeşil-

mavi yani RGB (Red Green Blue) gösteriminden gri-tonlamaya geçmek için Denklem 3.14'teki formül kullanılır.

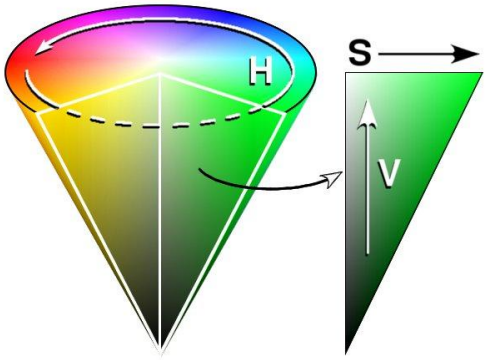
$$Gri - tonlama = 0.299 * kırmızı + 0.587 * yeşil + 0.114 * mavi \quad (3.14)$$

- İkili resim dönüşümü: Gri-tonlamalı resimden veya kırmızı, yeşil, mavi renk tonlarından oluşan RGB renk temsilinden, yalnızca bir bit ile ifade edilen, siyah-beyaz resim oluşturulur. RGB'den dönüşüm yapılırken öncelikle RGB resim gri-tonlamalı resme dönüştürülür. Gri-tonlamalı resimden ikili gösterime dönüştürebilmek için fonksiyon bir parametreye ihtiyaç duyar. Bu parametre değerinin üzerinde kalan değerler beyaz, altında kalan değerler ise siyah olarak belirlenerek resim oluşturulur.
- XYZ renk uzayı: XYZ renk uzayında x, y ve z değerleri üç ana rengin algılanmasını sağlayan sinirlerin beyne yolladıkları uyarıların toplamıdır. x, y, z değerlerinin toplamı bire eşittir. Her bir değerin algılanma oranı, kanal değerinin toplam algılanma oranına bölünmesiyle hesaplanır. RGB renk uzayından XYZ'ye dönüşüm Denklem 3.15'te gösterilmiştir. Tersi olarak XYZ uzayından RGB uzayına dönüşüm de Denklem 3.16'da gösterilmiştir. (XYZ hesaplamalarında RGB değerleri 0 ile 1 arasına sıkıştırılır.)

$$\begin{aligned} X &= 0.412453 * R + 0.35758 * G + 0.180423 * B \\ Y &= 0.212671 * R + 0.71516 * G + 0.0721169 * B \\ Z &= 0.019334 * R + 0.119193 * G + 0.950227 * B \end{aligned} \quad (3.15)$$

$$\begin{aligned} R &= 3.240479 * R - 1.537150 * G - 0.498535 * B \\ G &= -0.969256 * R + 1.875991 * G + 0.041556 * B \\ B &= 0.055648 * R - 0.204043 * G + 1.057311 * B \end{aligned} \quad (3.16)$$

- HSV renk uzayı: hsv renk uzayı sırasıyla renk özü, doygunluk ve parlaklık olarak tanımlanan bir renk uzayı oluşturur. Renk özü rengin baskın dalga boyunu belirler. Doygunluk rengin canlılığını belirler. Yüksek olması durumunda daha canlı renklerden oluşur. Parlaklık ise rengin aydınlığını belirler. HSV renk uzayını oluşturan koni Şekil 3.5'de gösterilmiştir. HSV ve RGB renk uzayı dönüşümleri Ek 1'de gösterilmiştir. (HSV hesaplamalarında RGB değerleri 0 ile 1 arasına sıkıştırılır.)



Şekil 3.5. HSV renk uzayının konik gösterimi.

- YCbCr renk uzayı: Bu renk uzayı JPEG formatında görüntüler oluşturmak için kullanılmaktadır. Y kanalı resmin gri-tonlamalı verisini içerir. Bunların yanında iki boyutlu parlaklık kanalları olan Cr ve Cb kanalları bulunur. RGB ve YCbCr renk uzayları arasındaki dönüşüm formülleri Denklem 3.17 ve Denklem 3.18'de gösterilmiştir.

$$Y = 0.257 * R + 0.504 * G + 0.098 * B + 16$$

$$Cb = -0.148 * R - 0.291 * G + 0.439 * B + 128 \quad (3.17)$$

$$Cr = 0.439 * R - 0.368 * G - 0.071 * B + 128$$

$$R = 1.164 * (Y - 16) + 1.596 * (Cr - 128)$$

$$G = 1.164 * (Y - 16) - 0.392 * (Cb - 128) - 0.813 * (Cr - 128) \quad (3.18)$$

$$B = 1.164 * (Y - 16) + 2.017 * (Cb - 128)$$

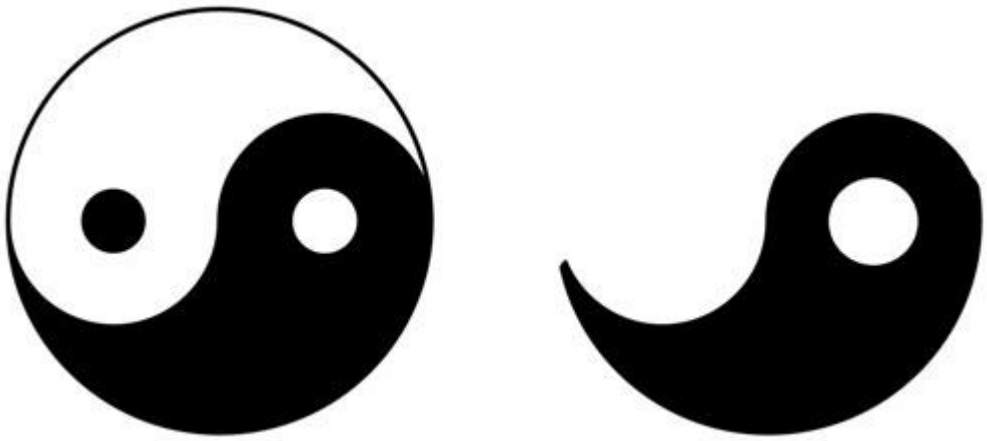
- LUV renk uzayı: LUV renk uzayı 1976 yılında Uluslararası Aydınlatma Komisyonu tarafından kabul edilen bir renk uzayıdır. Renkli ışıklar içeren bilgisayar grafiği işlemlerinde sıkça kullanılır. LUV dönüşümü için öncelikle XYZ dönüşümü yapılması gerekmektedir. RGB değerler 0 ile 1 arasına sıkıştırılmıştır. 8 bitlik işaretsiz değerler kullanıldığı için formüller bu aralığa göre düzenlenmiştir. D65 beyaz nokta değerine göre sabitler ayarlanmıştır. XYZ dönüşümünden sonra LUV uzayına dönüşümün elde edilmesi için Ek 2'deki formüller kullanılır.

3.1.5. Morfoloji Fonksiyonları

Morfoloji, şekilsel yapı ile ilgili fonksiyonları içermektedir. Resim içerisindeki içeriğin şekilsel yapısı ile ilgili ayrıntıları belirginleştirmek, gereksiz ayrıntıları ortadan kaldırmak gibi işlemlerde kullanılır. Morfolojik fonksiyonlar ikili resimler üzerine uygulanmaktadır. Morfolojik işlemlerde, kullanılan yapısal elemanın şekline göre işlem yapılır. Örneğin bir çizgi şeklinde yapısal eleman kullanılması ile artı şeklinde yapısal eleman kullanılması arasında ortaya çıkan veya kaybolan ayrıntılar arasında farklılıklar gözlemlenir. Morfolojik işlemlerde isabet ve ıska kavramları mevcuttur. Yapısal eleman ile resimdeki piksel değerleri birbiri ile örtüşen şekilde ise isabet, yapısal elemandaki değerlerden açıkta kalanlar olursa ıska sonucu açığa çıkar. İsabet olması durumunda, morfolojik işlem uygulanır. Morfolojik işlemler listede açıklanmıştır.

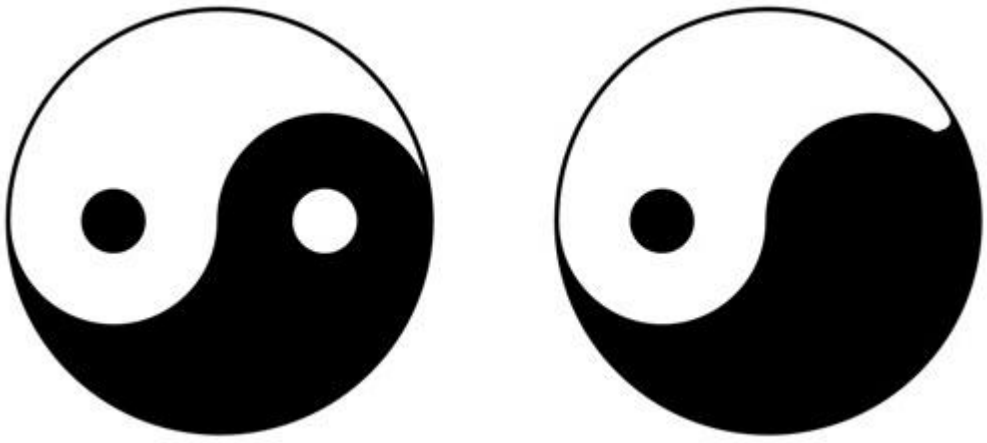
- Genleşme: Resim üzerinde yapısal elemanın herhangi bir elemanına karşılık gelen beyaz pikseller karşılaştırılarak ıska olmaması durumunda, yapısal elemanın merkezindeki nokta beyaza çevrilir. Herhangi bir noktanın beyaz olması kontrol edildiği için beyaz noktaların sayısında artış olur, beyaz

alanlar büyüyerek genleşir. Orijinal resim ve genleşme işlemi uygulanmış resim Şekil 3.6'te gösterilmiştir.



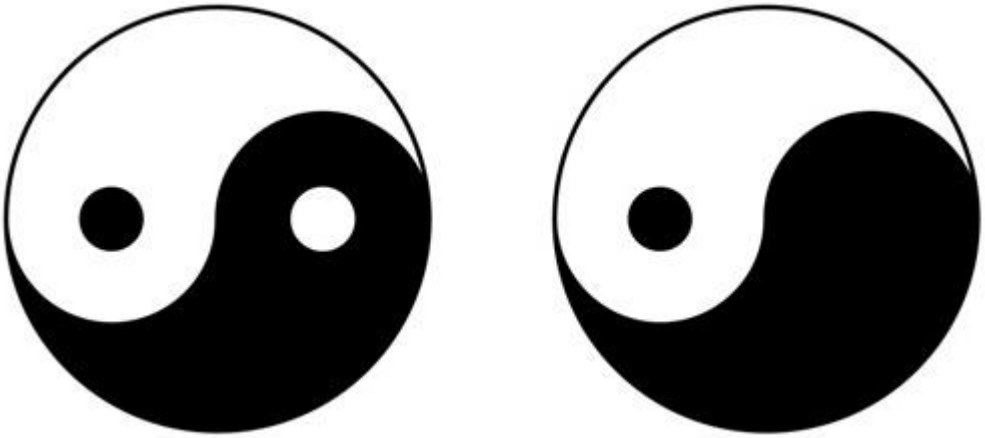
Şekil 3.6. Genleşme işlemi uygulanmış resim.

- Aşınma: Resim üzerinde yapısal elemanın bütün elemanlarına karşılık gelen beyaz pikseller karşılaştırılarak isabet olması durumunda, yapısal elemanın merkezindeki nokta beyaza çevrilir. Eğer bir ıska bile olursa, merkezdeki nokta siyaha çevrilir. Bütün noktaların kontrolü söz konusu olduğu için, beyaz noktaların sayısında azalma, yani aşınma gerçekleşir. Orijinal resim ve aşınma işlemi uygulanmış resim Şekil 3.7'te verilmiştir.



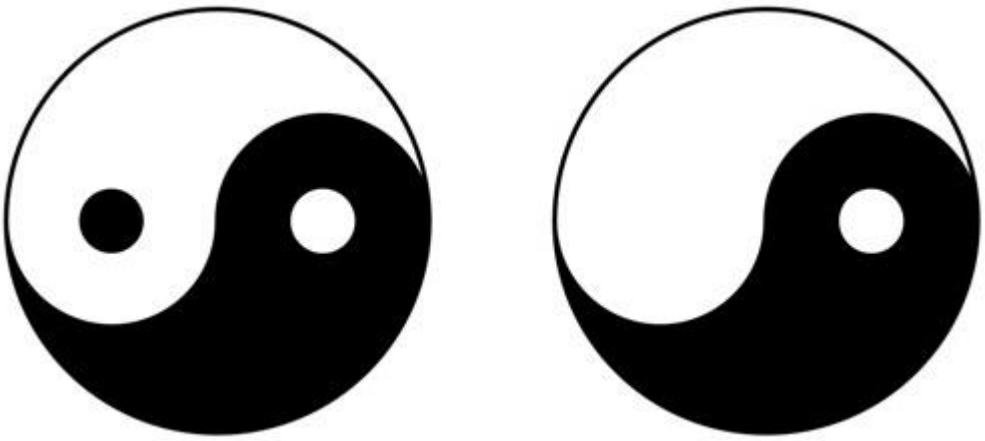
Şekil 3.7. Aşınma işlemi uygulanmış resim.

- Açılma: Resim üzerine aynı yapısal eleman ile önce aşınma sonra genleşme işlemi uygulanması sonucu elde edilir. Bu işlem birkaç aşınma sonucu aynı sayıda genleşme ile de kullanılabilmektedir. Açılma işlemi uygulanmış resim Şekil 3.8'te gösterilmiştir.



Şekil 3.8. Açılma işlemi gerçekleştirilmiş resim.

- Kapanma: Resim üzerine aynı yapısal eleman ile önce genleşme sonra aşınma işlemi uygulanması sonucu elde edilir. Bu işlem birkaç genleşme sonucu aynı sayıda aşınma ile de kullanılabilmektedir. Kapanma işlemi uygulanmış resim Şekil 3.9'te gösterilmiştir.



Şekil 3.9. Kapanma işlemi uygulanmış resim.

3.1.6. İçerik Tespit Fonksiyonları

İçerik tespiti resim içerisindeki faydalı bilgileri çıkarmak için kullanılır. Kenar, köşe tespiti, geometrik şekillerin tespiti bu bölümde yer alır. Filtreleme fonksiyonlarında kullanılan bazı fonksiyonlar aslında içerik tespiti için de kullanılabilmektedir.

- Doğrusal çizgi ve halka tespiti: Doğrusal çizgilerin ve halkaların tespiti Hough dönüşümü ile mümkündür. Algoritmanın temelinde, oylama mantığı ile çalışmaktadır. İkili resimler üzerinde kenarların belirlenmesi işleminden sonra kullanılarak çizgilerin doğrusal veya halka oldukları tespit edilebilir.
- Kenar tespiti: kenar tespiti için sık kullanılan Canny kenar tespiti algoritması içerik tespit fonksiyonları arasındadır.
- Köşe tespiti: köşe noktaların tespiti için Harris köşe tespiti algoritması kullanılmıştır.

3.2. KÜTÜPHANE FONKSİYONLARININ GERÇEKLENMESİ VE KULLANILAN ENİYİLEME YÖNTEMLERİ

Kütüphanenin içerdiği fonksiyonların gerçeklenmesi masaüstü ortamında yapılmıştır, ancak kütüphanenin çalışması gömülü platform üzerinde olacağı için eniyileme bu platforma göre yapılmıştır. Kullanılan platform Freescale firmasının ürettiği i.mx6q platformudur. Kartın üzerinde dört çekirdekli Arm Cortex A9 işlemci ve Vivante GC2000 grafik işlemcisi vardır. Platformun özellikleri Çizelge 3.2'de gösterildiği gibidir.

Fonksiyonların gerçeklemesi yapılırken, üzerinde çalışacağı grafik işlemcinin yapısı göz önünde bulundurulmuş ve eniyileme bu özel grafik işlemcisine göre yapılmıştır. Gerçekleme yapılırken grafik işlemcinin artıları ve eksileri göz önünde bulundurulmuştur. Grafik işlemcisinin artılarından bahsedecek olursak, CPU ve GPU'nun aynı yonga üzerinde bulunması sayesinde, tek belleğe erişim sağlarlar. Bu durum da masaüstü veya dizüstü platformlarda ortaya çıkan, verilerin veriyolu üzerinden aktarımı gecikmesini ortadan kaldırır. Masaüstü grafik işlemciler ile kıyaslandığında başka bir artısı bulunmamaktadır. Öte yandan gömülü CPU ile

karşılaştırıldığında artıları daha fazladır. Aynı sayıda çekirdek içermesine karşın, grafik işlemcisi her çekirdeğin içerisindeki işlem birimlerinin daha fazla olması sebebiyle aynı anda 16 iş-parçacığı çalıştırabilmektedir. Bu iş parçacıkları donanım düzeyinde aynı anda çalıştırılabildikleri için fazladan ek yük bindirmez.

Çizelge 3.2. Gömülü platformun özellikleri

Grafik işlemcisi	Vivante GC2000
Grafik işlemcisi saat sıklığı	500 Mhz
Grafik işlemcisi bellek saat sıklığı	1066 Mhz
Grafik işlemcisi çekirdek sayısı	4
İşlemci	Arm Cortex A9
İşlemci çekirdek sayısı	4
İşlemci saat sıklığı	1 Ghz
İşlemci bellek saat sıklığı	1066 Mhz

Grafik işlemcisinin içerdiği donanım çekirdekleri yapı olarak tek buyrukta çok veri (SIMD) işleme kabiliyetine sahiptir. Bu kabiliyet sayesinde bir iş parçacığı da aynı anda birden çok veri kelimesine (word) erişip bunlar üzerinden işlem yapabilir. GC2000 grafik işlemcisi 2, 3, 4 ve 8 kelimeyi aynı anda işleme kapasitesini desteklemektedir. OpenCL spesifikasyonunda 16 kelime uzunluğuna kadar SIMD desteği bulunmasına rağmen, GC2000 grafik işlemcisi 16 kelimelik veri vektörlerini 2 adet sekizlik vektör haline dönüştürüp işlediği için başarımı daha düşük olur.

Bu açılardan gömülü grafik işlemcisinin artıları bulunmasına rağmen, az güç tüketmesi, küçük alan kaplaması gereksinimlerini yerine getirebilmesi için, gömülü grafik işlemcisinin birtakım başarım ölçütlerinden de feragat etmesi gerekmiştir. Vivante GC2000 grafik işlemcisi üzerinde, OpenCL tarafından erişilebilen yerel belleğin boyutu 1 KB olup, yazmaçlar ile gerçeklenmiştir. Aynı anda yazmaç kullanımı fazla olan ve yerel bellek kullanmaya çalışan bir program çekirdeği, yazmaç taşması adı verilen olay ile karşılaşarak başarımdan kayıp verir. Halihazırda yazmaç sayısı az olan grafik işlemcisi, yazmaçlardan arta kalan kısmı yerel bellek olarak kullanmaya çalışmaktadır. Eğer yazmaç sayısını aşan miktarda yerel bellek

kullanılmaya çalışılırsa, veriler evrensel bellekte saklanır. Evrensel bellek yonga dışında olduğu için erişmek uzun zaman alır.

GC2000 işlemcisinin başka bir sorunu da desen belleği (texture memory) bulunmamasıdır. Grafik işleri için ayrılmış desen belleği genel amaçlı işlemlerde kullanılamadığı için ve desen işlemcileri de genel amaçlı kısımdan erişilemediği için, desen işlemcilerinin çok hızlı yapabildiği yeniden boyutlandırma, döndürme gibi işlemler grafik işlemcisinin özelleşmemiş birimlerinde gerçeklenmiştir. Bu kısıt yalnızca daha hızlı yapılabilecek fonksiyonların az hızlandırılabilmesi ile sonuçlanır.

Bellek ile ilgili başarım darboğazı ise bellek bant genişliğinden kaynaklanmaktadır. Masaüstü grafik işlemcileri aynı anda 256 bayt büyüklüğüne kadar veriyi tek seferde okuyup yazabilirken, GC2000'de bu miktar 32 bayttır. Bu yüzden bellek erişimi sırasında birleştirme (memory coalescing) yapılsa bile en fazla okunabilecek miktar 32 bayttır. 4 baytlık kelimeler üzerinden düşünüldüğünde en fazla 8 kelime okunabilmektedir.

GC2000 grafik işlemcisinin diğer bir sorunu da buyruk kısıtlamasıdır. 512 buyruktan daha fazlası çalıştırılamadığı için çok karmaşık gerçeklemeler yapılamamakta veya birkaç çekirdek (kernel) kullanılarak yapılması gerekmektedir. Çok sayıda çekirdek çalışması sırasında çekirdekler aynı anda komut-kuyruğuna atılırsa, donanım üzerinde çalışma sırasında aralarında geçiş yapılacağından ek yük sebebiyle başarım düşer. Farklı zamanda çalışması durumunda ise, tüm iş-kalemlerinin bitip bitmediği bir senkronizasyon noktası yardımı ile kontrol edilipiş bittikten sonra komut kuyruğuna çekirdeğin gönderilmesi gerekir ve bu da ayrı bir beklemeye, yani başarım kaybına yol açar.

Kütüphanenin gerçekleştirilmesi ve eniyilemesi sırasında gömülü platformun ve gömülü işlemcilerin artılarından yararlanılmış, eksiler de göz önünde bulundurularak başarıma etkileri en aza indirgenmeye çalışılmıştır. Tüm kütüphane düşünüldüğünde uygulanabilecek eniyileme çeşidi iki tanedir. Birincisi, tüm fonksiyonlara uygulanabilecek genel stratejiler, ikincisi ise fonksiyondan fonksiyona değişen özelleşmiş stratejileridir.

Genel stratejiler, OpenCL'in yazım ve çalışma şekli sebebiyle başarım kayıplarını azaltmaya yönelik yapılan işlemlerdir. GC2000 grafik işlemcisi aynı anda donanımsal olarak 16 adet iş-kalemi çalıştırma kapasitesine sahip olduğu için, üzerinde işlem yapılacak veriler 16'lık bölütlere ayrılarak işlenir. Daha az bölütler kullanıldığında boşta kalan, çalışmayan donanımlar olacağı için başarım da düşer.

Bellek işlemleri, grafik işlemciler üzerinde yapılan işlemlerden daha uzun sürmektedir. Başarımı arttırmak için bellekten okunan veriler, birleştirmeli okuma yöntemi kullanılarak, aynı anda işlenebilecek veri boyutu maksimize edilmiştir. Bu sayede daha az bellek erişimi amaçlanmıştır.

Grafik işlemcilerde dallanma tahmini mekanizmaları bulunmamaktadır. Donanımsal olarak çok yer tutan, karmaşıklığı arttıran ve aslında grafik işlemcilerde gerekli olmayan dallanma tahmini, grafik işlemciler genel amaçlı kullanılmaya başlandığı zaman grafik işlemciler üzerinde bir sorun haline gelmişlerdir. Dallanma öngörüsü olmadığı için, bir dallanma koşuluna rastlandığında grafik işlemciler bu dallanmanın her iki ihtimalini de işler. İşlemler bittiği zaman dallanmanın sonucunda yapılmaması gereken işin ürettiği sonuçlar yazılmaz, çöpe atılır. Bu sebeple bir dallanma olduğunda grafik işlemcisinin üzerinde yapılan işlem artar ancak yapılan anlamlı iş ortalaması azalır. Kütüphane gerçeklenirken bu durum göz önünde bulundurulmuş ve olabildiğince az dallanma kullanılmaya özen gösterilmiştir.

Gömülü işlemcilerin artısı olan belleğin işlemci ve grafik işlemcisi arasında paylaşımlı kullanılması büyük fayda sağlamaktadır. Masaüstü ortamlarda CPU üzerinde bulunan verinin, grafik işlemcisi üzerine aktarılması gerektiğinden bu kısımda ek yük oluşmaktadır. Paylaşımlı bellekte ise aktarıma gerek yoktur. Grafik işlemcisinin kullanacağı veriyi gösteren işaretçi (pointer), zaten halihazırda CPU'nun kullandığı işaretçi ile aynı yeri göstermektedir. Bu aktarımın yapılmaması ile başarım artışı sağlanmaktadır.

Bazı işlemlerin hesaplamasının CPU üzerinde yapılarak sabit bellek alanına yazılması ve grafik işlemcisinde kullanılacağı zaman buradan erişilmesi yöntemi de başarımı arttıran bir yöntemdir. Ayrıca yavaş olan matematiksel işlemler yerine, aynı hesaplama yapılan ve daha hızlı olan matematiksel işlemler de kütüphane

gerçeklenirken kullanılmıştır. Örnek vermek gerekirse, bir sayıyı 5'e bölmek yerine 0.2 ile çarpmak, çarpım işleminin bölmeye göre donanımsal olarak daha hızlı olması sebebiyle başarımda artış getirmektedir.

OpenCL çekirdek kodunun çalışma zamanında derlenerek çalıştırılmasının yerine, önceden derlenmiş ve ikili uzantılı dosya haline getirilmiş, çalıştırılabilir çekirdek kodları kullanılarak derleme zamanı sıfırlanmıştır.

Matris ve resim verileri üzerinde çalışıldığı için işlenen veriler 2 boyutludur. Bu yüzden boyut uzayı ve iş-kalemlerinin iş-grupları içerisindeki dağılımları her zaman 2 boyutlu olarak ayarlanmış ve bu şekilde kullanılarak başarım artışı sağlanmıştır.

Özelleşmiş stratejiler ise her kütüphane modülü için, hatta her fonksiyon için ayrı olarak düşünülmüş eniyileme yöntemleridir. Özelleşmiş stratejilerin açıklaması aşağıdaki listededir.

- Matris İşlemleri: matris işlemlerini hızlandırmak için kullanılan yöntemlerin başında OpenCL'de tanımlanmış olan vektör değişkenleri gelir. OpenCL destekli vektör boyutlarının 2,3,4,8 ve 16 olduğundan bahsedilmişti. GC2000 grafik işlemcisinin içerdiği SIMD işlemcisinin boyutu ancak 4 boyutundaki vektörlere yettiği için en hızlı bu boyuttaki vektörlerde çalışmaktadır. Bu sebeple matris işlemleri hızlandırılırken float4, int4 ve uchar4 veri tipleri kullanılarak başarım arttırılmıştır. Logaritma ve üssel fonksiyonun hesaplanmasında, grafik işlemcisinin üzerindeki donanımlardan yararlanılmıştır. Bir matrisin transpozunu alma işlemi, yoğun bellek işlemi gerektirmektedir. Bu sebeple transpoz fonksiyonu gerçekleştirilirken, küçük boyutlarda grafik işlemcisi kullanılmamıştır. Bir verinin bir transpozdan sonraki yerini hesaplamak için kullanılan hesaplamalar ancak çok büyük boyutlu matrislerde grafik işlemcisi üzerinde hesaplandığı zaman başarım artışı sağlamaktadır.
- Filtreleme: filtreleme fonksiyonlarında eniyileme için birkaç farklı OpenCL çekirdeği oluşturulmuştur. Farklı boyutlardaki filtreler için sabit boyutlu filtreler kullanılarak hızlandırma sağlanmıştır. Örneğin bulanıklaştırma filtresi için 3x3, 5x5 7x7 filtreleri için ayrı çekirdekler kullanılmış bu

boyuttan büyük filtreler için jenerik filtre kullanılarak küçük boyutlu filtrelerin çalışması eniyilenmiştir.

- Geometrik Dönüşüm: bu fonksiyonlarda kullanılan trigonometrik işlemler için, özelleşmiş donanımlardan yararlanılmıştır. Sin ve Cos fonksiyonlarının hesaplanması grafik işlemcisinin üzerindeki donanım sayesinde gerçekleşir.
- Renk Dönüşümleri: bu fonksiyonlarda işlemler hızlandırılırken vektör değişkenleri kullanılmıştır. Dönüşümlerde kullanılan sabit değerler, sabit bellek kullanılarak hem yazmaç sayısından feragat edilmemiş, hem de önbelleklendiği için erişim hızından kayıp verilmemiştir. Sabit değerler için bölme işlemleri yerine eşdeğer çarpma işlemleri kullanılmıştır.
- Morfoloji: isabet ve ıska değerlerine göre işlem yapıldığı için morfolojik işlemler çok sayıda dallanma içermektedir. Dallanmaları azaltabilmek algoritma üzerinde yapılabilecek bir iyileştirme bulunamamıştır. Yalnızca genel eniyileme yöntemleri kullanılmıştır.
- İçerik Tespiti: bu fonksiyonların gerçeklenmesinde de genel eniyileme yöntemlerinden başka eniyileme yöntemi kullanılmamıştır.

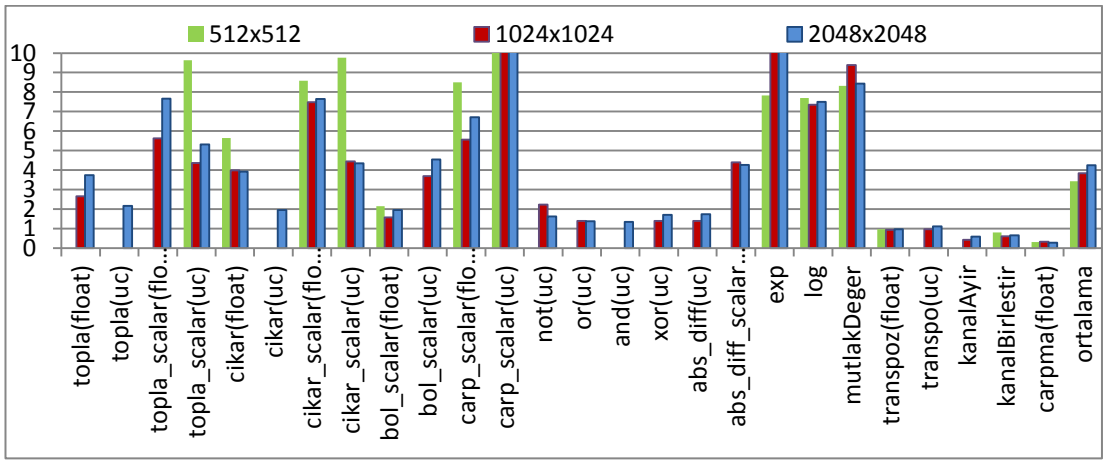
3.3. SONUÇLAR

Kütüphanenin başarımı zamanlama ile ölçülmüştür ve OpenCV ile karşılaştırma yapılmıştır. Başarım sonuçları hem gömülü platform üzerinde hem de bazı fonksiyonlar için masaüstü ortamda alınmıştır. Masaüstü ortamda sonuçların karşılaştırılmasının sebebi, bazı fonksiyonlarda kütüphanenin yüksek başarım elde edememesinin sebebinin kod bazlı mı yoksa gömülü platform bazlı mı olduğunu tespit etmektir.

Matris işlemlerinde farklı matris boyutları üzerinde sonuçlar alınmıştır. 256x256 boyutundaki matris işlemlerinde yalnızca logaritma fonksiyonu 7 kat hızlanma sağlamıştır. Bunun dışındaki fonksiyonlarda herhangi bir hızlanma söz konusu değildir. Ancak 256x256'dan büyük matrislerde hızlanmalar görülmektedir.

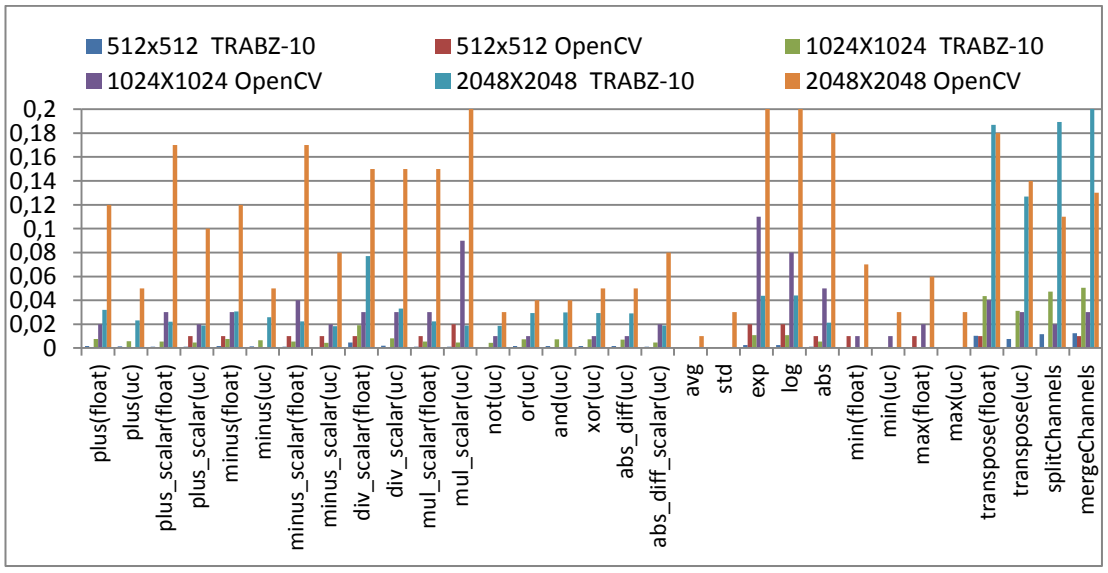
512x512, 1024x1024 ve 2048x2048 boyutlarındaki matrisler üzerinde yapılan testlerde skalar değerle çarpma işleminin 18 kat hızlandığı gözlenmiştir. Bu değere en yakın olan üs alma fonksiyonu 10 kata yakın hızlanma sağlamıştır. Bazı

fonksiyonlarda hızlanma hızlanma sağlanmazken, ortalamada 512x512 için 3.4 kat, 1024x1024 için 3.8 kat ve 2048x2048 için 4.2 kat hızlanma sağlanmıştır. Alınan sonuçlar Şekil 3.10'da gösterilmiştir.



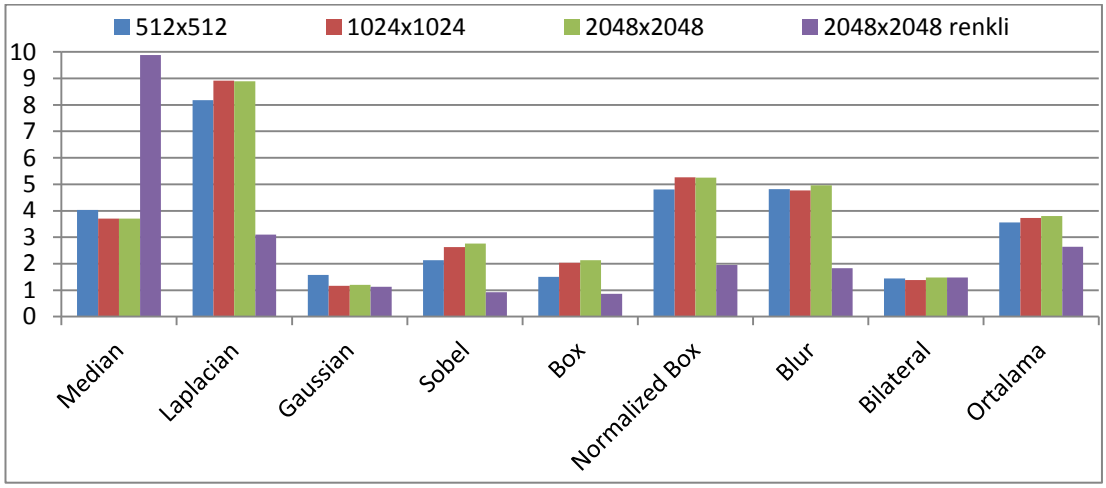
Şekil 3.10. Matris işlemleri hızlanma miktarları

Hızlanma miktarlarına bakıldığında OpenCL ile gerçeklenmiş kütüphanenin başarımı arttırdığı gözlenmektedir. Gömülü sistemler genellikle gerçek zamanlı işlerde kullanıldığı için, işlemlerin hızlanma miktarlarının yanında toplam işlem süresi de önem arz etmektedir. Şekil 3.11'de görüldüğü gibi, OpenCV kullanılarak yapılan işlemler 0.2 saniye sürerken, OpenCL kütüphanesi kullanılarak gerçekleştirilen işlemlerin çoğu 10 fps (saniye başına çerçeve) ve üzerine ulaşabilmektedir.



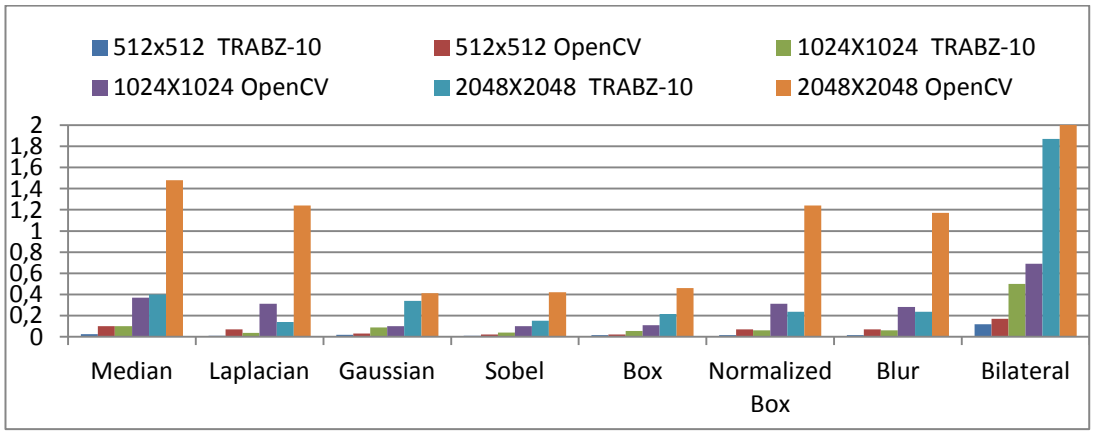
Şekil 3.11. Matris işlemleri çalışma süreleri

Filtre fonksiyonlarına gelecek olursak, gri-tonlamalı ve yalnızca en büyük seçilen boyut olan 2048x2048 boyutu için ölçülen sonuçlar Şekil 3.12'te gösterildiği gibidir.



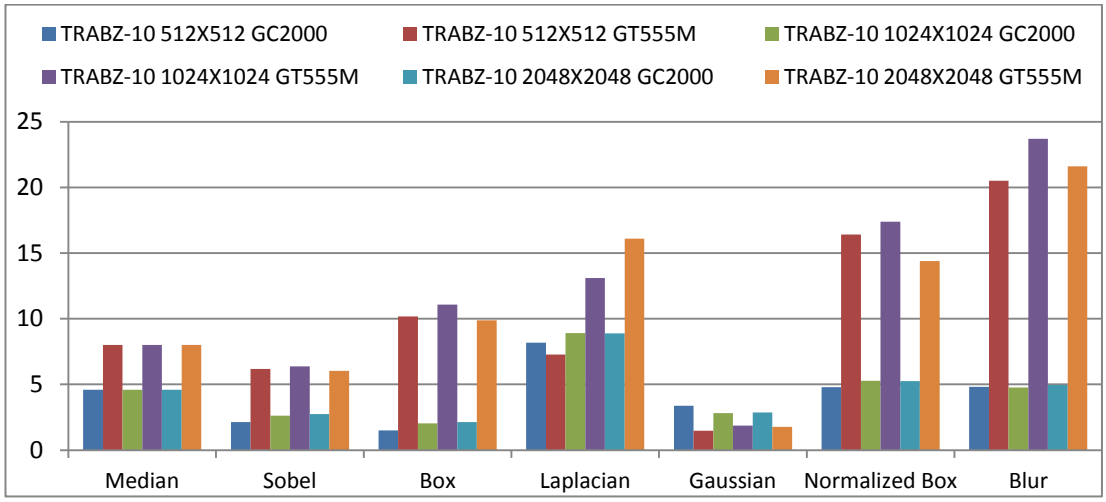
Şekil 3.12. Filtre fonksiyonları hızlanma miktarları

Ortanca (Median) filtresinin renkli resimde çok hızlı çalışmasının sebebi, algoritmasının paralelleştirmeye uygun olmasıdır. Gauss ve çift taraflı (Bilateral) filtre en az hızlanma gösteren filtrelerdir. Bunun sebebi, iki filtrede de birden fazla OpenCL çekirdeğinin çalışmasıdır. Bir çekirdek bitip diğerine geçilirken karşılaşılan ek yük, hızlanmanın önüne geçmektedir. Bu tespitten yola çıkarak, aynı çekirdek içerisine daha fazla işlem yüklemek, çok sayıda küçük işler yapan çekirdek çalıştırmaktan daha avantajlıdır. Bu tespit kütüphanenin üzerinde koştuğu platform için geçerlidir ancak her platform için aynı şeyi söyleyemeyiz.



Şekil 3.13. Filtre fonksiyonlarının çalışma süreleri

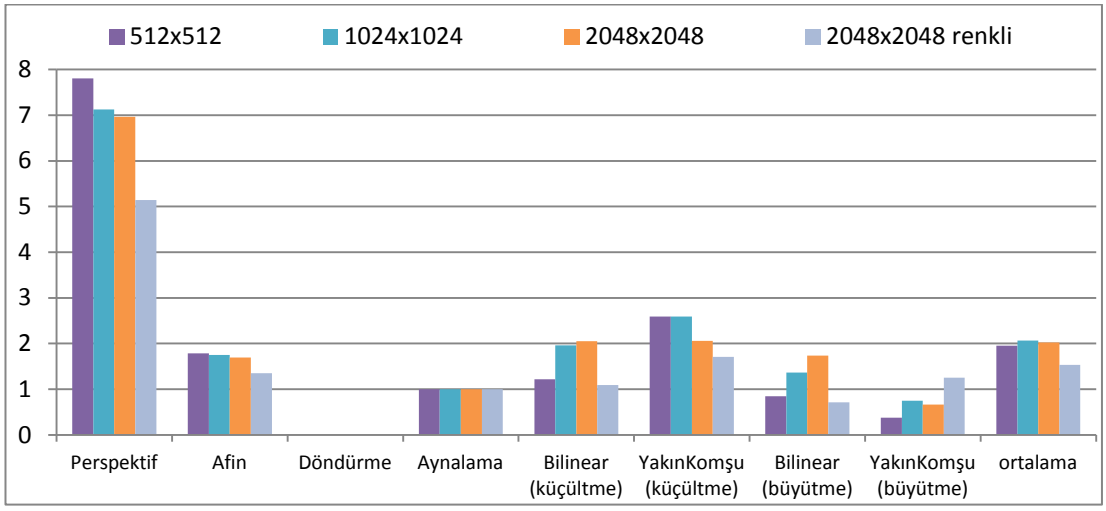
Matris işlemlerinde yakalanan hız, filtreleme fonksiyonlarında yakalanamamaktadır. Filtreleme fonksiyonları OpenCV'ye göre ortalama olarak 3.5 - 4 kat daha hızlı çalışmaktadır. OpenCV başarımlarını incelediğimizde yalnızca 512x512 gibi küçük boyutlu resimlerde gerçek zamanlı işlem hızına erişilebilirken, OpenCL kütüphanesinin 512x512'de ve 1024x1024 boyutlu resimlerde neredeyse tüm fonksiyonların 20 fps değerine yakınsadığı gözlenmiştir. 2048x2048 boyutlu resimlerde 5 fps'ye kadar düşüş Şekil 3.13'da gösterilmiştir.



Şekil 3.14. Gömülü grafik işlemcisi ve masaüstü grafik işlemcisinin OpenCV karşısında ulaştığı hızlanma miktarları

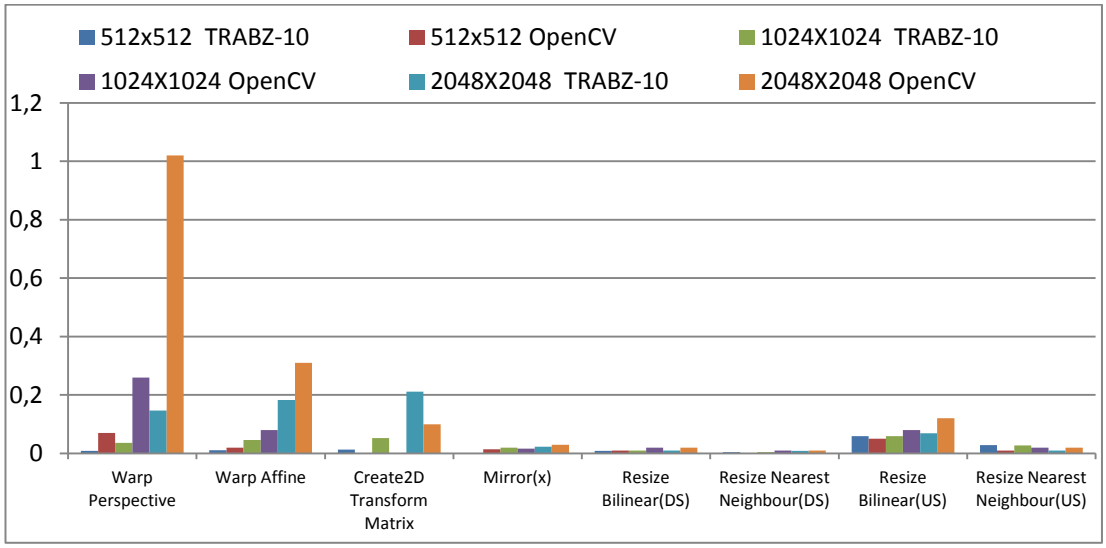
Gömülü grafik işlemcisi için yazılan OpenCL kütüphanesi, aynı zamanda masaüstü platformda da denenmiştir. Bu denemenin sonucunda, masaüstü platformda da hızlanma sağlandığı gözlenmiştir. Masaüstü platformda ve gömülü platformda kütüphanenin OpenCV'ye karşı ulaştığı hızlanma miktarları Şekil 3.14'de gösterilmiştir. Gömülü platformda ortalama 3.5 - 4 kata kadar hızlanma sağlayan kütüphane, masaüstü ortamda ortalama 10 kat hızlanma sağlamıştır.

Geometrik dönüşüm fonksiyonlarının hızlanma miktarları Şekil 3.15'de gösterilmiştir. Grafikte görüldüğü üzere, döndürme fonksiyonu OpenCV'de çok hızlı yapıldığı için hızlanma değil yavaşlama ile sonuçlanmıştır. Ortalamada 1.5 - 2 kat hızlanma sağlayan kütüphane, aynalama fonksiyonunda herhangi bir hızlanma veya yavaşlama sağlamamıştır.



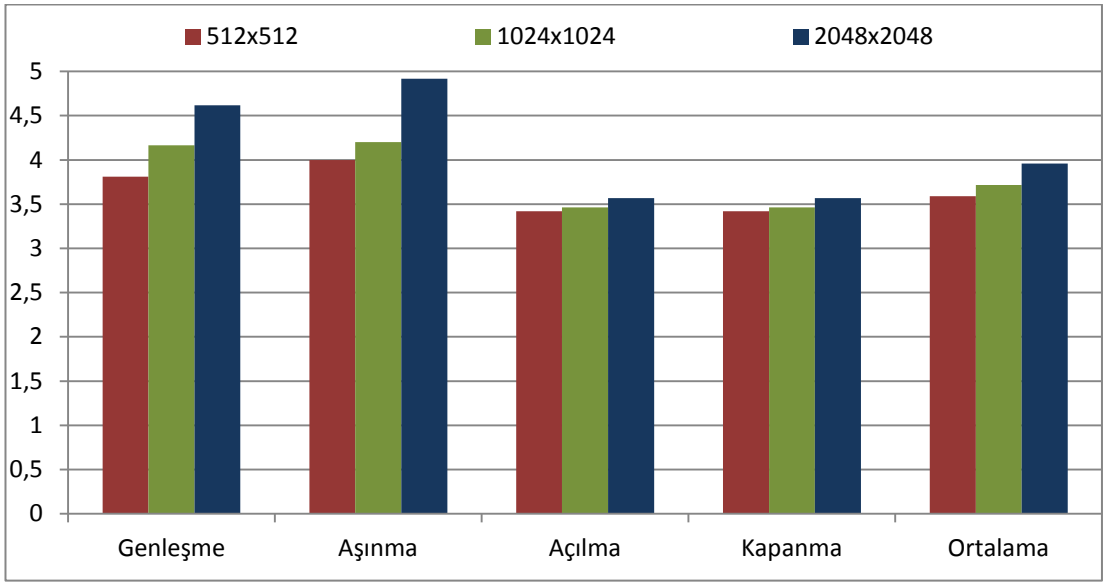
Şekil 3.15. Geometrik dönüşüm fonksiyonlarının hızlanma miktarı

Fonksiyonların çalışma sürelerine bakıldığında, hızlanmanın çok büyük miktarda olmamasının sebebinin, aslında OpenCV'de de geometrik dönüşüm fonksiyonlarının çok hızlı çalışması olduğu görülmektedir. Bütün boyutlarda gerçek zamanlı işlem yapılabilecek çalışma süreleri Şekil 3.16'te gösterilmiştir.



Şekil 3.16. Geometrik dönüşüm fonksiyonlarının çalışma süreleri

Morfoloji işlemlerini incelediğimizde, ortalama hızlanmanın 4 kata yaklaştığını görmekteyiz. Renkli resimler üzerinde değil, yalnızca siyah-beyaz resimler üzerinde işlem yapıldığından renkli resim sonuçlara dahil edilmemiştir. Şekil 3.17'te hızlanma miktarları gösterilmiştir.



Şekil 3.17. Morfolojik işlemlerin hızlanma miktarları

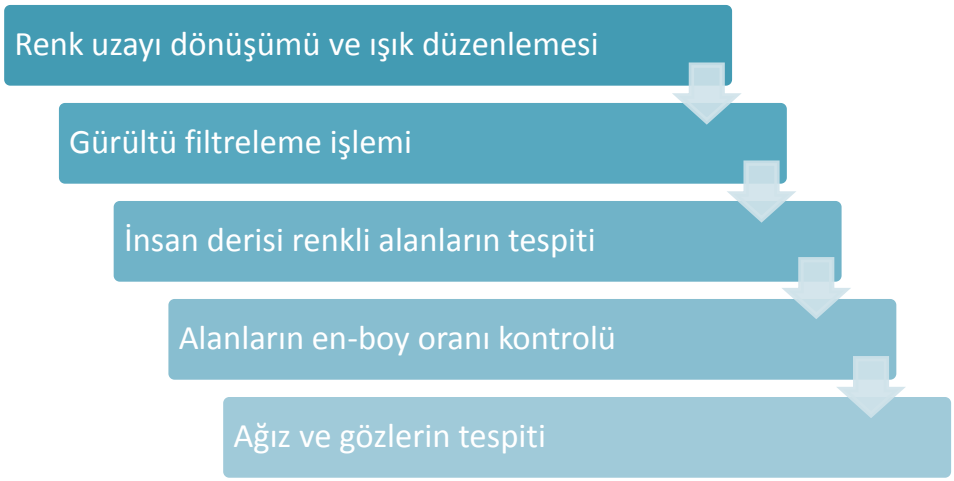
Genel olarak kütüphanenin, eşdeğer fonksiyonların çoğunda OpenCV'ye göre daha başarılı olduğu gözlenmiştir. Kütüphaneyi kullanılabilirlik açısından denemek için geliştirilen örnek insan yüzü tespit etme uygulamasının gerçeklenmesi ve sonuçları Bölüm 4'te anlatılmıştır.

4. ÖRNEK UYGULAMA: YÜZ TESPİT ETME

Kütüphane fonksiyonlarının gerçeklenmesi bittikten sonra, gerçek anlamda bir uygulamanın nasıl davrandığını gözlemek ve kullanımı hakkında bilgi edinmek için, kütüphane kullanılarak çok karmaşık olmayan bir insan yüzü tespit etme algoritması gerçeklenmiştir. Algoritma hem OpenCV'de hem kütüphanede gerçeklenerek, karşılaştırma yine ikisi arasında yapılmıştır. Gerçeklenen yüz tespit etme algoritması için [4] çalışmasından yararlanılmıştır.

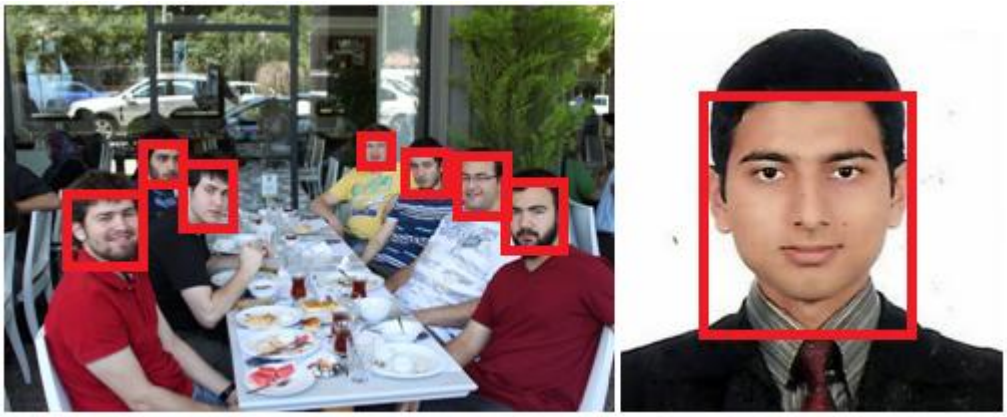
4.1. ALGORİTMA

Geliştirilen uygulamanın algoritması renk ve içerik tabanlı tespit mekanizması ile çalışmaktadır. Farklı ışık koşulları altında insan yüzünü yüksek başarımla ve yüksek hızla tespit edebilmektedir. Algoritmanın ilk adımı YCbCr renk uzayına geçiş yapmaktır. Bu renk uzayındaki kroma (Cr) içeriği insan derisinin rengini tespit etmekte kullanılmaktadır. Öncelikle resimdeki gürültüleri temizlemek için 5x5'lik bir alçak geçiren filtre kullanılmıştır. Filtrelemeden sonra insan derisinin rengine uygun alanlar belirlenmiş ve köşe noktaları bulunmuştur. Bu alanlardan belli bir en-boy oranına uyumsuz olanlar elenmiştir. Daha sonra renk açısından biraz daha koyu olan gözler belirlenmiştir. Gözü bulmak için dikey histogram ile aydınlatması daha koyu alanların varlığı aranır. Bu aşamaların hepsini geçen bir alan bulunduğunda insan yüzü olarak işaretlenir. Algoritmanın adımları Şekil 4.1'de gösterilmiştir.



Şekil 4.1. Yüz tespit etme algoritması

Bu algoritmanın başarı oranı %95 olarak ölçülmüştür. İnsan yüzünün, deri rengine benzer alanların yanında durması durumları dikkate alındığında başarı oranı düşerek %90 olmaktadır. Bu uygulamanın asıl amacı algoritmanın iyileştirilmesi değil, çalışma zamanı açısından başarım sonuçlarıdır. Şekil 4.2'de algoritmanın çalışması sonucu elde edilen resimler bulunmaktadır.

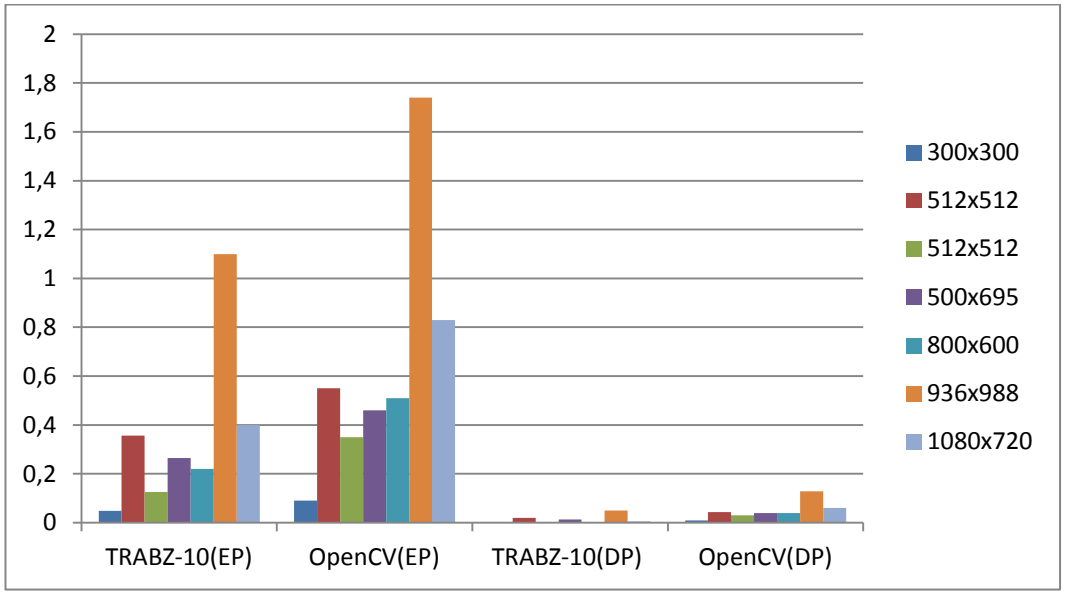


Şekil 4.2. Algoritmanın çalışması sonucu işaretlenen insan yüzleri

4.2. BAŞARIM SONUÇLARI

Gerçeklemesi yapılan algoritmanın başarım sonuçları hem gömülü platformda hem de masaüstü platformda alınmıştır. Her iki platformda da kütüphanenin başarımı OpenCV'den daha yüksek olmuştur. Farklı resim boyutlarında denenen algoritmanın çalışma süreleri Şekil 4.3'te verilmiştir. Bu şekilde EP, gömülü platformu, DP ise masaüstü platformu göstermektedir.

Aynı algoritmanın OpenCV kütüphanesinde ve geliştirilen OpenCL kütüphanesindeki sonuçlara göre, kütüphane OpenCV'den hem masaüstü ortamda hem de gömülü ortamda daha hızlı sonuç vermektedir.



Şekil 4.3. Yüz tespit algoritmasının gömülü ve masaüstü platformlarda çalışma süreleri

Bu çalışmada, OpenCL kullanılarak geliştirilen bir görüntü işleme kütüphanesi fonksiyonları, geliştirme yöntemleri, eniyileme yöntemleri ve örnek uygulama sonuçları ile birlikte anlatılmıştır. Anlık işlem ihtiyacı gün geçtikçe artmaktadır. Gömülü platformlarda grafik işlemcilerin işlem kapasitesine ne kadar katkı yapabileceği bu çalışmada gösterilmiştir.

5. BENZER ÇALIŞMALAR

Genel amaçlı grafik işlemciler üzerinde en sık gerçeklenen uygulamalar, görüntü işleme uygulamalarıdır. Görüntü işleme operasyonları aynı işlemi çok sayıda piksel üzerinde işlediği için SIMD ve SIMT mimari altyapısına çok uygundur. Bu yüzdendir ki son yıllarda GPGPU kullanılarak görüntü işleme fonksiyonlarını ve algoritmalarını hızlandırmak çok revaçta bir konudur. İlk uygulamalar grafik işlemcilerinin programlanabilir altyapısına gölgelendirici kodları ile erişerek işlemleri hızlandırmayı amaçlamışlardır [5] [6]. GLSL [7] grafik işlemciler üzerindeki gölgelendirici donanımları (shader) programlamaya yarayan (programmable shader) yapıdır. Bölüm 1'de anlatıldığı üzere, grafik işlemcileri genel amaçlı programlanabilir hale getiren yapı gölgelendiricidir. Grafik işlemciler üzerinde genel amaçlı işlemler yapma ile ilgili inceleme [8] çalışmasında anlatılmıştır.

Grafik işlemciler üzerinde görüntü işleme ile alakalı en kapsamlı çalışmaları GPUCV [9] ve MinGPU [10] kütüphaneleri oluşturmaktadır. OpenVIDIA [11] kütüphanesi de yalnızca NVidia grafik işlemciler üzerinde çalışan bir kütüphanedir. GPUCV yalnızca GLSL kullanılarak yazılmış bir kütüphane olup, OpenCV fonksiyonlarının aynısını grafik işlemciler üzerinde çalıştırmaktadır.

Öte yandan MinGPU kütüphanesi, hem GLSL hem de GPGPU programlama dilleri olan OpenCL [12] ve CUDA [13] kullanmıştır. Buradaki amaç, işlemler yapılırken hangi dilin daha hızlı işlem gerçekleştirdiği önceden ölçümlenir. Uygulama 2. kez çalıştırıldığında bu ölçümlemelere dayanarak fonksiyonlar çağrılır. GLSL'in hızlı olduğu yerde gölgelendirici kodu kullanılırken, OpenCL veya CUDA'nın hızlı olduğu yerde gerekli çekirdekler kullanılır.

OpenVIDIA kütüphanesi ise yalnızca NVidia grafik işlemciler üzerinde çalışmak üzere optimize edilmiştir. GPGPU kavramı çıktıktan sonra CUDA ile geliştirmeler de yapılmış, son zamanlarda OpenCL ile yazılmış fonksiyonlar da eklenmiştir.

Grafik işlemcilerin genel amaçlı kullanılmaya başlamasıyla birlikte, CUDA kullanılarak görüntü işleme fonksiyonlarını hızlandırma çalışmaları başlamıştır. Bazı çalışmalar sıfırdan görüntü işleme fonksiyonlarını gerçeklemişlerdir. [14] çalışmasında histogram eşitleme fonksiyonu gerçeklenmiş ve küçük resimlerde ortalama 20 kat, büyük resimlerde ise 40 kata varan hızlanmalar elde edilmiştir. Bir diğer çalışma olan [15]'da canny kenar tespit algoritması hızlandırılmaya çalışılmıştır.

Bunların yanı sıra bazı çalışmalar da halihazırda kullanılan kütüphanelerin fonksiyonlarını hızlandırmayı amaç edinmiştir. [16] çalışmasında MATLAB yazılımının görüntü işleme fonksiyonlarına ek olarak, OpenCL kullanılarak

paralelleştirilmiş fonksiyonlar eklenmiştir. Hatta aynı çalışmada CUDA ile OpenCL gerçeklemeleri karşılaştırılmıştır.

Gömülü platformlarda yapılan araştırmalara gelecek olursak, bu çalışma gömülü platform grafik işlemcilerinde OpenCL kullanan ilk çalışmadır. Gömülü platformlarda yapılan önceki çalışmalar OpenGL ES 2.0 [17] kullanılarak gerçeklenmiştir. Bu da GPGPU değil, GLSL kullanılarak grafik işlemcisi üzerinde yapılmış bir çalışmadır.

FPGA platformları son zamanlarda OpenCL desteğine başlamıştır. Altera firmasının Stratix adlı FPGA yongası, OpenCL kodu yazıldığında, bu kodu çalıştırabilecek bir donanım üretmekte ve bu donanım üzerinde yazılan kodu çalıştırmaktadır. FPGA kullanılarak yapılan çalışmalara [18] örnek olarak verilebilir. Bu çalışmada bir video sıkıştırma algoritması CPU, GPU ve FPGA'de gerçeklenmiş ve başarımları karşılaştırılmıştır. FPGA başarım sonuçlarının GPU sonuçlarından 3 kata kadar daha hızlı olduğu gözlenmiştir.

KAYNAKLAR

[1] J. Kriger and R. Westermann, "Linear algebra operators for GPU implementation of numerical algorithms," *ACM Transactions on Graphics (TOG),* pp. 908-916, 2003.

[2] D. Luebke ve G. Humprheys, «How gpus work,» *IEEE Computer,* cilt 40, no. 2, pp. 96-100, 2007.

[3] C. Tomasi ve R. Manduchi, «Bilateral filtering for gray and color images,» %1 içinde *IEEE Sixth International Conference on Computer Vision*, 1998.

[4] Y.-T. Pai, S.-J. Ruan, M.-C. Shie ve Y.-C. Liu, «A Simple and Accurate Color Face Detection Algorithm in Complex Background,» %1 içinde *IEEE International Conference on Multimedia and Expo*, Toronto, 2006.

[5] R. J. Rost, OpenGL Shading Language, Addison-Wesley Professionnal, 2009.

[6] R. Marroquim ve A. Maximo, «Introduction to GPU Programming,» %1 içinde *Tutorials of the XXII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI TUTORIALS),* Pisa, 2009.

[7] «OpenGL Shading Language,» Khronos Group, [Çevrimiçi]. Available: http://www.opengl.org/documentation/glsl/. [%1 tarihinde erişilmiştirMarch 2013].

[8] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn ve T. J. Purcell, «A Survey of General-Purpose Computation on Graphics Hardware,» *Computer Graphics Forum,* cilt 26, no. 1, pp. 80-113, 2007.

[9] Y. Alluse, P. Horain, A. Agarwal ve C. Saipriyadarshan, «GpuCV: A GPU-accelerated framework for,» %1 içinde *IEEE International Conference on Multimedia and Expo*, Toronto, 2006.

[10] P. Babenko ve M. Shah, «MinGPU: a minimum GPU library for computer vision,» *Journal of Real-Time Image Processing*, cilt 3, no. 4, pp. 255-268, 2008.

[11] J. Fung ve S. Mann, «OpenVIDIA: parallel GPU computer vision,» %1 içinde *Proceedings of the 13th annual ACM international conference on Multimedia*, Singapore, 2005.

[12] A. Munshi, B. R. Gaster, T. G. Mattson, J. Fung ve D. Ginsburg, OpenCL programming guide, Pearson Education, 2011.

[13] NVidia, «C. U. D. A., Compute unified device architecture programming guide,» NVidia, 2007.

[14] Z. Yang, Y. Zhu ve Y. Pu, «Parallel image processing based on cuda,» %1 içinde *International Conference on Computer Science and Software Engineering*, 2008.

[15] Y. Luo ve R. Duraiswami, «Canny edge detection on NVIDIA CUDA,» %1 içinde *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2008.

[16] J. Kong, M. Dimitrov, Y. Yang, J. Liyanage, L. Cao, J. Staples, M. Mantor ve H. Zhou, «Accelerating MATLAB image processing toolbox functions on GPUs,» %1 içinde *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, 2010.

[17] A. Munshi, D. Ginsburg ve D. Shreiner, OpenGL ES 2.0 Programming Guide, Addison-Wesley, 2008.

[18] D. Chen ve D. Singh, «Fractal video compression in OpenCL: An evaluation of CPUs, GPUs, and FPGAs as acceleration platforms,» %1 içinde *18th Asia and South Pacific Design Automation Conference*, 2013.

EKLER

Ek 1: RGB'den HSV'ye ve HSV'den RGB'ye dönüşüm formülleri.

RGB'den HSV'ye	HSV'den RGB'ye

Parlaklık:
V = max (R, G, B)
Doygunluk:
temp = min (R, G, B)
S = (V-temp)/V
Renk özü:
Cr = (V - R) / (V - temp)
Cg = (V - G) / (V - temp)
Cb = (V - B) / (V - temp)
if (R==V)
H = Cb - Cg
if (G==V)
H = 2 + Cr - Cb
if (B==V)
H = 4 + Cg - Cr
H = H * 60
if (H < 0)
H = H + 360
	if (S==0)
	R = G = B = V
	else
	{
	if (H == 360)
	H = 0
	else
	H = H/60
	I = floor(H)
	F = H - I
	M = V * (1 - S)
	N = V * (1 - S * F)
	K = V * (1 - s * (1 - F))
	if (I == 0)
	R = V
	G = K
	B = M
	if (I == 1)
	R = N
	G = V
	B = M
	if (I == 2)
	R = M
	G = V
	B = K
	if (I == 3)
	R = M
	G = N
	B = V
	if (I == 4)
	R = K
	G = M
	B = V
	if (I == 5)
	R = V
	G = M
	B = V
	}

Ek 2: RGB'den LUV'a ve LUV'dan RGB'ye dönüşüm formülleri.

XYZ'den LUV'a	LUV'dan XYZ'ye
xn = 0.312713 yn = 0.329016 Yn = 1.0 un = 4*xn / (-2*xn + 12*yn + 3) vn = 9*yn / (-2*xn + 12*yn + 3) u = 4*X / (X + 15*Y + 3*Z) v = 9*Y / (X + 15*Y + 3*Z) L = 116 * (Y/Yn)^(1/3) - 16 U = 13*L*(u-un) V = 13*L*(v-vn) L = L * 255 / 100 U = (U + 134) * 255 / 354 V = (V + 140) * 255 / 256	L = L * 100 / 255 U = (U * 354 / 255) - 134 V = (V * 256 / 255) - 140 xn = 0.312713 yn = 0.329016 Yn = 1.0 un = 4*xn / (-2*xn + 12*yn + 3) vn = 9*yn / (-2*xn + 12*yn + 3) u = U / (14 * L) + un v = V / (13 * L) + vn Y = Yn * ((L + 16) / 116) ^ 3 X = (9 / 4) * Y * u / v Z = (9 * Y - 15 * v * Y - v * X) / (3 * v)

(XYZ - RGB arası dönüşümlerin formülleri Denklem 3.15 ve 3.16'da verilmiştir.)

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, Adı	: ŞİMŞEK, Osman Seçkin
Uyruğu	: T.C.
Doğum Tarihi ve Yeri	: 25.11.1989
Medeni Hali	: Evli
Telefon	: 0 (537) 5862423
E-Posta	: ossimsek@etu.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Y. Lisans	TOBB ETÜ Bilgisayar Mühendisliği	2014
Lisans	TOBB ETÜ Bilgisayar Mühendisliği	2011

İş Deneyimi

Yıl	Yer	Görev
2011 – 2014	TOBB ETÜ	Burslu Yüksek Lisans Öğrencisi
2014 –	Milsoft Yazılım Teknolojileri A.Ş.	Yazılım Mühendisi

Yabancı Dil

İngilizce (İleri seviye)

Almanca (Başlangıç)

Yayınlar

Cavus, M., Sumerkan, H.D., Simsek, O.S., Hassan H., Yaglikci, A.G., & Ergin O. (2014, January). GPU based Parallel Image Processing Library for Embedded Systems. VISAPP 2014 9th International Conference on Computer Vision Theory and Applications. VISIGRAPP. Lizbon, Portekiz.

Aykenar, M. B., Ozgur, M., Simsek, O. S., & Ergin, O. (2013, October). Adapting the columns of storage components for lower static energy dissipation. In Very Large Scale Integration (VLSI-SoC), 2013 IFIP/IEEE 21st International Conference on (pp. 222-227). IEEE. İstanbul, Türkiye.

Koc, F., Simsek, O. S., & Ergin, O. (2011, October). Using content-aware bitcells to reduce static energy dissipation. In Computer Design (ICCD), 2011 IEEE 29th International Conference on (pp. 51-56). IEEE. Boston, Massachusetts.