POLİTEKNİK DERGİSİ

JOURNAL of POLYTECHNIC

ISSN: 1302-0900 (PRINT), ISSN: 2147-9429 (ONLINE)
URL: http://dergipark.org.tr/politeknik

GPU programlamada CUDA platformu kullanılan paralel görüntü işleme çalışmalarının incelenmesi

A survey on parallel image processing studies using CUDA platform in GPU programming

***Yazar(lar) (Author(s))**: Semra AYDIN[1], Refik SAMET[2], Ömer Faruk BAY[3]*

ORCID[1]: 0000-0002-1670-9677

ORCID[2]: 0000-0001-8720-6834

ORCID[3]: 0000 0002 6823 145X

Bu makaleye şu şekilde atıfta bulunabilirsiniz(To cite to this article): Aydın S., Samet R. ve Bay Ö.F., "GPU programlamada CUDA platformu kullanılan paralel görüntü işleme çalışmalarının incelenmesi", ***Politeknik Dergisi***, 23(3): 737-754, (2020).

Erişim linki (To link to this article): http://dergipark.org.tr/politeknik/archive

DOI: **10.2339/politeknik.563767**

GPU Programlamada CUDA Platformu Kullanılan Paralel Görüntü İşleme Çalışmalarının İncelenmesi

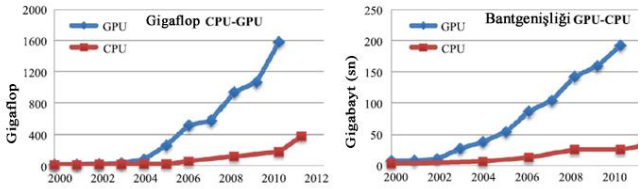
A Survey on Parallel Image Processing Studies Using CUDA Platform in GPU Programming

Önemli noktalar (Highlights)

- *GPU'da görüntü işlemenin paralelleştirilmesi /Parallelization of image processing in GPU*
- *CUDA ile GPU'nun programlanması / GPU programming with CUDA*
- *CUDA kullanılan görüntü işleme uygulamaları / Image processing applications using CUDA*
- *Paralel görüntü işleme çalışmaları / Studies on parallel image processing*

Grafik Özet (GraphicalAbstract)

GPU'nun performans ve bant genişliği CPU'ya göre üstel olarak artmaktadır. Bu çalışmada, GPU ve CUDA teknolojilerinin görüntü geriçatma, iyileştirme, bölütleme, çakıştırma ve sınıflandırma uygulamalarında kullanımı incelenmiş, değerlendirilmiş, avantajları ve dikkat edilmesi gereken hususlar belirlenmiştir. / The performance and bandwidth of GPU increases exponentially compared to the CPU. In this study, the use of GPU and CUDA technologies in image reconstruction, enhancement, segmentation, registration and classification applications are examined, evaluated, their advantages and issues to be considered are determined.



(a) (b)

Şekil. CPU ve GPU'nun performans (a) ve bant genişliği (b) açısından karşılaştırması /
Figure.CPU and GPU comparison of (a) performance and (b) bandwidth)

Amaç (Aim)

Bu çalışmanın temel amacı araştırmacılara ve konuya yeni başlayanlara görüntü işleme uygulamalarında GPU ve CUDA gibi donanım ve yazılım teknolojilerinin kullanımı konusunda bir başvuru kaynağı sağlamaktır./ The major purpose of this survey is to provide a comprehensive reference source for the starters or researchers involved in use of CUDA platform in GPU programming for image processing techniques.

Tasarım ve Yöntem (Design &Methodology)

Çalışmada GPU ve CUDA kullanan görüntü işleme çalışmaları beş bölümde incelenmiş ve değerlendirilmiştir./ Studies using CUDA platform in GPU programming have been classified and evaluated under 5 areas.

Özgünlük (Originality)

Makale, CUDA platformu kullanılarak gerçekleştirilen görüntü işleme çalışmalarının incelendiği, özetlendiği, analizi yapıldığı, avantaj ve dezavantajlarının değerlendirildiği Türkçe yapılan ilk çalışmadır./ The article is the first study in Turkish to examine, summarize, analyze and evaluate the advantages and disadvantages of image processing studies using the CUDA platform.

Bulgular (Findings)

GPU ve CUDA teknolojileri ile paralel görüntü işleme gerçekleştirmek için bir sistemli yaklaşım önerilmiştir./ A systematic approach is proposed to perform parallel image processing with GPU and CUDA technologies.

Sonuç (Conclusion)

GPU ve CUDA teknolojilerinin kullanıldığı görüntü işleme uygulamalarında dikkat edilmesi gereken hususlar belirlenmiştir. Çalışma GPU ve CUDA teknolojilerinin verimli kullanılmasına katkı sağlamaktadır./ Issues to be considered in image processing applications using GPU and CUDA technologies have been determined.The study contributes to the efficient use of GPU and CUDA technologies.

Etik Standartların Beyanı(*Declaration of EthicalStandards*)

Çalışmada kullanılan materyal ve yöntemler etik kurul izni gerektirmemektedir./ There is no need for ethical committee permission for the materials and methods used in this study.

GPU Programlamada CUDA Platformu Kullanılan Paralel Görüntü İşleme Çalışmalarının İncelenmesi

Derleme Makalesi / Review Article

Semra AYDIN[1*], Refik SAMET[2], Ömer Faruk BAY[3]

[1]Bilişim Enstitüsü, Gazi Üniversitesi, Türkiye
[2]Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Ankara Üniversitesi, Türkiye
[3]Teknoloji Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü, Gazi Üniversitesi, Türkiye

(Geliş/Received : 25.12.2018 ; Kabul/Accepted : 09.07.2019)

ÖZ

Görüntü işleme pek çok alanda kullanılmaktadır. Görüntü işleme teknikleri gün geçtikçe görüntülerin çözünürlüklerinin artmasıyla daha fazla işlemci gücüne ihtiyaç duymaktadır. Görüntü işleme sürecini hızlandırmak için paralel görüntü işleme teknikleri kullanılmaktadır. GPU programlama günümüzde çok kullanılan ve tercih edilen paralel görüntü işleme tekniklerinden biridir. CUDA ise GPU programlamada en çok kullanılan platformdur. Bu çalışmanın temel amacı araştırmacılara ve konuya yeni başlayanlara görüntü işleme uygulamalarında GPU ve CUDA gibi donanım ve yazılım teknolojilerinin kullanımı konusunda bir başvuru kaynağı sağlamaktır. Bu amaç kapsamında çalışmada GPU ve CUDA kullanılarak yapılan görüntü işleme çalışmaları incelenmiş ve değerlendirilmiştir. GPU ve CUDA kullanan görüntü işleme çalışmaları, görüntü geriçatma, görüntü iyileştirme, görüntü bölütleme, görüntü çakıştırma ve görüntü sınıflandırma olmak üzere beş bölümde incelenmiş ve değerlendirilmiştir. Elde edilen sonuçlar doğrultusunda, GPU ve CUDA kullanımının avantajları ve bu teknolojilerin kullanıldığı görüntü işleme uygulamalarında dikkat edilmesi gereken hususlar belirlenmiştir.

Anahtar Kelimeler: Görüntü işleme, paralel programlama, GPU, CUDA.

A Survey on Parallel Image Processing Studies Using CUDA Platform in GPU Programming

ABSTRACT

Image processing is used in a variety of fields. Image processing techniques need high processor performance due to increased image resolution day by day. Parallel processing techniques are used to satisfy the requirements related to high performance in real time image processing applications. Recently, GPU programming is one of the most commonly used and preferred methods in parallel processing. CUDA is the most popular platform in GPU programming. In this survey the studies where CUDA platform was used for image processing are presented and evaluated. The major purpose of this survey is to provide a comprehensive reference source for the starters or researchers involved in use of CUDA platform in GPU programming for image processing techniques. Studies using CUDA platform in GPU programming have been classified under 5 areas; image reconstruction, image enhancement, image segmentation, image registration and image classification. Advantages of using CUDA in GPU programming for image processing and issues to pay attention in applications have also been underlined.

Keywords: Image processing, parallel computing, GPU, CUDA.

1. GİRİŞ (INTRODUCTION)

Görüntü, piksel adı verilen ve her birinin özel bir konumu ve değeri olan sonlu sayıdaki elemanlardan oluşmaktadır. Görüntü işleme, girişteki bir görüntüyü (fotoğraf veya bir video karesi), başka bir görüntüye dönüştürmek veya görüntüye ait özellikleri çıkarmak amacıyla yapılan işlemler serisidir. Görüntü işleme güvenlik sistemleri, kontrol sistemleri, üretim, tıp, uzay bilimleri, savunma sanayi, vb. gibi pek çok alanda uygulanmaktadır. Gelişen teknolojiyle birlikte elde edilen görüntülerin çözünürlükleri artmaktadır. Dolaysıyla işlenecek veri miktarı da giderek artmakta ve büyük miktardaki verileri işlemek için daha fazla işlem gücüne ihtiyaç duyulmaktadır. Uygulamaların amacına uygun olarak değişik görüntü işleme teknikleri kullanılmaktadır. Örneğin, nesne tanıma, hareket algılama, sınıflandırma, hata bulma, vb. gibi pek çok işlem için uygun görüntü işleme teknikleri kullanılmaktadır. Bu işlemlerden herhangi biri için görüntü işleme tekniklerinde kullanılan algoritmalardan bir ya da birkaçının çalıştırılması gerekmektedir. Görüntü işlemede kullanılan algoritmalar genellikle her bir piksel için aynı işlem basamaklarının uygulanması şeklinde çalışmaktadır. Her bir piksel için birbirinden bağımsız olarak işlem yapılmaktadır. Görüntü işleme algoritmalarından birçoğu bu nedenle doğası gereği bir paralellik içermektedir.

Paralel programlama, bir programın işlem zamanını kısaltmak amacıyla işlemin birden fazla birime dağıtılarak çalıştırılmasıdır. Bu birimler farklı bilgisayarlar olabildiği gibi aynı bilgisayar üzerindeki işlemciler de olabilirler. Özellikle son yıllarda işlemciler

**Sorumlu Yazar (Corresponding Author)*
e-posta : semra.avar@gmail.com

ve grafik işlem birimleri (GPU-Graphics Processing Unit) üzerindeki çekirdek sayılarının artması tek bilgisayar üzerinde çalışan paralel programlama platformlarının gelişmesini sağlamıştır.

Grafik işlem birimleri, genel amaçlı GPU programlama (GPGPU-General Purpose Computing on GPU) ismiyle yaklaşık 10 yıldır kullanılmaktadır. Ancak son dönemlerde kullanımında bir artış gözlemlenmektedir. Merkezi işlem birimi (CPU-Central Processing Unit) güçlü ama az sayıda çekirdek içerirken GPU çok sayıda çekirdek içermektedir. GPU'ların donanımsal özelliklerinden dolayı her bir çekirdek diğerleri ile paralel olarak kendisine ait bir veriyi işleyebilmektedir. Son dönemlerde GPU'ların gücü giderek artmış ve GPU üreticileri genel amaçlı programlama için GPU teknolojilerini geliştirmişlerdir. Bu durum GPU'nun genel amaçlı programlamada kullanımının artmasını sağlamıştır. Farklı üreticiler tarafından değişik özelliklerde GPU kartları piyasaya sürülmüştür. Intel, AMD ve NVIDIA en büyük GPU kart üreticileridir. Intel en büyük CPU üreticisi olmasına rağmen sadece düşük performanslı GPU pazarına hakimdir. Yüksek performans pazarını ise AMD ve NVIDIA paylaşmaktadırlar. Akademik ve endüstriyel uygulamalarda ise NVIDIA, AMD'ye göre daha yaygın olarak kullanılmaktadır [1].

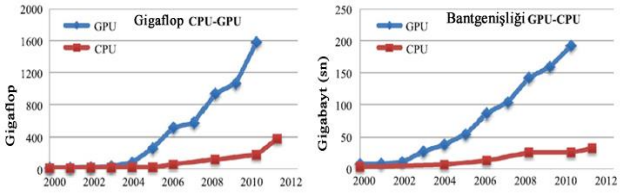
Genel amaçlı GPU programlama, özellikle farklı veriler üzerinde aynı işlemlerin yapıldığı uygulamalarda performans açısından çok iyi sonuçlar vermektedir. Görüntü işleme de bu alanlardan biridir. CUDA (Compute Unified Device Architecture) genel amaçlı GPU programlamada en çok kullanılan platformlardan biridir. Bu çalışmada CUDA platformu üzerinde gerçekleştirilen görüntü işleme çalışmaları incelenmekte ve değerlendirilmektedir.

Bu çalışma beş bölümden meydana gelmektedir. İkinci bölümde GPU ve CUDA konusunda temel bilgiler verilmektedir. Üçüncü bölümde GPU programlama ve CUDA platformunun kullanıldığı ilk uygulamalar incelenmektedir. Dördüncü bölümde görüntü işleme alanında CUDA ile ilgili yapılan çalışmalar sunulmaktadır. Görüntü işleme konusu görüntü geriçatma, görüntü iyileştirme, görüntü bölütleme, görüntü çakıştırma ve görüntü sınıflandırma olmak üzere beş bölümde ele alınarak yapılan çalışmalardan çıkarılan bulgular ve yorumlar dördüncü bölümde ifade edilmektedir. Son bölümde yapılan çalışmanın genel değerlendirilmesi yapılmakta ve görüntü işleme alanında CUDA kullanımı ile ilgili önerilerde bulunulmaktadır.

2. GPU VE CUDA TEKNOLOJİLERİ (GPU PROGRAMMING AND CUDA)

Bu bölümde önce GPU teknolojisi ile ilgili genel bilgi verilecek ve daha sonra bu teknolojiyi programlamak için kullanılan CUDA platformundan detaylı bir şekilde bahsedilecektir.

CPU'lar seri hesaplamalarda, giriş çıkış işlemlerinde iyi bir performansa sahiptirler. GPU'lar ise büyük ölçekli paralel hesaplamalar için uygun bir mimariye sahiptirler. Son yıllarda GPU'nun bant genişliği ve performansı CPU'ya göre çok hızlı artış göstermiştir (Şekil 1). Bu performans farkı iki işlemci arasındaki fiziksel çekirdek sınırlamaları ve mimari farkından kaynaklanmaktadır. CPU seri olarak çalışır ve seri operasyonların işlem hızını optimize etmeye çalışır. CPU'da geleneksel olarak performans artışı frekansın yükseltilmesiyle elde edilir. Frekans arttıkça hız da artar ve uzun yıllar performans artışı bu şekilde sağlanmıştır. 2000'li yıllardan sonra fiziksel sınırlardan dolayı frekans artışı istenildiği kadar gerçekleştirilememiştir. Frekans artışı fiziksel sınırlamalar nedeniyle pratikte yaklaşık 4.0 GHz'in üzerine çıkamamaktadır. İşlemci teknolojisinde seri hesaplamada en yüksek performansa ulaşılmakta olup performans artışı sağlamak için çoklu çekirdek ve vektör komutları kullanılmaktadır [1].



(a) (b)

Şekil 1. CPU ve GPU'nun performans (a) ve bant genişliği (b) açısından karşılaştırması [1] (CPU and GPU comprasion of (a) performance and (b) bandwidth)

İlk kullanılmaya başladığı yıllardan itibaren GPU'ların sağladığı performans üstel olarak artmaya devam etmektedir. Bu artış paralel uygulamaların performansının da artmasını sağlamaktadır. Bu nedenle GPU'lar büyük ölçekli paralel uygulamalarda performansı arttırmak için çok kullanışlıdırlar.

2.1 GPU Programlama (GPU Programming)

GPU, yoğun matematiksel grafik hesaplamaları yapmak üzere tasarlanmış ekran kartlarının işlemcisi için kullanılan bir isimdir. GPU'lar, CPU'dan daha verimli ve performanslı bir biçimde ekrana görüntü verirler. GPU'lar görüntüleri işleme ve görüntülemede son derece verimli olup yüksek paralel yapıları sayesinde karmaşık algoritmaların işlenmesinde CPU'lardan daha etkindirler.

Genel amaçlı GPU programlamada üç platform kullanılmaktadır:

- CUDA
- OpenCL (Open Computing Language)
- DirectCompute

CUDA, NVIDIA firması tarafından 2006 yılında piyasaya sürülen ve GPU'nun programlaması için kullanılan bir platformdur. Linux, Windows ve Mac OSX üzerinde çalışabilen hem düşük hem de yüksek seviyeli bir yazılım geliştirme arayüzü sunmaktadır. NVIDIA, CUDA platformunu destekleyen sırasıyla Fermi, Kepler,

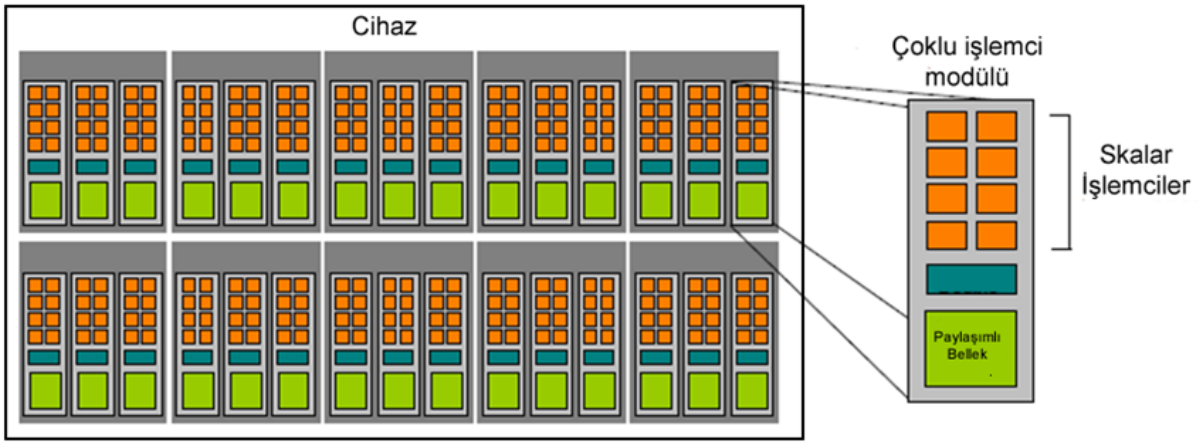
Maxwell ve Pascal mimarilerini geliştirmiştir. OpenCL, Apple tarafından 2008 yılında piyasaya sürülmüş olup Intel, AMD ve NVIDIA başta olmak üzere pek çok ekran kartı üreticisi tarafından desteklenmektedir. DirectCompute Windows'un desteklediği GPU kartlarının programlanmasında kullanılmakta olup son sürümü DirectX 12'dir [2]. CUDA, ekran kartlarını programlamak için kullanılan en yaygın platformdur. Performans açısından bakıldığında CUDA, diğer platformlara göre pek çok uygulamada daha hızlı çalışmaktadır [3]. Paralel işlemcilerin hedefinin de yüksek hız olduğu düşünüldüğünde CUDA platformu pek çok kullanıcı tarafından tercih edilmektedir.

2.2. CUDA

CUDA, GPU gücünü kullanarak bilgisayarın işlem performansına yüksek oranda katkı yapan bir paralel programlama platformudur. C, C++, C#, Fortran, Java, Python gibi programlama dilleri ile yazılmış algoritmaların GPU üzerinde çalışmasını sağlayan bir sistem olarak da tanımlanabilir. En yaygın olarak kullanılan diller C ve C++'dır. Windows, Linux ve Mac OSX işletim sistemleri tarafından desteklenmektedir.

2.2.1. **CUDA mimarisi** (CUDA architecture)

GPU programlamada yönetici (host) CPU'yu, cihaz (device) ise GPU'yu ifade etmektedir. Şekil 2'de cihaz mimarisi gösterilmektedir.



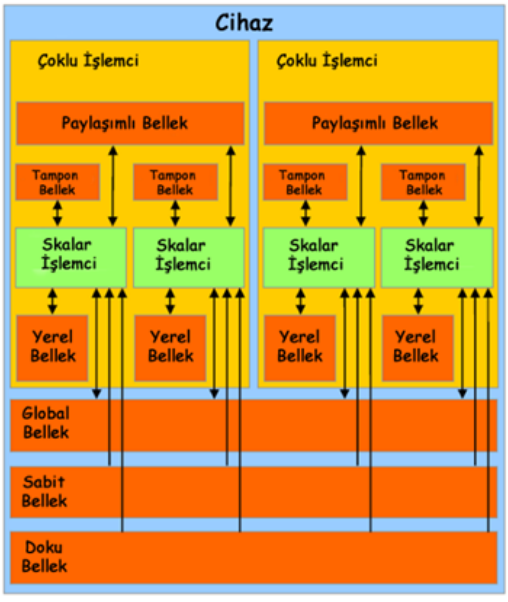
Şekil 2. Cihaz mimarisi [4] (Device architecture)

Cihaz içerisinde çok sayıda çoklu işlemci modülü (SM - Streaming Multiprocessor), her bir SM içerisinde skalar işlemciler (SP - Scalar Processor) ve bellekler bulunmaktadır.

CUDA platformu tarafından kullanılan ekran kartı içerisinde farklı bellek çeşitleri bulunmaktadır:

- Tampon Bellek (Register Memory)
- Yerel Bellek (Local Memory)
- Paylaşımlı Bellek (Shared Memory)
- Global Bellek (Global Memory)
- Sabit Bellek (Constant Memory)
- Doku Bellek (Texture Memory)

NVIDIA GPU içerisinde belleklerin şematik gösterimi Şekil 3'te verilmektedir. Her bir SM içerisinde SP, tampon bellek, yerel bellek, paylaşımlı bellek ve önbellekler bulunmaktadır.



Şekil 3. GPU Bellek Modeli [5] (GPU memory model)

Tampon ve yerel bellek en hızlı bellek çeşitleridir. Yerel bellek her bir iş parçacığı için kendine ait verileri depolar. Paylaşımlı bellek hızlı çalışan bellek çeşitlerindendir. Paylaşımlı bellek aynı SM içerisinde tüm iş parçacıkları tarafından kullanılır. Çoklu işlemci içerisindeki işlemcilerin tümü paylaşımlı belleğe aynı hızla ulaşabilmektedir. Global, sabit ve doku bellek ise ekran kartındaki çoklu işlemci modüllerinin dışında bulunmaktadır. Skalar işlemcilerin bu belleklere ulaşım süresi çoklu işlemci içerisindeki belleklerden daha fazladır. Veriler rasgele erişimli bellek (RAM -Random Access Memory)'ten GPU'ya aktarılırken global belleğe gelirler. Aslında sabit ve doku bellekler de global bellek içerisinde ayrılan bir bölümdedir.

2.2.2 CUDA programlama (CUDA programming)

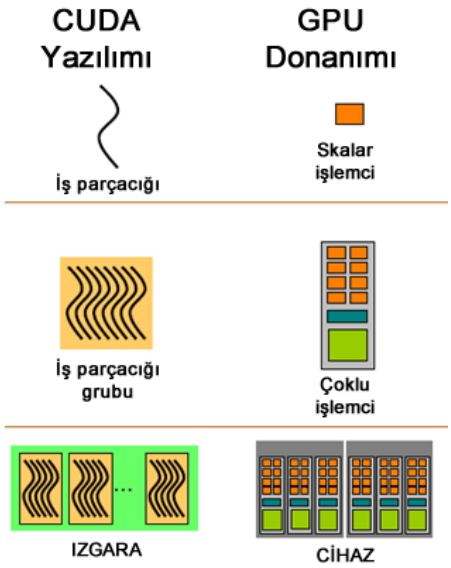
Bir GPU, içerisinde ızgara (grid) adı verilen yapıları barındırır. Bir ızgara içerisinde bloklar bulunur. Bu bloklar içerisindeki işlemcilerde iş parçacıkları (threadler) çalıştırılır. GPU üzerinde çalışan fonksiyonlara "kernel" adı verilmektedir. Bir kernel bir ızgaranın içerisindeki tüm iş parçacıklarında aynı anda tekli işlem çoklu veri (SIMD-Single Instruction Multiple Data) mimarisine uygun olarak farklı veriler üzerinde aynı işlemi yapacak şekilde çalışır. Fiziksel olarak paralel işleyen iş parçacığı gruplarına warp adı verilmektedir. Warp'ların her biri 32 iş parçacığı grubundan oluşmaktadır.

Bir CUDA programı hem CPU'da hem de GPU'da işlenecek olan kodları içerir. Bu kodların CPU'da işlenecek olan bölümlerinde herhangi bir veri paralelliği olması zorunlu değildir. GPU'da işlenecek olan kod veri paralelliği içeren kodlardan oluşur. NVIDIA derleyicisi kodu derleme sırasında iki bölüme ayırır. Yönetici kodu CPU'da çalışır ve standart C derleyicisi tarafından derlenerek çalıştırılır. Cihaz kodları ise kernel'lerdir. Kernel'ler CUDA Yazılım Geliştirme Kiti (SDK-Software Development Kit) tarafından çalıştırılırlar. Kernel'ler çok sayıda iş parçacığından oluşur. Bu iş parçacıkları CPU'da seri olarak ve çok sayıda devirde işlenirken, GPU'da aynı anda her biri farklı bir işlemcide farklı veriler ile ve aynı kodla işlenirler.

GPU üzerinde çalışacak bir kernel'in işlem adımları şu şekildedir:

1. İşlenecek olan veriler CPU'dan GPU'ya aktarılır.
2. CPU, GPU'ya kernel'in işlenmesi için komut verir.
3. GPU kernel'i işler.
4. Sonuçlar GPU'dan CPU'ya aktarılır.

CUDA platformunda yazılım ve donanım kavramları eşleştirildiğinde Şekil 4'de verilen durum ortaya çıkmaktadır. Buna göre her bir iş parçacığı bir skalar işlemcide işlenir. Bir blok içerindeki iş parçacıkları bir çoklu işlemcide işlenirler. Bu işlemler eşzamanlı olarak gerçekleştirilir. Blokların bir araya gelmesiyle ızgara adı verilen yapılar oluşur. Bir kernel bir ızgara içerisindeki bloklarda çalıştırılır. Bir blokta aynı anda ancak bir kernel çalıştırılabilir.



Şekil 4. CUDA platformunda yazılım donanım kavramlarının eşleştirilmesi (Matching of software and hardware concepts in CUDA platform)

2.2.3 CUDA Kütüphaneleri (CUDA Libraries)

NVIDIA tarafından CUDA kullanımını yaygınlaştırmak amacıyla C ve C++ geliştiricileri için GPU hızlandırmalı uygulamalar içeren kütüphaneler geliştirilmiştir. Bu kütüphaneler program geliştiricilerin kodlarını daha kısa sürede GPU ile hızlandırmayı hedeflemektedir. En yaygın kullanılan GPU kütüphaneleri ve uygulama alanları Çizelge 1'de verilmektedir.

Her bir kütüphane için kullanıcı kılavuzlarında içerdikleri fonksiyonlar ve kullanımları anlatılmaktadır. NVIDIA gelişmiş dokümantasyonu [6] ile CUDA kullanımını arttırmayı hedeflemektedir.

Çizelge 1. CUDA Kütüphaneleri (CUDA Libraries)

Kütüphane Adı	Uygulama Alanları
cuBLAS	Temel matematiksel işlemler
cuSPARSE	Seyrek matrisler ile ilgili işlemler
cuSOLVER	cuBLAS ve cuSPARSE kütüphanelerini temel alan üst seviye paketler
cuRAND	Rasgele sayı üretme
NPP	Görüntü ve video işlemede kullanılan temel işlemler
nvGRAPH	Graf işlemleri
Thrust	CUDA, TBB (Threading Building Blocks) ve OpenMP'yi kullanan paralel programlama kütüphanesi

3. CUDA PLATFORMU KULLANILAN İLK UYGULAMALAR (FIRST APPLCATIONS USING CUDA PLATFORM)

CUDA platformu ilk olarak Şubat 2007'de yayınlanmıştır. Bu nedenle CUDA ile ilgili yapılan çalışmalar bu tarihten itibaren literatürde yer almaktadır. CUDA platformu ilk olarak matrislerin çarpımı gibi genel amaçlı problemlerin çözümünde kullanılmıştır. Yapılan uygulamalarda çok yüksek performans elde edilmemesine rağmen CUDA'nın ilk uygulamaları GPU teknolojilerinin hızla gelişmesini ve yaygınlaşmasını sağlamıştır. Bu bölümde CUDA ile ilgili ilk çalışmalar özetlenmekte ve değerlendirilmektedir.

Es ve İşler'in [7] 2007 yılında yayınladıkları çalışma GPU kullanımının ilk örneklerindendir. Çalışmada ışın izleme uygulaması GPU kullanımı ile gerçekleştirilmiştir. Çalışmada ileri eşyönsüz (anizotropik) satranç tahtası mesafe dönüşümleri kullanarak GPU tabanlı kesişim algoritması geliştirilmiştir. Önerilen kesişim algoritmasının diğer ızgara tabanlı algoritmalara göre paralel akış işlemeye daha uygun olduğu ve hızlı çalıştığı belirtilmiştir. Yine aynı yılda Göddeke ve diğerlerinin [8] yaptıkları çalışmada GPU hesaplama yapabilecek bir kümede (cluster) bir uygulamanın çalıştırılması sonucunda ölçülen işlem süreleri karşılaştırılmıştır. Çalışmada 160 makinadan oluşan bir küme üzerinde matris çarpımı yapılmaktadır. Uygulamada küme kullanıldığından iletişim MPI ile sağlanmakta ve veri iletişimi için de bir zaman harcanmaktadır. CPU ve GPU kullanıldığında elde edilen işlem zamanlarına göre tüm sonuçlarda GPU'nun avantaj sağladığı görülmektedir. Zwartva ve diğerleri [9] ise yerçekimli N-body simülasyonunu GPU kullanarak gerçekleştirmişlerdir. NVIDIA'nin Quadro ve Geforce ekran kartları kullanılmıştır. GRAPE serisi bilgisayarlarla sonuçlar karşılaştırılmıştır. [7] ve [8]'da herhangi bir hızlanma oranı verilmemesine rağmen [9]'da 20 kata kadar hızlanma elde edilmiştir. Algoritma Cg (C Graphics) programlama dili kullanılarak geliştirilmiştir. Cg programlama dili, OpenGL (Open Graphics Library) ve DirectX kütüphanelerini kullanmaktadır. Anderson ve diğerleri [10] ise çalışmalarında Kuantum Monte Karlo metodunun çözümünü gerçekleştirmişlerdir. Bu metot oldukça fazla işlem yükü gerektirmektedir. Ancak CPU kümelerde paralelleştirilmesi oldukça zordur. Oldukça fazla veri paralelliği barındırmaktadır. Çalışmada Kuantum Monte Karlo metodu GPU kullanılarak paralelleştirilmiştir. NVIDIA ekran kartı kullanılmaktadır. Kernellerde 30 kata kadar, uygulamada ise 6 kata kadar hızlanma sağlanmıştır. Kahan toplama formülü kullanılarak yapılan matris çarpımında en iyi performans elde edilmiştir. Monte Carlo metodunun uygulandığı başka bir çalışma Martinsen ve diğerlerinin [11] 2009 yılında yayınlanan çalışmasıdır. Foton taşıma modelleme için Monte Karlo algoritması CUDA araçları kullanılarak NVIDIA 8800GT grafik kartında uygulanmıştır. Algoritmanın seri hali C++ programlama dilinde yazılmıştır. Seri kısmı tek çekirdekli işlemcide çalıştırılmıştır. Paralel kısmı ise C ile CUDA 1.1 versiyonu kullanılarak yazılmıştır. Uygulamada hafıza kullanımına önem verilmektedir. Saniyede 110 milyon işlemi grafik kartı ile 70 kat hızlandıran bir uygulama gerçekleştirilmiştir.

Che ve diğerleri [12] yaptıkları çalışmada OpenMP (Open Multi-Processing) ve CUDA konusunda kapsamlı bir araştırma yayınlamışlardır. Belirli bir uygulama olmamakla birlikte o dönemde çok yeni bir konu olmasından dolayı genel sonuçlar ortaya koymuşlardır. Çalışmada tek iş parçacığı ve çoklu iş parçacıklarında çalışmanın etkileriyle birlikte tek iş parçacığıyla çalışmanın hızı arttıracağı belirtilmektedir. Bellek kullanımı ve yığın çatışmasının (bank conflict) oluşumu anlatılmakta ve kazancı artırmak için yığın çatışmasının önüne geçilmesi gerektiği belirtilmektedir. Elde edilen sonuçlara göre 70 kata kadar hızlanma sağlanmıştır.

Görüntü işlemede GPU kullanımı ile ilgili literatürde rastlanan ilk çalışma 2008 yılında Diez ve diğerlerinin [13] çalışmasıdır. Çalışmada farklı yöntemlerle görüntü geriçatma işlemi gerçekleştirilmiştir. Bu yöntemler GPU üzerinde paralelleştirilmiş, işlem zamanları ve kazançları farklı çözünürlükteki resimler için gerçekleştirilmiştir. Sürekli iteratif geriçatım (SIRT-Simultaneous Iterative Reconstruction Technique) ve sürekli cebirsel geriçatım (SART- Simultaneous Algebraic Reconstruction Technique) teknikleri GPU üzerinde uygulanmıştır. CPU ile karşılaştırıldığında 60 ile 80 kat arasında kazanç elde edilmiştir. Elde edilen kazanç bu konuda yapılacak çalışmalar için çok ümit verici olmuştur. Aynı yıl yayınlanan Stone ve diğerlerinin [14] çalışması ise görüntü işlemede CUDA kullanılmasının ilk örneğidir. Çalışmada manyetik rezonans (MR-Magnetic Resonance) görüntüleri üzerinde geriçatma algoritmaları GPU üzerinde çalışacak hale getirilmektedir. CUDA 1.0 ve CUDA 1.1 karşılaştırması yapılmıştır. FHd algoritmasının seri ve CUDA'da yazılmış kodları makalede verilmektedir. Ayrıca CUDA çalışırken verilerin hafızadaki hareketi şematik olarak gösterilmiştir. Hafızada sabit ve tampon belleğe verilerin aktarım farkının hıza etkisi tablo halinde verilmiştir. 2 ile 9 kat arasında hızlanma sağlanmıştır. Global belleği çok kullanan uygulamalar daha yavaş çalışırken yazmaçları ve sabit belleği kullanan uygulamaların daha hızlı çalıştığı belirtilmektedir.

2008'de yapılan diğer çalışmalarda çeşitli uygulamalar paralelleştirilmiştir. Schenk ve diğerleri [15] çalışmada matris çarpımını paralelleştirmektedirler. Farklı matris boyutlarına göre performans ve bant genişliği incelenmiştir. 32 bit CPU, 64 bit CPU ve GPU'nun performans karşılaştırması yapılmış ve 64 bit CPU'ya göre GPU'da 6.5 kata kadar hızlanma görülmüştür. Matris çarpımının paralelleştirildiği diğer bir çalışma Diez ve diğerlerinin [16] çalışmasıdır. Matris çarpımının yanı sıra Hızlı Fourier Dönüşümü (FFT -Fast Fourier Transform) de GPU ile gerçekleştirilmiştir. Çalışmada 2 Boyutlu (2B) Hızlı Fourier Dönüşümü ve 3B FFT GPU uygulaması CPU ile karşılaştırılmıştır. Fourier dönüşümü farklı filtreler ile uygulanmıştır. Elde edilen sonuçlara göre en iyi hızlanma lineer filtre ile 256X256X256 büyüklüğündeki verilerde 24 kat ölçülmüştür. Belleman ve diğerleri [17] yaptıkları

çalışmada N-body benzetimini GPU kullanarak gerçekleştirmişlerdir. Uygulamalar Geforce 8800 GTX ekran kartı üzerinde çalıştırılmıştır. Çalışmanın ilk sonuçları 2007 yılında yayınlanmıştır. 2007'de Cg dili ile uygulama yapılmışken bu makalede CUDA kullanılmıştır. Global belleğe ulaşım yavaşken (400-600 çevrim(cycle)) paylaşımlı belleğin hızlı olduğu (4 çevrim) belirtilmektedir. Bu nedenle çalışmada paylaşımlı bellek en verimli şekilde kullanılmıştır. Veriler küçük parçalara ayrılmıştır. Uygulama 128 iş parçacığı içermektedir. GRAPE 6 kütüphanesi kullanılarak uygulama gerçekleştirilmektedir. Elde edilen sonuçlara göre 230 kata kadar performans artışı görülmüştür. Çalışma, temel bilgiler içermekte olup GPU donanımının verimli kullanımı konusunda iyi bir kaynaktır.

Ram ve diğerleri [18] çalışmalarında en küçük kareler yöntemini kullanmışlardır. Metroloji sistemleri yüzey bilgilerini (x,y,z) noktası olarak koordinat bilgileri şeklinde alırlar. Bu bilgiler en küçük kareler yöntemi ile koni, silindir, düzlem gibi şekillere dönüştürülür. En küçük kareler yöntemi noktalardan şekillere dönüştüren tipik bir yöntemdir. Noktalar arasındaki uzaklıkların toplamını minimize eder. Büyük sistemlerde bu problemin çözümü yüksek işlem zamanı gerektirir. Çalışmada Gauss-Newton metodunun GPU ile CUDA platformu kullanılarak uygulaması yapılmıştır. Farklı nokta sayıları ile yapılan uygulama sonuçları karşılaştırıldığında GPU ile elde edilen hızlanma CPU'ya göre en fazla 18 kat olarak ölçülmüştür. Geometrik şekillere göre farklı hızlanmalar elde edilmiş ve en iyi hızlanma 18 kat ile halkada gerçekleşmiştir. Uygulaması C++'da gerçekleştirilmiştir. Ryoo ve diğerleri [19] çalışmalarında mevcut programların GPU kullanarak hızlı bir şekilde yapılması için kod optimizasyon yöntemi önermiş ve dört farklı uygulama üzerinde gerçekleştirmişlerdir. Uygulamalarda Geforce 8800 GTX ekran kartı ve CUDA platformu kullanmışlardır. Çalışmada performansı arttırmak için veriler bölünerek GPU'da işlenmektedir. Paylaşımlı bellek kullanılması ile global belleğe erişimin azaltılması sayesinde verilere erişim kısaltılarak hızlanma sağlanmaktadır.

2009 yılındaki yayınlar arasında Walsh ve diğerlerinin [20] çalışması yer almaktadır. Lattice-Boltzman kodu, sonlu elemanlar analizi ve en küçük kareler yöntemini uygulamışlardır. Makalede 3 uygulamanın GPU implementasyonu gerçekleştirilmiştir: Akışkanlar dinamiği, sismik dalga yayılımı ve kaya manyetiği. Bu uygulamalar önemli sayısal modelleme tekniklerini içermektedirler ve fen ve mühendislik alanında kullanılan önemli uygulamalardır. Uygulamalarda 10 ile 30 kat arasında hızlanma sağlanmıştır. GPU programlamada grafik kartının hafıza büyüklüğü, GPU ve CPU arasındaki veri transferi hızı ve simülasyondaki elemanların karmaşıklık derecesinin karşılaşılan sınırlamalar olduğu belirtilmektedir. Sonlu elemanlar analizinin uygulandığı başka bir çalışma Komatitsch ve diğerlerinin [21] çalışmasıdır. Depremlerden alınan sonuçlar üzerinde uygulama yapılmıştır. Uygulama CUDA kullanılarak tek hassasiyetli kayan noktalı (single precision) sayılar ile gerçekleştirilmiştir. Uygulamalar Geforce 8800 GTX ve Geforce GTX 280 ekran kartları üzerinde çalıştırılmıştır. Veri iletim süresi hariç sırasıyla 15 ve 25 kata kadar, veri iletim süresi dahil 5 ve 7 kata kadar hızlanma elde edilmiştir.

GPU'nun kullanıldığı ilk çalışmalar incelendiğinde, genel amaçlı GPU programlamanın çok farklı alanlarda kullanıldığı görülmektedir. Standart ve bilinen bazı algoritmaların paralelleştirilmesi GPU kullanılarak gerçekleştirilmiştir. Bu çalışmaların bazılarında hızlanma için sayısal değer verilmemekle birlikte sadece hızlanma sağlandığı beyan edilmiştir. İlk GPU uygulamaları; GPU programlamada donanım kullanımı, hafıza yönetimi, kernel'lerin çalıştırılması gibi konulara açıklık getirdiğinden elde edilen sonuç ve uygulanan alan ne olursa olsun çok önemli çalışmalardır. Gerçekleştirilen uygulamalar bilinen algoritmaların paralelleştirilmesi olmakla birlikte GPU programlamanın ilk ve referans alınacak çalışmalarıdır. Yüksek performans elde etmek için GPU programlama ve CUDA kullanılan ilk çalışmalarda elde edilen sonuçlar umut verici olmuş, bu teknolojinin hızlı gelişimini tetiklemiştir.

4. GÖRÜNTÜ İŞLEMEDE CUDA UYGULAMALARI (CUDA APPLICATIONS ON IMAGE PROCESSING)

GPU programlamada CUDA platformu kullanılan paralel görüntü işleme çalışmaları; görüntü geriçatma, iyileştirme, bölütleme, çakıştırma ve sınıflandırma konu başlıkları altında aşağıda detaylı bir şekilde incelenmiştir.

4.1 Görüntü Geriçatma (Image Reconstruction)

Nesnelerin 2B ve 3B yüzey görüntüleri kameralar kullanılarak elde edilebilir. Öte yandan nesnelerin kesitlerinin görüntüsünü elde etmek için özel cihazlar ve sensörler kullanılmaktadır. Bu cihazlarla toplanan verilerin matematiksel işlemlere tabii tutularak bilgisayar ortamında görüntüye dönüştürülmesi işlemine görüntü geriçatma denir. Görüntünün oluşturulması için gerçek bir nesneden cihazlar ile farklı açılardan elde edilen sinyallerin her biri dönüştürülüp tek bir görüntü üzerinde birleştirilerek gerçek nesneye en yakın görüntü elde edilmeye çalışılır. Geriçatma işlemi ileri ve geri izdüşüm (forward and backward projection), ağırlıklandırılmış geri izdüşüm (WBP), iteratif geriçatma, eşzamanlı iteratif geriçatma (SIFT), FDK (Feldkamp Davis Kress), adaptif en küçük dikdörtgen kapama (AMER-Adaptive Minimum Enclosing Rectangle), vb. algoritmalarla gerçekleştirilmektedir. Bu algoritmalar uzun zaman alabilmektedir. Bu zamanın kısaltılması için GPU uygulamaları yapılmaktadır. Görüntü geriçatmada CUDA kullanımı ile ilgili literatürde karşılaşılan çalışmalar aşağıda değerlendirilmektedir.

İncelenen çalışmalarda görüntü geriçatmada ileri ve geri izdüşüm yönteminin CUDA ile paralelleştirmede en çok kullanılan yöntem olduğu görülmektedir [22, 23, 24, 25]. Vazquez ve diğerleri [22] 2010 yılında yaptıkları çalışmada ağırlıklandırılmış geriye izdüşüm (weighted back

projection) yöntemi ile tomografik görüntülerin geriçatımını CUDA ile gerçekleştirmişlerdir. Çalışmada dört farklı metot kullanılmış ve GPU'da gerçekleştirilmiştir. Farklı veri seti büyüklükleri için yapılan uygulamalar CPU ve GPU'da çalıştırılmış ve karşılaştırılmıştır. Uygulamalar Geforce GTX 295 ekran kartında çalıştırılmıştır. Ekran kartı her biri 8 çekirdek içeren 30 çoklu işlemci ve 1.2 GHz, 896 MB hafıza içermektedir. GPU'da CPU'ya göre 40 kata kadar hızlanma sağlanmıştır. Geriye projeksiyon yöntemini kullanan diğer bir çalışma Noel ve diğerlerinin [23] 2010 yılında yaptıkları çalışmadır. Çalışmada ağırlıklandırılmış filtrelenmiş geriye izdüşüm ile görüntü geriçatma işlemi CUDA kullanarak çift hassasiyetli kayan noktalı mimari ve basit mimari ile uygulanmaktadır. Konik ışınlı (cone beam) bilgisayarlı (computed) tomografi ile 3B görüntüler elde edilmektedir. Konik ışınlı tomografi kullanılarak kısa tarama ile 60 saniyede tarama yapılmaktadır. Standart yöntemlerle 256x256x256 boyutlarındaki bir görüntünün geriçatımı 25 dakikanın üzerinde zaman almaktadır. GPU kullanılarak bu işlem çok daha kısa sürede gerçekleştirilmektedir. Makalede CUDA kullanılarak NVIDIA Geforce GTX 280 ekran kartlarında 3B görüntüler geriçatılmaktadır. İşlem zamanı 256x256x256 görüntülerde 3.2 saniyeye, 512x512x512 görüntülerde 8.5 saniyeye düşürülmüştür. Çift hassasiyetli kayan noktalı mimaride, basit mimari ile karşılaştırıldığında %60 oranında hızlanma görülmüştür. Zheng ve diğerleri [24] çalışmalarında iteratif geriçatma yöntemini GPU sayısını artırarak daha kısa sürede gerçekleştirmektedirler. Herraiz ve diğerleri [25] ise çalışmalarında pozitron emisyon tomografisi ile elde edilen görüntüleri ileri ve geri projeksiyon yöntemini kullanarak geriçatmaktadırlar. Bu işlemi GPU ile CPU'ya göre 72 kata kadar hızlandırmışlardır.

Xu ve diğerleri [26] 2010 yılında 3B elektron tomografi iteratif geriçatma algoritmasını CUDA platformunu kullanarak hızlandırmışlardır. Başka bir çalışmada ise Xu ve diğerleri [27] yine görüntü geriçatmanın paralelleştirilmesini GPU ile gerçekleştirmiş ancak farklı bir algoritma kullanmışlardır. Olasılıkları en iyileme (expectation maximization) ve eş zamanlı iteratif geriçatma tekniği bilgisayarlı tomografide sık kullanılan geriçatma algoritmalarıdır. Bu teknikler yakınsamanın artmasıyla popüler ve daha performanslı teknikler haline gelmektedirler. Bu çalışmada eşzamanlı iteratif geriçatma tekniği GPU ile hızlandırılarak gerçek zamanlı uygulamalarda kullanılabilirliği arttırılmış ve optimize edilmiştir. Palenstijn ve diğerleri [28] 2011 yılında yaptıkları çalışmada Xu ve diğerlerinin [26] çalışmasını iyileştirmişlerdir. Çalışmada thread bloklarının ve belleğin daha verimli kullanılması sayesinde 10 kata varan hızlanma elde edilmiştir. Başka bir çalışmada ise Blas ve diğerleri [29] 2014 yılında X-ray görüntüleri üzerinde geri izdüşüm yöntemi ile görüntü geriçatım işlemini gerçekleştirmişler ve GPU implementasyonu ile hızlandırmışlardır. Bu konuda yapılan çalışmalar ile karşılaştırılmış ve çalışmada 3 kat hızlanma sağlanarak diğer benzer çalışmalardan daha iyi sonuçlar alınmıştır.

İteratif izdüşüm kullanmasına rağmen Flores ve diğerlerinin [30] 2013 yılında yaptıkları çalışmanın yukarıdakilerden farkı CUBLAS ve CUSPARSE kütüphanelerini kullanmaları olmuştur. Bu çalışmanın amacı da yüksek kalitedeki resimleri düşük örnekleme ve gürültülü izdüşüm verileri ile geriçatmada hızlandırmalı GPU tabanlı algoritma kullanmaktır. Çözünürlüğün yükselmesiyle 19 kata kadar hızlanma elde edilmiştir. Aynı kütüphaneleri kullanan diğer bir çalışma ise Flores ve diğerleri [31] tarafından 2014 yılında yapılmıştır. Farklı çözünürlüklere göre işlem süreleri değişmekle birlikte en iyi sonuçlar yüksek çözünürlüklerde elde edilmiştir. GPU'da elde edilen işlem süresi CPU'ya göre 4 kata kadar azaltılmıştır. Feng ve diğerleri [32] 2013 yılında yaptıkları çalışmada görüntü geriçatmada GPU'yu farklı bir kütüphane üzerinde kullanmışlardır. PROPELLER klinik uygulamalarda veri ve hareket doğrulama metotlarını kullanarak görüntü geriçatma gerçekleştiren bir sistemdir. Bu çalışmada hareket ve yön doğrulama ile görüntü geriçatmada GPU kullanılmaktadır. Gerçekleştirilen çalışmada 6 kata kadar hızlanma elde edilmiştir. Kütüphane kullanan diğer bir çalışma ise Li ve diğerlerinin [33] 2010 yılında yaptıkları çalışmadır. 3B görüntü geriçatma için FREALIGN isimli bir kütüphane kullanmışlardır. Kütüphane ile dağıtık programlama için çok işlemcili küme üzerinde uygulama kolaylıkla yapılabilmektedir. Kütüphanenin işlemi tamamlama süresi, veri setindeki parça sayısı ve CPU sayısına göre değişmektedir. Son versiyonu MPI desteklidir. Bu çalışmada FREALIGN'a GPU programlama eklenmiştir. 10-240 kat arasında hızlanma sağlanmıştır. GPU üzerinde tek parçacıklı 3B geriçatma algoritması gerçekleştirilmiştir. Fourier dönüşümü kullanılarak geriçatma gerçekleştirilmiştir. Ayrıca tek GPU ile 4 ve 8 GPU karşılaştırılmıştır.

Okitsu ve diğerleri [34] 2010 yılında yaptıkları çalışmada geriçatmada çok kullanılan algoritmalardan biri olan FDK algoritmasını kullanarak CUDA platformunda hızlanma sağlamışlardır. Konik ışın görüntü geriçatma işleminde 3 teknik kullanmaktadırlar. Bu teknikler: (1) Bellek bant genişliğini korumak için GPU yongası dışındaki belleğe ulaşımın en aza indirilmesi, (2) Bellek gecikmesini önlemek için döngülerin açılması, (3) Birden fazla GPU için çoklu iş parçacıklarının kullanılmasıdır. Bu teknikler görüntü geriçatma için kullanılmaktadır. Elde edilen sonuçlarda teorik bant genişliğinin %83'ü kullanılmış ve 512x512x512 boyutlarında olan vokselde (bir noktayı 3B uzayda tanımlayan grafik bilgisi) saniyede 64.3 izdüşüm değerine ulaşılmıştır. Bu performans standart CUDA tabanlı metottan %41 daha hızlı, CPU'dan ise 24 kat daha hızlıdır. Aynı algoritmayı kullanan başka bir çalışma Scherl ve diğerleri [35] tarafından 2012 yılında gerçekleştirilmiştir. Yapılan çalışmada hem CPU çekirdekleri üzerinde hem de GPU üzerinde algoritmayı paralelleştirmişlerdir. GPU ile paralelleştirmede CUFFT kütüphanesini kullanmışlardır. Optimize edilmiş geriye izdüşüm uygulaması ile GPU'da 22 kata kadar hızlanma elde edilmiştir.

2011 yılında Chang ve diğerleri [36] incelenen geriçatma yöntemlerinden farklı bir yöntem üzerinde çalışmışlardır.

Çalışmada yeni çok görüntülü stereo tabanlı yüzey eleman algoritması önerilmektedir. Farklı noktalardan alınan görüntüler ile stereo eşleme gerçekleştirilmekte, daha sonra yüzey elemanları yerleştirilmekte, ardından da çizgeler (graph) oluşturulmaktadır. 3B görüntülerin geriçatımı bu şekilde sağlanmaktadır. İşlem zamanını düşürmek amacıyla GPU kullanılmaktadır. Bu yaklaşım ile 3B görüntüler CPU'ya göre 100 kat daha hızlı geriçatılmaktadır.

2012 yılında Jiang ve diğerleri [37] CT görüntüleri üzerinde buğday fidanlarını sayan uygulamayı GPU ile paralelleştiren oldukça farklı bir çalışmaya imza atmışlardır. Buğday, yetiştirilen tarlalarda elle sayılmaktadır. Bu çalışmada fidanları Xray bilgisayarlı tomografi ile otomatik sayan bir sistem geliştirmişlerdir. CT görüntülerini geriçatmayı gerçek zamanlı bir uygulamada çalıştırabilmek için zaman kısıtlaması vardır. Bunun için geriçatma GPU ile hızlandırılmıştır. Çalışmada AMER yöntemi ile çalışma alanı üzerinde oluşturulan görüntü üzerindeki piksel sayısı çıkarılmaktadır. Kullanılan iki yöntem de GPU ile hızlandırılmıştır. CPU ile GPU karşılaştırılmasında 200 kata kadar hızlanma elde edilmiştir. AMER ve GPU'nun birlikte kullanıldığı durumda ise CPU'ya göre 1100 kata kadar hızlanma elde edilmiştir.

Agulleiro ve diğerleri [38] 2012 yılında yaptıkları çalışmada 3B tomografi görüntülerini geriçatmada hibrit bir sistem kullanmışlardır. En çok kullanılan görüntü geriçatma yöntemlerinden ağırlıklandırılmış geri izdüşüm ve eşzamanlı iteratif geriçatma algoritmalarını paralelleştirmişlerdir. Hibrit sistemde çok çekirdekli CPU ve GPU'yu birlikte kullanmışlardır. Sadece CPU ve sadece GPU paralelliğine göre 1,5-2 kat hızlanma elde etmişlerdir.

2013 yılında yayınlanan ve manyetik rezonans (MR-Magnetic Resonance) görüntülerinin geriçatımında farklı algoritmalar kullanan diğer çalışmalar şu şekildedir: Yang ve diğerleri [39] MR görüntü geriçatımında ızgaralama (gridding) algoritmasını geleneksel olarak Fourier dönüşümü kullanarak gerçekleştirmektedirler. Çalışmada ters ızgaralama algoritmasında geleneksel ızgaralama algoritmasından farklı olarak her bir eğri için bir ızgara penceresinde işlem yapılmaktadır. CUDA tabanlı ters ızgaralama algoritmasında 7.5 kat hızlanma elde edilmiştir. Piccialli ve diğerleri [40] ikinci derecede türevli olan bir yaklaşım kullanmaktadırlar. 3B görüntüler üzerinde geriçatma işlemi çok çekirdekli CPU ve çoklu GPU'da çalıştırılmış ve 25 kata kadar hızlanma elde edilmiştir. Monte ve diğerleri [41] Monte Karlo yöntemini GPU paralel programlama teknikleri kullanarak uygulamakta ve seri algoritma ile paralel algoritmanın işlem sürelerini karşılaştırmaktadırlar. Gai ve diğerleri [42] yaptıkları çalışmada MR görüntüleri üzerinde geriçatım gerçekleştirmekte ve GPU ile hızlandırmada optimizasyon sağlamaktadırlar. GPU kullanımında cuFFT kütüphanesini, hızlanma için paylaşımlı belleği kullanmaktadırlar. Bununla birlikte görüntü ızgara şeklinde bölümlere ayrılarak gereksiz bölümlerin işlenmesi engellenmektedir. Böylece performans artışı elde edilmektedir.

Bilgisayar mimarilerini karşılaştıran ve özgün çalışmalardan biri olan Birk ve diğerlerinin [43] 2014 yılında yaptıkları çalışmada 3B bilgisayarlı tomografi görüntülerinin geriçatımı için iki algoritma kullanılmıştır. Bu algoritmalar GPU ve alanda programlanabilir kapı dizileri (FPGA-Field Programmable Gate Array) üzerinde çalıştırılmış ve performans analizi yapılmıştır. Çalışmada GPU ve FPGA'lar donanımsal olarak incelenmiştir. Fermi ve Kepler mimarisinin donanımsal farkları verilmiştir. 40 nm (nanometre) ve 28 nm mimarilerinin benzerlikleri ve farkları ortaya konmuştur. 40 nm mimarili GPU ve FPGA'larda çalıştırılan iki algoritmada birbirine yakın performans elde edilmiştir. 28 nm mimarili FPGA'lar ise GPU'ya göre farklı algoritmalarda %86 ve % 39 oranında daha iyi sonuç vermiştir. Chang ve diğerleri [44] yaptıkları çalışmada MR görüntülerinin geriçatması için yayılım ağırlıklı görüntüleme yöntemini GPU kullanarak yüksek hızlanma ile gerçekleştirmişlerdir. Veri setlerine göre farklı oranlarda hızlanma sağlanmıştır. Verilen sonuçlara göre 490 kata kadar hızlanma sağlamışlardır. 3B model oluşturmada eğitimsiz bir yapay sinir ağı algoritması olan nöral gaz algoritmasını Orts-Escolano ve diğerleri [45] 2015 yılında yaptıkları çalışmada kullanmışlardır. Bu algoritma ile nesnelerin 3B modellerini oluşturmuşlar ve görüntü geriçatma işlemini gerçekleştirmişlerdir. Algoritmayı GPU kullanarak hızlandırmışlar ve farklı boyutlardaki görüntülerde farklı hızlanmalar elde etmişlerdir. CUDA kullanarak 180 kata kadar hızlanma sağlamışlardır.

Görüntü işlemede CUDA uygulanan çalışmalar içerisinde en geniş yerin görüntü geriçatmada olduğu görülmektedir. Görüntü geriçatmada en çok kullanılan ileriye ve geriye izdüşüm yöntemlerinin, paralelleştirilen CUDA platformu uygulamalarında da en yoğun kullanılan algoritmalar olduğu görülmektedir. Uygulamalar büyük oranda tıbbi veriler üzerinde, tıbbi uygulamalar da genellikle CT ve MR görüntüleri üzerinde gerçekleştirilmiştir. Görüntü geriçatmada iteratif yöntemlerin kullanılması bu nedenle yüksek işlem gücü gerektirmesi CUDA'nın bu uygulamalarda yoğun bir şekilde kullanılmasına neden olmuştur. Ayrıca alınan görüntülerin 3B olması nedeniyle sayısal olarak fazla veri içermesi daha fazla performans elde edilmesini sağlamaktadır. Her bir veri üzerinde aynı algoritmanın uygulanması ve komşu piksel ilişkisinin olmaması ise paralelleştirme seviyesini artırmakta ve paralelleştirmeyi kolaylaştırmaktadır. Çünkü paylaşımlı bellek kullanımına ve karmaşık bellek yönetimine ihtiyaç duyulmadan en iyi hızlanma sağlanabilmektedir.

4.2. Görüntü İyileştirme (Image Enhancement)

Görüntü iyileştirme alanında uygulanan tekniklerin nihai hedefi, görüntünün niteliğini istenilen ve beklenen şekilde yükseltmektir. Görüntü iyileştirme, gerçek görüntüden farklı olarak alınan görüntülerde var olan gürültü, bulanıklık, renk, ışık vb. gibi problemlerin ortadan kaldırılması ve görüntünün mümkün olduğunca gerçeğine yakın hale getirilmesi işlemlerinin tümüdür. Öte yandan görüntü işleme adımlarının başarısını artırmak için de görüntü iyileştirme algoritmaları kullanılabilir. Kontrast ve

parlaklık düzenlemeleri, kenar zenginleştirme, histogram eşitleme gibi uygulamalar, kişinin görsel algılamasını etkilemeyi hedefleyen ve bu konuda başarı sağlayabilen en temel görüntü iyileştirme teknikleridir. Görüntü iyileştirmede kullanılan algoritmalar genellikle temel teknikleri içermektedir. Bu algoritmaların işlem süreleri de (görüntünün çözünürlüğü ile değişmekle birlikte) genellikle diğer görüntü işleme adımlarına göre daha azdır. Görüntü iyileştirme teknikleri konusunda yapılan CUDA uygulamaları tarih sırasına göre aşağıda verilmektedir.

Görüntü iyileştirme konusunda ilk çalışmayı 2010 yılında Huhle ve diğerleri [46] yapmışlardır. Çalışmada renkli görüntüler üzerinde gürültü ayıklama işlemi gerçekleştirmişlerdir. Çalışmada GPU uygulaması olmasına rağmen GPU'ya çok az yer verilmiş ve hızlanmayla ilgili herhangi bir veri, tablo veya grafik verilmemiştir. Sadece hızlandığı bir cümle ile söylenmiştir. GPU ve CUDA kullanımında öncelikli hedef hızlanma olduğundan elde edilen hızlanmaların net olarak verilmesi beklenmesine rağmen çalışmada sayısal bir değer verilmemektedir ancak çalışma gürültü temizleme konusunda ulaşılan ilk çalışma olması açısından önemlidir.

Gürültülerin temizlenmesinde CUDA kullanan başka bir çalışma Sanchez ve diğerleri [47] tarafından 2013 yılında yayınlanmıştır. Algoritmada fuzzy metriği kullanılmaktadır ve eş grup konsepti tabanlıdır. Algoritma aynı zamanda çok çekirdekli platformlardan OpenMP kullanılarak gerçekleştirilmiştir. Performans işlem zamanına göre değerlendirilmiş ve büyük miktardaki veriler CPU ve GPU'ya paylaştırılmıştır. Çalışmada elde edilen sonuçlar gürültü yok edici filtrenin paralel halinin gerçek zamanlı uygulamalara kapı açacak nitelikte olduğunu göstermektedir. Uygulama 3 farklı şekilde gerçekleştirilmiştir. Birinci uygulamada çok çekirdekli işlemcide OpenMP kullanılmış ve 16 çekirdekli makinada 10 kata kadar hızlanma elde edilmiştir. İkinci uygulamada GPU'da CUDA kullanılarak 4 GPU'da 30 kata kadar hızlanma elde edilmiştir. Bir de hem GPU hem de CPU kullanılarak uygulama çalıştırılmıştır. İşin 1/8 i CPU'ya 7/8 'i GPU ya yaptırılarak en iyi hızlanma elde edilmiştir.

GPU'da çözünürlük arttıkça hızlanma oranının da arttığını [48] ve [49] çalışmalarında belirtmişlerdir. Urena ve diğerleri [48] yaptıkları çalışmada yeni bir kontrast iyileştirme tekniği önermektedirler. 320x240, 640x480 ve 600x800 olmak üzere üç farklı çözünürlükte elde edilen sonuçlara göre en iyi sonuç 600x800 çözünürlüğünde görüntülerde elde edilmektedir. CPU ile saniyede 2 görüntü işlenirken GPU kullanılarak saniyede 20 görüntü işlenir hale gelmektedir. Ma ve diğerleri [49] ise 2014 yılında yaptıkları çalışmada Laplace kullanarak görüntü keskinleştirmeyi CUDA ile hızlandırmaktadırlar. Klasik Laplace görüntü keskinleştirme algoritmasında tüm pikseller tek tek işlenmekte ve yüksek işlemci gücü gerektirmektedir. Özellikle yüksek çözünürlüklü resimler için Laplace keskinleştirme işleminde CPU işlem zamanı yüksek olmaktadır. Bu çalışmada Laplace keskinleştirme algoritmasının GPU üzerinde uygulamasında farklı hafıza tipleri kullanımı incelenmiş ve paylaşımlı hafıza kullanımının genel hafızaya göre daha performanslı çalıştığı görülmüştür. İki algoritmanın sonuçları verilmiştir. Farklı görüntü boyutlarına göre keskinleştirme işlemi OpenCV (Open Source Computer Vision) kütüphanesi fonksiyonları ile global bellek kullanılarak GPU ile ve paylaşımlı bellek kullanılarak GPU ile gerçekleştirilmiştir. En iyi sonuç 6,8 kat hızlanma ile paylaşımlı hafıza kullanılarak gerçekleştirilen GPU uygulamasında olmuştur. OpenCV ile karşılaştırıldığında çözünürlük yükseldikçe gerçekleştirilen paralel GPU algoritmasının kazancı artmaktadır.

Saha ve diğerleri [50] 2016 yılında yayınlanan çalışmalarında alçak geçiren filtre, yüksek geçiren filtre, Sobel kenar yakalama algoritmasını GPU ile paralelleştirmişlerdir. 3000X3000 çözünürlükteki görüntüler üzerinde elde edilen hızlanma sonuçlarını vermişlerdir. Sobel kenar yakalama algoritmasında 11 kat, alçak ve yüksek geçiren filtrelerde 12 kat hızlanma elde etmişlerdir.

Görüntü iyileştirme amacıyla kullanılan algoritmalar genel olarak komşu pikselleri kullanmaktadırlar. Bu algoritmalar GPU'da gerçekleştirilirken pencere boyutu dikkate alınarak algoritmanın paralelleştirilmesi gerekmektedir. Global bellekten verilere ulaşıldığında ulaşılmak istenen piksel sayısının artması ile işlem süresi artmaktadır. Paylaşımlı bellek kullanarak hızlanma arttırılmaya çalışıldığında ise bellek miktarı sınırlayıcı bir faktör olarak karşımıza çıkmaktadır. Tüm bu nedenler hem algoritmanın paralelleştirilmesini zorlaştırmakta hem de hızlanma oranını sınırlamaktadır. GPU kullanılarak hızlanma oranının arttırılması için optimizasyon yapılması gerekmektedir. Bu da her algoritma ve uygulama için farklı yaklaşımlar içermelidir. Bu nedenle üzerinde çalışma gerektirmektedir. Yapılan çalışmalarda standart algoritmaların GPU uygulamaları bulunmaktadır. Ancak bunların her biri geliştirilmeye açıktır.

Görüntü iyileştirme görüntü işlemenin önemli alanlarından biridir. Bu konuda yapılan uygulamalarda farklı iyileştirme metotları kullanılmaktadır. Bu metotların pek çoğunda belli bir büyüklükteki pencere resim üzerinde gezdirilmektedir. Bu pencerenin sayısal değerlerinin değişmesi ile kullanılan metot ismi de değişmiş olur. Ancak temelde paralelleştirme açısından değerlendirildiğinde aynı çözüm yolunu uygulamaktadır. Bu nedenle çalışmalar GPU paralelleştimesi açısından farklı bir yaklaşım getirmeyeceğinden çalışma sayısının sınırlı kaldığını söylemek yerinde olacaktır. Bunun yanında iyileştirme algoritmalarının toplam işlem sürelerinin çözünürlük arttıkça artmasına rağmen gerçek zamanlı uygulamalarda kullanılabilecek işlem zamanlarında gerçekleştirilebildiği söylenebilir. Bu da CUDA ile hızlanma ihtiyacını azaltmaktadır. Yapılan çalışmalarda elde edilen hızlanmalar iyi olmasına rağmen yukarıda açıklanan nedenlerden dolayı görüntü iyileştirme konusunda literatürde karşılaşılan çalışma sayısının görüntü geriçatım ve bölütleme çalışmalarından daha az olduğu görülmektedir. Ancak görüntü iyileştirme algoritmalarının çok kullanılmasından dolayı GPU uygulamalarının önemli

olduğu ve diğer görüntü işleme yöntemleri ile birlikte paralelleştirilmesinin ve optimizasyonunun yapılmasının işlem zamanına önemli katkılar sağlayacağı söylenebilir.

4.3. Görüntü Bölütleme (Image Segmentation)

Bölütleme görüntüyü her biri içerisinde farklı özelliklerin tutulduğu anlamlı bölgelere ayırma işlemidir. Uygulamaya bağlı olarak, kullanılan yöntemler ve yöntemlerin performansı değişiklik gösterebilir. Tüm görüntülere uygulanabilecek genel bir bölütleme yöntemi yoktur. Yani görüntü bölütleme için tasarlanan yöntemler ve bu yöntemlerin başarımları uygulamaya göre değişiklik göstermektedir. Bu nedenle de üzerinde çok çalışılan konulardan biridir. Görüntü bölütleme, görüntü işlemenin en zor işlemlerinden biri olmakla birlikte pek çok farklı yaklaşımı da içermektedir [51, 52]. Bu yaklaşımlar; histogram tabanlı, kümeleme tabanlı, bölge büyütme (region growing), ayırma ve birleştirme ve morfolojik olarak gruplandırılabilir. Bölütleme konusunda CUDA kullanılarak yapılan uygulamalar bu bölümde incelenmiştir.

Görüntü bölütleme konusunda karşılaşılan ilk çalışma 2008 yılında Pan ve diğerleri [53] tarafından yapılan çalışmadır. Çalışmada bölge büyütme ve havza (watershed) algoritmaları kullanılmaktadır. Tıbbi veriler üzerinde GPU ile bölge büyütme ve havza algoritmaları paralelleştirilmektedir. CUDA kullanılan uygulamalarda 2.5 kata kadar hızlanma sağlanmıştır. Hızlanma oranı düşük gibi görünse de bölütleme konusunda karşılaşılan ilk çalışma olduğundan önemlidir. Tıbbi veriler üzerinde uygulanan başka bir çalışma da Zhuge ve diğerlerinin [54] yaptıkları çalışmadır. Çalışmada diğerlerinden farklı olarak bulanık mantık kullanılmaktadır. Bulanık mantık ile görüntü bölütleme işleminin CUDA ile uygulanmasında 14.4 kata kadar hızlanma elde edilmiştir. Bulanık mantık kullanılan tek çalışma olması bu çalışmayı diğerlerinden ayırmaktadır. 2010 yılında Abramov ve diğerlerinin [55] yaptıkları çalışmada ise gerçek zamanlı olarak video görüntüleri üzerinde kullanılabilecek bir bölütleme işlemi gerçekleştirilmiştir. 1.2 Mega piksel boyutlarındaki renkli görüntüler 10 ms gibi kısa bir sürede bölütlenmiştir. Bölütleme işleminde sürenin kısaltılması için CUDA ile paralelleştirme kullanılmıştır. Elde edilen sonuçlarla algoritma gerçek zamanlı olarak uygulamalarda kullanılabilecektir. SAR (Synthetic Aperture Radar) görüntüleri üzerinde Sui ve diğerlerinin [56] 2012 yılında yaptıkları çalışmada bölütleme işlemi GPU ile hızlandırılmaktadır. Markov rasgele alan (Markov Random Field-MRF) metodu kullanılmış ve CUDA uygulamasında 10 kata kadar hızlanma elde edilmiştir. Membrane işleme kullanılarak gradient tabanlı kenar yakalama ile görüntü bölütleme işlemi gerçekleştiren Pernil ve diğerleri [57] GPU ile 8 kata kadar hızlanma elde etmişlerdir.

Özdemir ve Altılar [58] 2014 yılında yaptıkları çalışmada bölütleme için CUDA kullanmaktadırlar. Bitki büyütme işlemi için otomatik olarak analiz yapan bir sistem için 15 farklı yöntemle görüntü bölütleme gerçekleştirmektedirler. Bölütlemeyi GPU kullanarak hızlandırmışlar ve 6.5 kata kadar kazanç elde etmişlerdir. Çalışma farklı bir alanda CUDA kullanılması açısından önemlidir. Aynı yılda bölütleme konusunda yapılan başka bir çalışmanın uygulama alanı ise medical görüntüler olmuştur. Alvarado ve diğerleri [59] yaptıkları çalışmada bölütleme için OpenMP ve CUDA kullanmışlar ve 8 kata kadar hızlanma elde etmişlerdir.

Bergen ve diğerleri 60] 2015 yılında yaptıkları çalışmada MR (Manyetik Rezonans) görüntüleri üzerinde faz kontrast bölütleme algoritmasını basit ortalama tabanlı hesaplama ve en küçük kareler yöntemiyle gerçekleştirmişlerdir. Bu bölütleme işlemi çok çekirdekli CPU'da 10 saniye kadar sürmektedir. GPU tabanlı algoritma ile ise 0.5 saniyede gerçekleştirilmiştir. D'ambra ve diğerleri [61] 2016 yılında yayınlanan çalışmalarında 2B görüntüler üzerinde PSBLAS tabanlı bir bölütleme işlemi gerçekleştirmişler ve GPU ile seriye göre 14 kata kadar hızlanma elde etmişlerdir. Singh ve diğerleri [62] yaptıkları çalışmada karakter tanıma için Otsu eşikleme yöntemini kullanmışlardır. Otsu eşiklemeyi GPU ile paralelelleştirerek GeForce 9500 GT ekran kartında uygulamalarını çalıştırmışlardır. Hem eşikleme işlemi hem de Otsu eşik değerini hesaplamak için iki farklı kernel yazmışlar ve 32 çekirdekli ekran kartında 1.6 kat hızlanma elde etmişlerdir. Hızlanmanın az olmasının nedeni ekran kartındaki çekirdek sayısının azlığıdır. Güçlü bir ekran kartında çalıştırılması durumunda hızlanma oranı artacaktır. Jin ve diğerleri [63] çalışmalarında kenar ve bölge tabanlı bölütleme işlemini GPU ile gerçekleştirmişlerdir. GPU uygulamasında blokları bir, iki ve üç boyutlu olarak planlamışlardır. İş parçacıklarının nasıl dağıtıldığı verilmekte ancak hızlanma oranı veya işlem süresi verilmemektedir. Wang ve Chen [64] çalışmalarında bölütleme için en kısa yol algoritmasını GPU ile paralelleştirmişlerdir. Algoritma medikal veriler üzerinde uygulanmış ve seriye göre 8 kat hızlanma elde edilmiştir. Smistad ve diğerleri [65] yaptıkları çalışmada medical görüntülerde görüntü bölütleme işleminde GPU kullanılan çalışmaları incelemişlerdir. Bu inceleme sonucunda farklı bölütleme algoritmaları veri paralelliği, iş parçacığı sayısı, dallanma, bellek kullanımı ve senkronizasyon konusunda derecelendirilmiştir. Bu verilerden elde edilen sonuçlara göre algoritlamaların GPU için uygunluğu belirlenmiştir. Buna göre bölütleme algoritmalarından eşikleme, morfoloji, aktif kontur çıkarma, Markov rasgele alan ve merkez çizgi çıkarma algoritmalarının üst seviyede paralelliğe uygun olduğu belirtilmiştir. Buna göre bu algoritmalardaki uygulamalarda seriye göre kazanç yüksek olacaktır.

2016 yılında D'Ambra ve Pilippone [66] yaptıkları çalışmada görüntü bölütleme için Jacobi ve Gauss-Seidel algoritmalarını kullanmışlardır. Paralelleştirme için ise PSBLAS kütüphanesini kullanmışlardır. GPU ile seriye göre 14 kata kadar hızlanma elde etmişlerdir. Diaz-Pernil ve diğerleri [67] ise retina içi görüntüleri üzerinde otomatik bölütlemeyi GPU kullanarak gerçekleştirmişlerdir. DRIVE ve DIARETDB1 veritabanından 129 retina içi görüntüsü kullanarak sonuçları değerlendirmişlerdir. Otomatik bölütleme algoritmasında Sobel operatörü, ikili bölütleme ve Hough algoritmasını kullanmışlardır.

%99,6 doğruluk oranına ulaşmışlardır. Hızlanmaları konusunda herhangi bir oran belirtilmemiş sadece CPU'dan daha iyi sonuç alındığı söylenmiştir. Başka bir çalışmada Jaros ve diğerleri [68] k-means bölütleme algoritmasını medikal görüntüler üzerinde CPU ve GPU ile paralelleştirmişlerdir. CPU ile paralelleştirmede OpenMP, GPU ile paralelleştirmede ise CUDA kullanmışlardır. Hızlanma oranları işlenen görüntü sayısına göre farklılık göstermekle birlikte CUDA ile elde edilen hızlanmalar görüntü sayısı arttıkça yükselmektedir.

Görüntü bölütleme görüntü işleme adımları içerisinde çok kullanılan işlemler arasındadır. Bu uygulamalar özellikle çözünürlük arttığında ve bölütleme işlemi çoklu eşikleme şeklinde olduğunda yüksek işlem zamanı gerektirmektedir [69]. Bu nedenle özellikle piksellerin bağımsız olarak işlendiği algoritmalar için GPU paralelleştirmesi ve dolayısıyla CUDA kullanılması ile yüksek oranda hızlanma sağlanabilecektir. Yapılan çalışmaların bir kısmında da yüksek oranlarda paralelleştirme sağlandığı görülmektedir. Bundan sonra yapılacak çalışmalarda da CUDA kullanımı bölütleme algoritmalarında hızlanma sağlanması açısından önemlidir. Algoritmaların mümkün olduğu kadar komşu piksellerden bağımsız çalışması paralelleştirmenin daha kolaylaşmasını sağlasa da iteratif ve bir adımın sonucunun diğer adımın girdisi olarak kullanıldığı algoritmalarda GPU üzerinde paralelleştirmeyi zorlaştırdığı ve yeterli hızlanmalara ulaşılamayacağı söylenebilir. Yapılan çalışmalarda da hızlanma oranının sınırlı kaldığı algoritmalar bulunmaktadır. Görüntü bölütlemede birbirinden farklı pek çok yöntem kullanılmaktadır. Yapılan çalışmalarda da bu çeşitlilik göze çarpmaktadır. Bölütleme algoritmalarından bir kısmının (örneğin; eşikleme, morfoloji, aktif kontur çıkarma, Markov rasgele alan ve merkez çizgi çıkarma) GPU ile paralelleştirmeye daha uygun olduğu [65] ve CUDA uygulamaları sonucu bu algoritmalarda elde edilecek hızlanmaların daha yüksek olacağı söylenebilir. Kalman filtresi, istatistik kenar modeli, Ridge traversal gibi algoritmaların ise GPU kullanımı için uygunluk derecesinin daha düşük olduğu görülmektedir.

4.4. Görüntü Çakıştırma (Image Registration)

Görüntü çakıştırma farklı kaynaklardan, farklı açı veya farklı zamanlarda elde edilen görüntüler içerisindeki ortak alt görüntülerin her noktası geometrik olarak üst üste gelecek şekilde hizalanması olarak tanımlanır. Görüntü çakıştırmada öteleme, döndürme ve boyutlandırma işlemleri gerçekleştirilir. Bu konuda kullanılan pek çok algoritma bulunmaktadır. Görüntü çakıştırma işlemlerinin CUDA uygulamaları bu bölümde incelenmiştir.

Görüntü çakıştırma konusunda yapılan ilk çalışma 2008 yılında Muyan-Özçelik ve diğerlerinin [70] yaptığı GPU hızlandırma uygulamasıdır. Çalışmada Demons algoritması ile 3B CT akciğer görüntüleri üzerinde görüntü çakıştırma işlemleri CUDA kullanılarak paralelleştirilmiş ve 55 kata kadar hızlanma elde edilmiştir. Demons algoritmasının GPU ile paralelleştirildiği başka bir çalışma da 2010 yılında Gu ve diğerlerinin [71] uygulamasıdır. [68] gibi CT görüntüleri üzerinde Demons algoritmasını CUDA ile hızlandırmışlardır. CPU'ya göre 100 kata kadar hızlanma elde etmişlerdir.

Huang ve diğerleri [72] 2011 yılında yaptıkları çalışmada 3B MR görüntülerinde görüntü çakıştırma için popüler bir yöntem olan istatistiksel parametre haritalama yöntemini CUDA kullanarak paralelleştirmişlerdir. 643, 1283, 2563 boyutlarında görüntülerde uygulama gerçekleştirilmiştir. Çalışmada 14 kata kadar hızlanma sağlanmıştır. Aynı yıl yayınlanan başka bir çalışmada Ruijters ve diğerleri [73] görüntü çakıştırmada B-spline tabanlı esnek bir yöntem kullanmışlardır. İç cerrahide bölgesel hareketli biyomedikal görüntüler esnek görüntü çakıştırma kullanılarak telafi edilebilir. Çalışmada paralel işlemci gücü kullanılarak esnek çakıştırma işlemi GPU ile hızlandırılmaktadır. Bunun için benzerlik ölçümü ve türevlerin hesaplanması kullanılmaktadır. 3B görüntüler üzerinde GPU ile 50 kat hızlanma sağlanmıştır. Gerçek zamanlı bir uygulamada Park ve diğerleri [74] GPU kullanmışlardır. İteratif en yakın nokta ve iteratif izdüşüm noktası tekniklerini kullanarak CPU işlem zamanını ölçmektedirler. İteratif izdüşüm noktası tekniğini CUDA kullanarak paralelleştirmişler ve CPU ile GPU işlem zamanını karşılaştırmışlardır. Aynı tekniğin CUDA uygulamasında yaklaşık 10 kat hızlanma sağlanmıştır.

2012'de Osama ve diğerleri [75] 2B ve 3B röntgen görüntüleri üzerinde CUDA kullanarak çakıştırma işlemini paralelleştirmişlerdir. Çakıştırmada görüntüleri 0 ile 20 derece arasında çevirerek düzeltmektedirler. Paralelleştirmede iki farklı algoritma kullanmışlardır. İlk algoritmada çevirme işlemi GPU'da gerçekleştirilirken, benzerlik ölçümü adımı CPU üzerinde çalıştırılmaktadır. İkincide ise hem çevirme işlemi hem de benzerlik ölçümü adımı GPU üzerinde çalıştırılmaktadır. 3B görüntülerde 12 kata kadar hızlanma elde edilmiştir. Sah ve diğerleri [76] yine aynı yılda yaptıkları çalışmada çakıştırma için öteleme, döndürme ve ölçeklendirme işlemlerini FFT (Fast Fourier Transform) kullanarak uygulamışlardır. CUDA ve OpenCL ile FFT kütüphanelerini kullanarak paralelleştirme gerçekleştirmişlerdir. CUDA ile 345, OpenCL ile 116 kata kadar hızlanma elde etmişlerdir. CUDA ile OpenCL'e göre daha iyi sonuç alınmasının kütüphanesinde kullanılan komutların optimizasyonunun daha iyi olmasından kaynaklandığı öne sürülmüştür. Çalışma frekans modülasyonu kullanılan örneklerdendir.

2013 yılında Lu [77] Affine dönüşümünü CUDA uygulaması ile hızlandırmış ve gerçek zamanlı uygulamalarda kullanılabileceğini belirtmiştir. 255 kata kadar hızlanma elde edilmiştir. Yine aynı yılda Marchelli ve diğerleri [78] çalışmalarında CUDA'nın Thrust kütüphanesini kullanarak görüntü çakıştırma işlemini gerçekleştirmektedirler. Uygulamalarında bellekleri etkin bir biçimde kullanmaktadırlar. Uygulamalarını farklı NVIDIA kartlarında çalıştırmışlar ve sonuçları karşılaştırmışlardır. 1 MB'lık veriyi yaklaşık 1 ms'de işlemişlerdir. Zhang ve diğerleri [79] 2014 yılında yaptıkları çalışmada görüntü çakıştırma işlemi için esinleştirme ve ayrıntılandırma (coarse-to-fine) yaklaşımı kullanmaktadırlar. Algoritmanın hızlandırılması amacıyla GPU kullanılmaktadır.

Yüksek çözünürlüklü görüntülerde algoritma çalıştırılmış ve farklı yöntemlerle olan sonuçlar verilmiştir. İşlemlerin çoğunda GPU uygulamasında CPU'ya göre hızlanma sağlanmıştır. Li ve diğerleri [80] 2015 yılında yayınlanan çalışmalarında görüntü çakıştırma işlemini GPU ile paralelleştirmişlerdir. Çalışmada 5 farklı korelasyon oranı tabanlı metot geliştirmişlerdir. Geliştirdikleri algoritmaları Fermi, Kepler ve Maxwell mimarilerinde çalıştırmışlardır. CPU uygulamasına göre 4 GPU ile 55 kata kadar hızlanma elde etmişlerdir. 145 saniye olan görüntü çakıştırma süresini 2.6 saniyeye kadar indirmişlerdir.

Görüntü çakıştırma görüntü işleme alanları içerisinde nispeten yeni olan ve halen de üzerinde çalışılan alanlardandır. Bu nedenle bu konu geliştirilmeye açıktır ve yapılabilecek çalışmalar bulunmaktadır. GPU ile paralelleştırmede farklı algoritmalar üzerinde uygulamaların yapıldığı görülmektedir. Yapılan çalışmalar algoritmaların GPU ve CUDA kullanılarak paralelleştirilmesi sonucunda çok yüksek oranlarda hızlanma elde edildiğini göstermektedir. Görüntü çakışma algoritmalarında amaç bir resim üzerinde belli bir modelin aranması olmaktadır. Seri işlemlerde aranan modelin büyük resim üzerinde aranması sırasında genellikle algoritmanın daha az zaman alması için resim belli oranlarla küçültülerek aramalar gerçekleştirilmektedir. Bu bakımdan GPU tüm piksellerde aynı işlemi eşzamanlı gerçekleştirdiğinden algoritmanın küçük iş parçalarına ayrılması kolay ve veri miktarı büyük olduğundan elde edilen hızlanmalar da yüksek olmaktadır. GPU kartlarının çekirdek sayılarının artarak güçlendiğini düşünürsek görüntü çakıştırma alanında GPU kullanımı hız açısından çok büyük avantaj sağlayacaktır.

4.5. Görüntü Sınıflandırma (Image Classification)

Sınıflandırma birçok bilim dalında bir karar verme işlemidir. Görüntü sınıflandırması görüntünün çeşitli özelliklerini analiz eder ve onları kategorilere ayırır. Görüntü sınıflandırma teknikleri kullanılırken iki farklı yaklaşım vardır [81]. Bunlardan birincisi piksel tabanlı görüntü sınıflandırma yaklaşımıdır. Bu yaklaşım eğitimli ve eğitimsiz görüntü sınıflandırma olarak ikiye ayrılmaktadır. Bu sınıflandırma yaklaşımında görüntü üzerindeki piksel değerlerinden çeşitli algoritmalar kullanılarak sınıflandırma gerçekleştirilir. Diğeri ise nesne tabanlı (object-based) görüntü analizi kullanarak yapılan sınıflandırmadır [82]. Bu yaklaşımla eldeki görüntülerin sınıflandırılması için diğer görüntü işleme teknikleri kullanılmaktadır. Komşu pksellerin gruplandırılması ve nesnelerin ölçeklerine göre ayarlanması işlem basamakları içerisinde yer alır. Sınıflandırma için genellikle diğer görüntü işleme tekniklerinden bir veya birkaçı uygulanır. Bu bölümde görüntü sınıflandırma konusundaki CUDA uygulamalarına yer verilmiştir.

Görüntü işleme üzerinde sınıflandırma yapan ilk çalışma 2011 yılında yapılmıştır. Gumbau ve diğerleri [83] sınıflandırma konusunda farklı bir uygulama yapmışlardır. Ağaçların türlerinin tespit edilmesi konusunda ağaçlardan elde edilen görüntüleri kullanmışlardır. Bu görüntülerden ağaç türleri belirlenmektedir. Algoritmanın daha hızlı çalışması için GPU kullanılmaktadır. Başka bir çalışmada ise Cesnovar ve diğerleri [84] gökyüzü görüntülerini sınıflandırmaktadırlar. Yapısal benzerlik tabanlı algoritmalarını CUDA kullanarak paralelleştirmişlerdir. Gabor filtrelerini ve hızlı Fourier dönüşümünü kullanmışlardır. Algoritmalarında paylaşımlı belleğin kullanıldığı ve kullanılmadığı algoritmalar bulunmakta ve ayrı ayrı değerlendirilmektedir. Ortalama 39 kat hızlanma elde etmişlerdir.

Haythem ve diğerleri [85] 2014 yılında frekans dönüşümünün sınıflandırmada kullanıldığı bir çalışma yapmışlardır. Çalışmalarında görüntülerde nesne tanıma için Fourier dönüşümünü kullanmışlardır. İki boyutlu renkli görüntülerde nesne tanımayı CUDA kullanarak hızlandırmışlardır. CUDA'nın cuFFT kütüphanesini kullanmışlar ve 142 kata kadar hızlanma elde etmişlerdir. Aynı yıl Ganiau ve Onchiş [86] yaptıkları çalışmada bazı algoritmaları GPU'da paralelleştirmişlerdir. Dijital görüntü analizi yapan pek çok kullanım yerinde hızlı ve doğru yüz tanıma sistemine ihtiyaç duyulmaktadır. Bu çalışmada yüz tanıma noktalarının işaretlemesinde kullanılan Gabor kareleri GPU kullanılarak paralelleştirilmiştir. Matris çarpımı, FFT, kenar yakalama, yüz tanıma algoritmaları GPU'da çalıştırılmıştır. Matris çarpımında 48 kat, FFT'da 10 kat, kenar yakalama algoritmasında 9 kat, yüz tanıma algoritmasında 5 kat hızlanma sağlanmıştır. Werff ve Bakker [87] çalışmalarında görüntü sınıflandırma için en çok kullanılan mesafe ölçme algoritmalarını kullanmışlardır. Çalışmada Öklit uzaklığı ve iki vektörün benzerliği olmak üzere iki farklı algoritma kullanılmaktadır. Bunların GPU uygulamaları çalıştırılmış ve sonuçları çalışmada verilmiştir. Verilen sonuçlarda GPU'ya veri transferinin ne kadar zaman aldığı ayrıca verilmektedir. Çalışma zamanı veri transferleri ve kernel süresi olarak ayrıca belirtilmektedir. İki vektörün benzerliği (spectral angle) metodu ile Öklit uzaklığına göre daha iyi hızlanma sağlanmıştır.

Lopez-Fandino ve diğerleri [88] 2015 yılındaki çalışmalarında aşırı öğrenme makinelerini (Extreme machine learning) [89, 90] kullanan bir sınıflandırma gerçekleştirmişlerdir. Sınıflandırmada destek vektör makinelerini kullanmışlardır. Tıbbi görüntüler üzerinde aşırı öğrenme makinelerini kullanarak sınıflandırmayı GPU ile hızlandırmışlardır. OpenMP ile çok çekirdekli işlemci ile karşılaştırıldığında 5,9 kata kadar hızlanma elde etmişlerdir. Çalışma aşırı öğrenme makinesi algoritmasının GPU ile hızlandırıldığı karşılaşılan tek çalışmadır.

Görüntü sınıflandırma konusunda elde edilen çalışmalarda farklı algoritmaların ve farklı yaklaşımların kullanıldığı görülmektedir. Çalışmaların bir kısmında ise piksel tabanlı sınıflandırma yaklaşımı ile görüntü sınıflandırma işlemi CUDA kullanılarak paralelleştirilmiştir. Bu çalışmalarda Öklit uzaklığı, iki vektörün benzerliği, destek vektör makineleri gibi çok kullanılan algoritmalar üzerinde uygulama yapıldığı görülmektedir. Bazı çalışmalarda ise sadece sınıflandırma aşaması değil görüntü

işleme sürecinin tüm aşamalarında CUDA kullanılmaktadır. Çünkü görüntü sınıflandırma her ne kadar bir görüntü işleme alanı olsa da bu adım genellikle görüntü işlemenin son adımıdır ve yapılan uygulamalarda görüntü işlemenin GPU ile paralelleştirildiği ve elde edilen sonuçlardan sınıflandırma yaparak bir yargıya varıldığı görülmektedir. Bu çalışmaların sayısı az da olsa bundan sonra yapılacak olan çalışmalara ışık tutmaktadır. Çünkü CUDA kullanımının artık gerçek uygulamalar için kullanılması ve sonuç elde edilebilecek çalışmalarda hızlanma sağlaması mümkün olabilmektedir. Uygulamalar genellikle nesne tanıma işlemlerinin farklı alanlarda ve farklı algoritmalar kullanılarak GPU paralelleştirmesi olup hızlanma oranları çalışmaya göre farklılık göstermektedir.

4.6. Diğer Uygulamalar (Other Applications)

Bu bölümde görüntü işleme konusunda GPU kullanılan CUDA uygulamalarından yukarıdaki konu başlıkları dışında yapılan çalışmalar incelenmiştir.

Shi ve diğerleri [91] 2012 yılında GPU ullanılan tıbbi görüntü işleme tekniklerini kullanan çalışmaları incelemişlerdir. Çalışmaları bölütleme, çakıştırma ve görüntüleme olarak üç gruba ayırmışlardır. Her bir grupta kullanılan algoritmaların ne tür veriler üzerinde kullanıldığını ve elde edilen sonuçları karşılaştırmışlardır.

İncelenen çalışmalarda 2014 yılında yayınlananlar şu şekildedir: Sugimoto ve diğerleri [92] çalışmalarında desen (texture) tabanlı hacim kaplama (volume rendering) metotlarından ışın izleme (ray-casting) tekniğini ön belleği kontrollü kullanarak GPU ile hızlandırmışlardır. Çünkü GPU mimarisinde işlemci ve hafıza birimlerinin kullanımında ön bellek optimizasyonu performansı arttırmada çok önemlidir. Önerilen metotta doku bellek kullanımı belirlenmiş ve iş parçacığı bloklarının genişlik ve yüksekliği dinamik olarak seçilmiştir. Elde edilen sonuçlara göre 2,2 kat hızlanma görülmüştür. Nam ve diğerleri [93] yaptıkları çalışmada makro öznitelikler kullanarak görüntü üzerinde hareket algılama ve nesne yakalama işlemi gerçekleştirmektedirler. Standart insan algılama algoritması geliştirilerek daha kısa zaman alan bir algoritma oluşturulmuştur. Bu iki algoritmanın da GPU uygulaması gerçekleştirilmiş ve hızları karşılaştırılmıştır. Önerilen algoritma hem CPU'da hem de GPU'da daha hızlı çalışmaktadır. Liu ve diğerleri [94] çalışmalarında yeni bir merkez çizgi çıkarma algoritması önermektedirler. Merkez çizgi çıkartma şekil analizi, geometri işleme, sanal endoskopi gibi farklı görüntüleme uygulamalarında önemli bir yer almaktadır. Merkez çizgi nesnelerin uzunluklarının ölçülmesini sağlar. 3B şekillerde merkez çizginin bulunması yeteri kadar hızlı gerçekleştirilememektedir. Bu nedenle CPU üzerinde çalışan seri metotların performansı sınırlıdır. Bu çalışmada GPU üzerinde çalışan yeni bir paralel merkez çizgi çıkarma algoritması uygulanmıştır. Bu yeni algoritma var olan Brute Force algoritmasından daha iyi sonuç vermiştir. [90] ve [91] çalışmalarında sadece GPU uygulaması yaparak onun sonuçlarını karşılaştırmamış kendi algoritmalarını önermişler ve bu algoritmayı GPU ile hızlandırmışlardır. Kütüphane kullanarak uygulama yapan çalışma örneklerinden biri de Zhang ve diğerlerinin [95] yaptıkları çalışmadır. Geniş veri setleri üzerinde uzaysal koordinatları toplama işlemi gerçekleştirmişlerdir. Çok çekirdekli CPU'lar ve GPU üzerinde uygulama çalıştırılmış ve elde edilen sonuçlar verilmiştir. Paralelleştirme CUDA platformunda gerçekleştirilmiş ve CUDA kütüphanelerinden olan Thrust kütüphanesi kullanılmıştır.

Galizia ve diğerleri [96] 2015 yılında yaptıkları çalışmada heterojen sistemler üzerinde çalışmışlardır. Modern yüksek başarımlı hesaplama sistemleri çoklu çekirdekli CPU, çoklu GPU'lar ve FPGA'lardan oluşan heterojen sistemlerden oluşmaktadırlar. Bu sistemlerde performans açısından iyi yazılımlar geliştirmek daha zordur. Bu çalışmanın amacı GPU kümelerinde görüntü işleme çalışmalarını yapmak amacıyla PIMA(GE)2 kütüphanesi ve Paralel Görüntü İşleme GEnoa kütüphanesini kullanmaktır. Bu kütüphane geleneksel kümeleri MPI ile, GPU'ları ise CUDA ile kullanabilmektedir. Burada ilk hedef GPU kümelerini tanımak olmaktadır. Kütüphane operasyonları seri arayüzlerle tanımlanmış ve paralel işleme kullanıcıdan gizlenmiştir. Çalışmada kütüphanenin geliştirilmesi yaklaşımı daha öne çıkmış ve tek GPU'ya göre işlem zamanı açısından %30'luk bir kazanç elde edilmiştir. Quin ve diğerleri [97] çalışmalarında video görüntülerinde arka zemin çıkarmayı GPU ile hızlandırmışlardır. Arka zemin çıkarma yöntemi ile hareket eden nesneleri tespit etmektedirler. Bunun için Gabor filtresini kullanmışlardır. Gabor filtresi kullanılarak ve kullanılmadan video görüntülerinde arka zemin çıkarma yöntemi ile hareket algılaması gerçekleştirmişlerdir. Gabor filtresi uygulanmadan 10 kat hızlanma sağlanırken, Gabor filtresi ile 14.5 kat hızlanma elde edilmiştir.

Parmak izi tanıma konusunda CUDA ile paralelleştirmede literatürde iki çalışmaya rastlanmıştır. Her iki çalışmada da parmak izi tanıma algoritması CUDA kullanarak paralelleştirilmektedir. Lastra ve diğerleri [98] algoritmalarını tek GPU ile 15 kat, birden fazla GPU kullanarak ise 54 kat hızlandırmışlardır. Cappeli ve diğerleri ise [99] GPU ile hızlandırmada saniyede 35 milyon parmak izi taramışlardır.

Stereo görüntüler üzerinde derinlik haritası çıkarma işleminin GPU uygulamasına literatürde 2016 yılında karşılaşılmıştır. Hernandez-Juarez ve diğerleri [100] derinlik haritası çıkarılmasında semi global matching algoritmasını kullanmışlar ve GPU ile daha önce yapılan çalışmalardan iyi sonuçlar almışlardır. 640X480 çözünürlükteki görüntülerde 128 seviyeli derinlik haritası çıkarmada saniyede 48 kare (frame) işleme hızına ulaşmışlardır. Algoritmalarında paylaşımlı hafıza kullanmışlardır. Stereo görüntüler üzerinde derinlik haritası çıkarırken GPU kullanan başka bir uygulamayı Wang ve diğerleri [101] yapmışlardır. 20 kata kadar hızlanma elde etmişlerdir.

Bu bölümde incelenen çalışmalar kullanılan algoritmalar ve yöntemler açısından CUDA kullanımının farklı örnekleridir. Yapılan çalışmalarda algoritmaların paralelleştirilmesi kazancın yüksek olmamasına rağmen önemlidir. Kazancın artırılması için bundan sonra yapılacak çalış-

malar için rehber niteliğinde kullanılabilecek çalışmalardır. Ayrıca Zhang ve diğerlerinin [95] 2014 ve Galizia ve diğerlerinin [96] ise 2015 yılında yaptıkları çalışmada kütüphaneler kullanılmaktadır. GPU programlamada bir algoritmanın piksel düzeyinde paralelleştirilmesi hem zor hem de uzun bir çalışma gerektiren bir iştir. Görüntü işlemede kullanılan gerçek bir uygulama için pek çok algoritma (gürültü temizleme, bulanıklaştırma, keskinleştirme, bölütleme, çakıştırma, sınıflandırma vb.) kullanmak gerekmektedir. Bu nedenle gerçek uygulamalarda GPU kullanımı uzun bir süreç gerektirmektedir.

Çizelge 2. GPU ve CUDA kullanılan paralel görüntü işleme çalışmaları özeti (Summary of parallel image processing studies using GPU and CUDA)

Görüntü İşleme Alanı	**Görüntü İşleme Tekniği**	**Çalışma**	**Maksimum hızlanma oranı**
Görüntü geriçatma	İleri ve geri izdüşüm	[22] Vazquez ve diğ., 2010	40
		[23] Noel ve diğ., 2010	1.6
		[24] Zhengz ve diğ., 2010	Belirtilmemiş
		[25] Herraiz ve diğ., 2010	72
		[29] Blas ve diğ., 2014	3
	İteratif geriçatma	[26] Xu ve diğ., 2010	Belirtilmemiş
		[28] Palenstjin ve diğ., 2011	Belirtilmemiş
	İteratif izdüşüm, hareket ve yön doğrulama	[30] Flores ve diğ., 2013	19
		[31] Flores ve diğ., 2014	4
		[32] Feng ve diğ., 2013	6
	3B geriçatma	[33] Li ve diğ., 2010	240
	FDK algoritması	[34] Okitsu ve diğ., 2010	24
		[35] Scherl ve diğ., 2012	22
	Stereo tabanlı yüzey eleman algoritması	[36] Chang ve diğ., 2011	100
	Adaftif en küçük dikdörtgen kapama yöntemi	[37] Jiang ve diğ., 2012	200
	WBP ve SIFT	[38] Aqullerio ve diğ., 2012	2
	Izgaralama algoritması	[39] Yang ve diğ., 2013	7.5
	Monte Karlo	[41] Monte ve diğ., 2013	Belirtilmemiş
	Yayılım ağırlıklı görüntüleme	[44] Cahang ve diğ., 2014	490
	3B modelleme	[45] Orts-Escolano ve diğ., 2015	180
Görüntü iyileştirme	Gürültü temizleme	[46] Huhle ve diğ., 2010	Belirtilmemiş
		[47] Sanchez ve diğ., 2013	30
	Kontrast iyileştirme	[48] Urena ve diğ., 2013	10
	Laplace keskinleştirme	[49] Ma ve diğ., 2014	6.8
	Sobel filtresi	[50] Saha ve diğ., 2016	11
Görüntü bölütleme	Bölge büyütme	[53] Pan ve diğ., 2008	2.5
	Bulanık mantık	[54] Zhuge ve diğ., 2009	14
	Markov rasgele alan metodu	[56] Sui ve diğ., 2012	10
	Gradient tabanlı kenar yakalama	[57] Pernil ve diğ., 2013	8
	Çeşitli yöntemler	[58] Ozdemir ve Altılar, 2014	6.5
		[59] Alvarado, 2014	8
		[61] D'ambra ve diğ., 2016	14
	Faz kontrast	[60] Bergen ve diğ., 2015	20
	Jacobi ve Gauss-Seidel algoritmaları	[66] D'ambra ve Pilippone, 2016	14
	Hough algoritması	[67] Diaz-Pernil ve diğ., 2016	Belirtilmemiş
	K-means algoritması	[68] Jaros ve diğ., 2017	Belirtilmemiş
Görüntü çakıştırma	Demons	[70] Ozcelik ve diğ., 2008	55
		[71] Gu ve diğ., 2010	100
	Haritalama	[72] Huang ve diğ., 2011	14
	B-spline tabanlı esnek çakıştırma	[73] Rujiters ve diğ., 2011	50
	İteratif izdüşüm	[74] Park ve diğ., 2011	10
	FFT	[76] Sah ve diğ., 2012	345
	Affine dönüşümü	[77] Lu, 2013	255
	Korelasyon oranı	[80] Li ve diğ., 2015	55
Görüntü sınıflandırma	Gabor filtresi	[84] Cesnovar ve diğ., 2011	39
	Fourier dönüşümü	[85] Haythem ve diğ., 2014	142
	Çeşitli yöntemler	[86] Ganiau ve Onchis, 2014	48
	Öklit uzaklığı	[87] Werff ve Bakker, 2014	Belirtilmemiş
	Destek vektör makinaları	[88] Lopez-Fandino ve diğ., 2015	6
Diğer	Işın izleme	[92] Sugimoto ve diğ., 2014	2.2
	Merkez çizgi çıkarma	[94] Liu ve diğ., 2014	Belirtilmemiş
	Parmak izi tanıma	[98] Lastra ve diğ., 2015	54
		[99] Cappeli ve diğ., 2014	Belirtilmemiş
	Derinlik haritası çıkarma	[100] Hernandez-Juarez ve diğ., 2016	10
		[101] Wang ve diğ., 2016	20

GPU programlamada oluşturulan kütüphaneler standart algoritmaların kütüphanelerden kullanılmasını sağlayarak bu sürecin kısalmasını sağlamaktadır. GPU programlamada CUDA platformu kullanılan ve yukarıda detaylı bir şekilde incelenen paralel görüntü işleme çalışmaları ve teknikleri Çizelge 2'de özetlenmiştir.

Çizelge 2'de görüldüğü gibi CUDA platformunda GPU kullanılarak paralel görüntü işleme ile ilgili görüntü geriçatma, iyileştirme, bölütleme, çakıştırma ve sınıflandırma alanlarında çalışmalar yapılmıştır. Bu alanlar arasında en fazla çalışmanın görüntü geriçatma alanında olduğu görülmektedir. Bu alanda yapılan çalışmalar incelendiğinde ise ileri ve geri izdüşüm tekniğinin en fazla tercih edildiği görülmektedir. Diğer alanlarda ise öne çıkan bir tekniğin olmadığı görülmektedir. Hızlanmanın en yüksek olduğu çalışma da yine görüntü geriçatma alanında olmaktadır.

5. SONUÇ (CONCLUSION)

Son yıllarda GPU programlama her alanda kullanılmaya başlamıştır. CUDA, GPU programlamada en çok kullanılan platformdur. GPU programlama her alanda kullanılmakla beraber doğası gereği veri paralelliği içeren uygulamalar için daha iyi sonuçlar vermektedir. Görüntü işleme, her piksel üzerinde aynı işlemin yapılmasından dolayı veri paralelliği içeren algoritmaların en çok kullanıldığı alanlardan biridir. Bu çalışmada görüntü işlemede CUDA platformu kullanılan GPU programlama uygulamaları incelenmiştir. CUDA platformu GPU kullanarak paralel programlama uygulaması gerçekleştirmeyi sağlamaktadır. Paralel programlama bir programın daha kısa sürede çalıştırılmasını sağlamaktadır. GPU kullanmanın asıl amacı CPU'nun performansını artırmaktır. Performansın artırılması için GPU programlamada donanım kullanımı ve hafıza yönetimi konusuna dikkat edilmesi gerekmektedir. İncelenen çalışmalarda hızlanmanın çok az olduğu çalışmalar bulunmakla birlikte 490 kata kadar hızlanma sağlanan çalışmalar da bulunmaktadır.

GPU programlamada paylaşımlı hafıza kullanımı hızlanmayı artıran önemli etkenler arasındadır. Görüntü işleme uygulamalarında hızlanmaya etki eden bir diğer unsur çözünürlük olmaktadır. Yapılan çalışmalarda görülmektedir ki GPU uygulamalarında çözünürlük arttıkça elde edilen hızlanma oranı da artmaktadır. Hızlanmanın olumsuz olarak etkilendiği noktalardan biri CPU'dan GPU'ya ve GPU'dan CPU'ya veri aktarımı için geçen süredir. Yapılan işlemin durumuna bağlı olarak veri aktarımı için geçen sürenin işlem süresine oranı çok yüksek olabilmektedir. Buna rağmen yapılan tüm çalışmalarda önemli oranlarda hızlanma elde edilmiştir. GPU kullanımı ile elde edilen sonuçlar değerlendirildiğinde son dönemlerde yapılan çalışmalarda çok yüksek oranlarda hızlanma elde edilmiştir. Bu durumun hem donanımsal hem de yazılımsal nedenleri bulunmaktadır. Donanımsal olarak son dönemlerde GPU gelişimi çok hızlı olmuştur. GPU'ların içerdiği çekirdek sayısı çok yükselmiş, her bir SM içerisindeki çift hassasiyetli kayan nokta mimarisi gelişmiş, bellek miktarları artmış ve CPU ile GPU arasındaki veri iletişimi hızlanmıştır. GPU'nun hızlanma oranı CPU'ya göre çok daha yüksek olmuştur. Yazılımsal olarak da hem CUDA platformunun yetenekleri arttırılmış, hem de açık olarak kullandırılan kütüphaneler arttırılmıştır. Bunun yanında çalışmalarda gerçekleştirilen algoritmalarda GPU'nun yetenekleri daha iyi kullanılmaya başlanmıştır. Çekirdeklere işlerin paylaştırılması ve bellek kullanımı ile uygulamalarda paralellik daha etkin bir şekilde kullanılmaktadır.

GPU programlamada algoritmaların başarılı olarak paralelleştirilmesi için GPU donanımının iyi bilinmesi gerekmektedir. Bu, donanımlarla ilgili darboğazların aşılması açısından önemlidir. Geliştirilecek algoritmaların performansının yüksek olması için, görüntüleri piksel ve bit seviyesinde işlenmesine hakim olmayı gerektirecek düzeyde uzmanlık gerekmektedir. Görüntü işleme ile ilgili araç ve kütüphanelerin gelişmesi sayesinde kullanıcılar uzman olmasalar bile GPU ve CUDA teknolojilerini verimli bir şekilde kullanabileceklerdir. Son dönemlerde CUDA kullanımını yaygınlaştırmak amacıyla üst düzey programlamayı destekleyen araçlar geliştirilmekte ve bazı kütüphaneler oluşturulmaktadır.

Görüntü işlemede incelenen alanlar içerisinde görüntü geriçatma alanı en verimli uygulamaların bulunduğu alandır. Hızlanmaların en yüksek olduğu alan olmasının temel nedeni bu alanda kullanılan algoritmaların paralelleştirmeye çok uygun olmasıdır. Hızlanmanın yüksek olduğu diğer alan ise görüntü çakıştırmadır. Bunun nedeni ise çakıştırma işlemlerinde aynı işlemin farklı veriler üzerinde tekrarının yüksek olması ve bu nedenle işlem zamanın fazlalığıdır. Bu algoritmalarda da paralelleştirme sonucu kazancın yüksek olduğu görülmektedir. İki alanda da GPU'ların donanımları daha verimli çalışmaktadır.

Gerçek zamanlı görüntü işleme uygulamalarında yüksek performanslı yöntemlere ihtiyaç duyulmaktadır. Bu kapsamda görüntü işleme için kullanılan diğer paralelleştirme tekniklerinin araştırılması ve bu konuda yapılan çalışmaların incelenmesi gelecekte yapılabilecek çalışmalardandır. GPU ile paralelleştirilen çalışmaların optimize edilerek hızlandırılması gerçek zamanlı uygulamalarda kullanılmasını sağlayacaktır. Optimize için CPU ve GPU arasında veri iletişim sürelerinin kısaltılması ve paylaşımlı hafıza kullanımının arttırılması sağlanabilir. Görüntü işlemede yaygın olarak kullanılan OpenCV kütüphanesinin CUDA fonksiyonlarının arttırılması ve optimizasyonun yapılması ile CUDA konusunda uzman olmayanların bu kütüphaneyi kullanım oranı artacaktır.

KAYNAKLAR (REFERENCES)

[1] Brodtkorb A.R., Hagen T.R., Saetra M.L., "Graphics processing unit (GPU) programming strategies and trends in GPU computing", **Journal of Parallel and Distributed Computing**, 73: 4-13, (2013)

[2] Cook S., CUDA Programming A developer's Guide to Parallel Computing with GPUs, **Elsevier**, USA, (2013).
[3] Schneider M., Fey D., Kapusi D., Macheidt T., "Performance comparison of designated preprocessing white light interferometry algorithms on emerging multi- and many-core architectures", **Procedia Computer Science**, 4:2037–2046, (2011).
[4] Ruetsch G., Oster B., Getting Started with CUDA, NVISION08: The World of Visual Computing, California USA, August 25-27, (2008).
[5] NVIDIA CUDA Compute Unified Device Architecture Programming Guide, Version 1.0, (2007)
[6] NVIDIA Accelerated Computing. GPU-Accelerated Libraries. Yayın tarihi 2017. Erişim tarihi Nisan 10, (2017).
[7] Es A., İşler V., "Accelerated regular grid traversals using extended anisotropic chessboard distance fields on a parallel stream processor", **Journal of Parallel and Distributed Computing**, 67: 1201-1217, (2007)
[8] Göddeke D., Strzodka R., Mohd-Yusof J., McCormick P., Buijssen S.H.M., Grajewski M., Turek S., "Exploring weak scalability for FEM calculations on a GPU-enhanced cluster", **Parallel Computing**, 33: 685-699, (2007)
[9] Zwart S.F.P., Belleman R.G., Geldof P.M., "High-performance direct gravitational N-body simulations on graphics processing units", **New Astronomy**, 12: 641-650, (2007)
[10] Anderson A.G., Goddard III W.A., Schröder P., "Quantum Monte Carlo on graphical processing units", **Computer Physics Communications**, 177: 298–306, (2007)
[11] Martinsen P., Blaschke J., Künnemeyer R., Jordan R., "Accelerating Monte Carlo simulations with an NVIDIA graphics processor", **Computer Physics Communications**, 180: 1983–1989, (2009).
[12] Che S., Boyer M., Meng J., Tarjan D., Sheaffer J.W., SKadron K., "A performance study of general-purpose applications on graphics processors using CUDA", **Journal of Parallel and Distributed Computing**, 68: 1370–1380, (2008)
[13] Diez D.C., Mueller H., Frangakis A.S., "Implementation and performance evaluation of reconstruction algorithms on graphics processors", **Journal of Structural Biology**, 157: 288-295, (2007)
[14] Stone S.S., Haldar J.P., Tsao S.C., Hwu W.W., Sutton B.P., Liang Z.P., "Accelerating advanced MRI reconstructions on GPUs", **Journal of Parallel and Distributed Computing**, 68: 1307-1318, (2008)
[15] Schenk O., Christen M., Burkhart H., "Algorithmic performance studies on graphics processing units", **Journal of Parallel and Distributed Computing**, 68: 1360-1369, (2008)
[16] Diez D.C., Moser D., Schoenegger A., Pruggnaller S., Frangakis A.S., "Performance evaluation of image processing algorithms on the GPU", **Journal of Structural Biology**, 164: 153–160, (2008)
[17] Belleman R.G., Bedorf J., Zwart S.F.P., "High performance direct gravitational N-body simulations on graphics processing units II: An implementation in CUDA", **New Astronomy**, 13: 103-112, (2008)
[18] Ram M.P.M., Kurfess T.R., Tucker T.M., "Least-squares fitting of analytic primitives on a GPU", **Journal of Manufacturing Systems**, 27: 130-135, (2008)
[19] Ryoo S., Rodrigues C.I., Stone S.S., Stratton J.A., Ueng S., "Program optimization carving for GPU computing", **Journal of Parallel and Distributed Computing**, 68: 1389-1401, (2008)
[20] Walsh S.D.C., Saar M.O., Bailey P., Lilja D.J., "Accelerating geoscience and engineering system simulations on graphics hardware", **Computers & Geosciences**, 35: 2353–2364, (2009)
[21] Komatitsch D., Michea D., Erlebacker G., "Porting a high-order finite-element earthquake modeling application to NVIDIA graphics cards using CUDA", **Journal of Parallel and Distributed Computing**, 69: 451-460, (2009)
[22] Vazquez F., Garzon E.M., Femandez J.J., "A matrix approach to tomographic reconstruction and its implementation on GPUs", **Journal of Structural Biology**, 170: 146–151, (2010)
[23] Noel P.B., Walczak A.M., Xu J., Corso J.J. Hoffmann K.R., Schafer S., "GPU-based cone beam computed tomography", **Computer Methods and Programs in Biomedicine**, 98: 271–277, (2010)
[24] Zheng S.Q., Branlund E., Kesthelyi B., Braunfeld M.B., Cheng Y., Sedat J.W., Agard D.A., "A distributed multi-GPU system for high speed electron microscopic tomographic reconstruction", **Ultramicroscopy**, 111: 1137–1143, (2011)
[25] Herraiz J.J., Espana S., Cal-Gonzalez J., Vaquero J.J., Desco M., Udias J.M., "Fully 3D GPU PET reconstruction", **Nuclear Instruments and Methods in Physics Research A**, 648: S169–S171, (2011)
[26] Xu W., Xu F., Jones M., Keszthelyi B., Sedat J., Agard D., Mueller K., "High-performance iterative electron tomography reconstruction with long-object compensation using graphics processing units (GPUs)", **Journal of Structural Biology**, 171: 142–153, (2010)
[27] Xu F., Xu W., Jones M., Keszthelyi B., Sedat J., Agard D., Mueller K., "On the efficiency of iterative ordered subset reconstruction algorithms for acceleration on GPUs", **Computer Methods and Programs in Biomedicine**, 98: 261–270, (2010).
[28] Palenstjn W.J., Batenburg K.J., Sijbers J., "Performance improvements for iterative electron tomography reconstruction using graphics processing units (GPUs)", **Journal of Structural Biology**, 176: 250–253, (2011)
[29] Blas J.G. Abella M., Isaila F., Carretero J., Desco M., "Surfing the optimization space of a multiple-GPU parallelimplementation of a X-ray tomography reconstruction algorithm", **The Journal of Systems and Software**, 95: 166–175, (2014)
[30] Flores L. A., Vidal V., Mayo P., Rodenas F., Verdu G., "CT Image Reconstruction Based on GPUs", **Procedia Computer Science**, 18: 1412–1420, (2013)
[31] Flores L.A., Vidal V., Mayo P., Rodenas F., Verdu G., "Parallel CT image reconstruction based on GPUs", **Radiation Physics and Chemistry**, 95: 247–250, (2014)
[32] Feng C., Yang J., Zhao D., Liu J., "CUDA accelerated method for motion correction in MR PROPELLER imaging", **Magnetic Resonance Imaging**, 31: 1390–1398, (2013)
[33] Li X., Grigorieff N., Cheng Y., "GPU-enabled FREALIGN: Accelerating single particle 3D reconstruction and refinement in Fourier space on graphics processors", **Journal of Structural Biology**, 172: 407–412, (2010)
[34] Okitsu Y., Ino F., Hagihara K., "High-performance cone beam reconstruction using CUDA compatible GPUs", **Parallel Computing**, 36: 129–141, (2010)
[35] Scherl H., Kowarschik M., Hofmann H.G., Keck B., Hornegger J., "Evaluation of state-of-the-art hardware architectures for fast cone-beam CT reconstruction", **Parallel Computing**, 38: 111–124, (2012)

[36] Chang J.Y., Park H., Park I.K., Lee K.M., Lee S.U., "GPU-friendly multi-view stereo reconstruction using surfel representation and graph cuts", **Computer Vision and Image Understanding**, 115: 620–634, (2011)

[37] Jiang N., Yang W., Duan L., Xu X., Huang C. Liu Q., "Acceleration of CT reconstruction for wheat tiller inspection based on adaptive minimum enclosing rectangle", **Computers and Electronics in Agriculture**, 85: 123–133, (2012)

[38] Agulleiro J.I., Vazquez F., Garzon E.M., Femandez J.J., "Hybrid computing:CPU+GPU co-processing and its application to tomographic reconstruction", **Ultramicroscopy**, 115: 109–114, (2012)

[39] Yang J., Feng C., Zhao D., "A CUDA-based reverse gridding algorithm for MR reconstruction", **Magnetic Resonance Imaging**, 31: 313–323, (2013)

[40] Piccialli F., Cuomo S., Michele P.D., "A regularized MRI image reconstruction based on Hessian penalty term on CPU/GPU systems", **Procedia Computer Science**, 18: 2643–2646, (2013)

[41] Monte C.F.P., Pccoli F., Luciano C., Rizzi S., Bianchini G., Scutari P.C., "Estimation of Volume Rendering Efficiency with GPU in a Parallel Distributed Environment", **Procedia Computer Science**, 18: 1402–1411, (2013)

[42] Gai J., Obeid N., Holtrop J.L., Wu X.W., Lam F., Fu M., Haldar J.P., Hwu W.W. Liang Z. Sutton B.P., "More IMPATIENT: A gridding-accelerated Toeplitz-based strategy for non-Cartesian high-resolution 3D MRI on GPUs", **Journal of Parallel and Distributed Computing**, 73: 686–697, (2013)

[43] Birk M., Balzer M., Ruiter N.V., Becker J.," Evaluation of performance and architectural efficiency of FPGAs and GPUs in the 40 and 28 nm generations for algorithms in 3D ultrasound computer tomography", **Computers and Electrical Engineering**, 40: 1171–1185, (2014)

[44] Chang L., El-Araby E., Dang V.Q., Dao L.H., "GPU acceleration of nonlinear diffusion tensor estimation using CUDA and MPI", **Neurocomputing**, 135: 328–338, (2014)

[45] Orts-Escolano S., Garcia-Rodriguez J., Serra-Perez J.A., Jimeno-Morenilla A., Garcia-Garcia A., Morell V., Cazorla M., "3D model reconstruction using neural gas accelerated on GPU", **Applied Soft Computing**, 32: 87–100, (2015)

[46] Huhle B., Schairer T., Jenke P., Straber W., "Fusion of range and color images for denoising and resolution enhancement with a non-local filter", **Computer Vision and Image Understanding**, 114: 1336–1345, (2010)

[47] Sanchez M.G., Vidal V., Bataller J., Arnal J., "A Parallel Method for Impulsive Image Noise Removal on Hybrid CPU/GPU Systems", **Procedia Computer Science**, 18: 2504–2507, (2013)

[48] Urena R., Morillas C., Pelayo F., "Real-time bio-inspired contrast enhancement on GPU", **Neurocomputing**, 121: 40–52, (2013)

[49] Ma T., Li L., Ji S., Wang X., Tian Y., Al-Dhelaan A., Al-Rodhaan., "Optimized Laplacian image sharpening algorithm based on graphic processing unit", **Physica A**, 416: 400–410, (2014)

[50] Saha D., Darji K., Patel N., Thakore D., "Procedia Computer Science, Implementation of Image Enhancement Algorithms and Recursive Ray Tracing using CUDA", **7th International Conference on Communication, Computing and Virtualization**, 79: 516–524, (2016)

[51] Aydın S. "Paralel veri işleme teknikleri kullanılarak silindirik metal nesnelerin yüzey hatalarının gerçek zamanlı olarak belirlenmesi". Gazi Üniversitesi, Bilişim Enstitüsü, **Doktora Tezi,** (2019)

[52] Uslu R., "Elektronik bir hücresel yapay sinir ağı gerçekleşmesi olan ACE16K üzerinde görüntü bölütleme", İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, **Yüksek Lisans Tezi**, (2007)

[53] Pan L., Gu L., Xu J., "Implementation of Medical Image Segmentation in CUDA", **5th International Conference on Information Technology and Application in Biomedicine**, Shenzhen, China, 30-31, Mayıs, (2008)

[54] Zhuge Y., Cao Y., Miller R.W., "GPU Accelerated Fuzzy Connected Image Segmentation by using CUDA", **31st Annual International Conference of the IEEE EMBS**, Minneapolis, Minnesota, USA, 2-6 Eylül, (2009)

[55] Abramov A., Kulvicius T., Worgotter F., Dellen B., "Real-Time Image Segmentation on a GPU", **Facing the Multicore-Challenge:Aspects of New Paradigms and Technologies in Parallel Computing**, 6310: 131-142, (2010)

[56] Sui H., Peng F., Xu C., Sun K., Gong J., "GPU-accelerated MRF segmentation algorithm for SAR images", **Computers & Geosciences**, 43: 159–166, (2012)

[57] Diaz-Pernil D., Berciano A., Pena-Cantillana F., Gutierrez-Naranjo M.A., "Segmenting images with gradient-based edge detection using Membrane Computing", **Pattern Recognition Letters,** 34: 846–855, (2013)

[58] Özdemir A., Altılar T., "GPU Based Parallel Image Processing for Plant Growth Analysis", **Third International Conference on Agro-Geoinformatics**, 315-320, (2014)

[59] Alvarado R., Tapia J.J., Rolon J.C., "Medical image segmentation with deformable models on graphics processing units", **Journal of Supercomputing**, 68:, 339-364, (2014)

[60] Bergen R.V., Lin H., Alexander M.E., Bidinosti C.P., "4D MR phase and magnitude segmentations with GPU parallel computing", **Magnetic Resonance Imaging**, 33: 134–145, (2015)

[61] D'Ambra P., Pilippone S., "A parallel generalized relaxation method for high-performance image segmentation on GPUs", **Journal of Computational and Applied Mathematics**, 293: 35-44, (2016)

[62] Singh B.M., Sharma R., Mittal A., Ghosh D., "Parallel Implementation of Otsu's Binarization Approach on GPU", **Internatinal Journal of Computer Applications**, 32(2): 16-21, (2011)

[63] Jin X., Kang D.J., Jeong M., "GPU-Based Real-Time Range Image Segmentation", **Intelligent Computing Methodologies**, 8589: 293-297, (2014)

[64] Wang J., Chen W., "MRI Image Segmentation Based on a GPU Shortest Path Algorithm", **2nd International Conference on Intelligent Computing and Cognitive Informatics**, 1-4, (2015)

[65] Smistad E., Thomas L.F., Bozorgi M., Elster A.C., Lindseth F., "Medical Image Segmentation on GPUs- A Comprensive Reiview", **Medical Image Analysis**, 20(1): 1-18, (2015)

[66] D'Ambra P., Pilippone S., "A parallel generalized relaxation method for high-performance image segmentation on GPUs", **Journal of Computational and Applied Mathematics**, 293: 35-44, (2016)

[67] Diaz-Pernil D., Fondon I., Pena-Cantillana F., Gutierrez-Naranjo M.A., "Fully automatized parallel segmentation of the optic disc in retinal fundus images", **Pattern Recognition Letters**, 83 (1): 97-107, (2016)

[68] Jaros M., Strakos P., Karasek T., Riha L., Vasatova A., Jarosova M., Kozubek T., "Implementation of K-means segmentation algorithm on Intel Xeon Phi and GPU:

Application in medical imaging", **Advances in Engineering Software**, 103: 21-28, (2017)

[69] Aydin S., Samet R., Bay O.F., "Real-time parallel image processing applications on multicore CPUs with OpenMP and GPGPU with CUDA", **The Journal of Supercomputing**, 74(6): 2255-2275, (2018)

[70] Muyan-Özçelik P., Samant J.X.S.S., "Fast Deformable Registration on the GPU:A CUDA Implementation of Demons", **International Conference on Computational Sciences and Its Applications**, 223-233, (2008)

[71] Gu X., Pan H., Liang Y., Castillo R., Yang D., Choi D., Castillo E., Majumdar A., Guerrero T., Jiang S.B., "Implementation and evaluation of various demens deformeble image registration algorithms on a GPU", **Physics in Medicine and Biology**, 55: 207:219, (2010)

[72] Huang T.Y., Tang Y., Ju S., "Accelerating image registration of MRI by GPU-based parallel computation", **Magnetic Resonance Imaging**, 29: 712–716, (2011)

[73] Ruijters D., Romeny B.M.H., Suetens P., "GPU-accelerated elastic 3D image registration for intra-surgical applications", **Computer Methods and Programs in Biomedicine**, 103: 104–112, (2011)

[74] Park S., Choi S., Kim J., Chae J.S., "Real-time 3D registration using GPU", **Machine Vision and Applications**, 22: 837-850, (2011)

[75] Dorgham O.M., Laycock S.D., Fisher M.H., "GPU Accelerated Generation of Digitally Reconstructed Radiographs for 2D/3D Image Registration", **IEEE Transactions on Biomedical Engineering**, 59 (9): 2594-2603, (2012)

[76] Sah S., Vanek J., Roh Y., Wasnik R., "GPU Accelerated Real Time Rotation, Scale and Translation Invariant Image Registration Method", **Image Analysis and Recognition**, 7324: 224-233, (2012)

[77] Lu M., "3D Medical Images Registration Based on GPU Parallel Computing", **Applied Mechanics and Materials**, 241-244: 3010-3013, (2012)

[78] Marchelli G., Haynor D., Ledoux W., Tsai R., Storti D., "A Flexible toolkit for rapid GPU-based generation of DDRs for 2D-3D registration", **Medical Imaging 2013: Image Processing**, 8669, (2013)

[79] Zhang Y., Zhou P., Ren Y., Zou Z., "GPU-accelerated large-size VHR images registration via coarse-to-fine matching", **Computers & Geosciences**, 66: 54–65, (2014)

[80] Li A., Kumar A., Ha Y., Corporaal H., "Correlation ratio based volume image registration on GPUs", **Microprocessors and Microsystems**, 39 (8): 998-1011, (2015)

[81] GISGeography. Image Classification Techniques in Remote Sensing. Yayın tarihi January 22, 2017. Erişim tarihi Nisan 5, (2017)

[82] BEÜ Mühendislik Fakültesi. Görüntü Sınıflandırma. Yayın tarihi 2016. Erişim tarihi Nisan 2, (2017)

[83] Gumbau J., Chover M., Remolar I., Rebollo C., "View-dependent pruning for real-time rendering of trees", **Computers & Graphics**, 35: 364–374, (2011)

[84] Cesnovar R., Risojevic V., Babic Z., Dobravec T., Bulic P., "A GPU implementation of a structural-similarity-based aerial-image classification", **J. Supercomputing**, 65: 978-996, (2013)

[85] Haythem B., Mohamed H., Marwa C., Fatma S., Mohamed A., "Fast Generalized Fourier Descriptor for object recognition of image using CUDA", **World Symposium on Computer Applications and Research (WSCAR)**, : Sousse, TUNISIA, JAN 18-20, (2014)

[86] Ganianu M., Onchiş D.M., "Face and marker detection using Gabor frames on GPUs", **Signal Processing**, 96: 90–93, (2014)

[87] Werff H.M.A., Bakker W.H., "Implementation and performance of a general purpose graphicsprocessing unit in hyperspectral image analysis", **International Journal of Applied Earth Observation and Geoinformation,** 26: 312–321, (2014)

[88] Lopez-Fandino J., Quesada-Barriuso P., Heras D.B., Argüello F., "Efficient ELM-Based Techniques for the Classification of Hyperspectral Remote Sensing Images on Commodity GPUs", **IEEE Journal Of Selected Topıcs In Applıed Earth Observatıons And Remote Sensıng**, 8 (6): 2884-2893, (2015)

[89] Alçin Ö.F., Şengür A., İnce M.C., "İleri-geri takip algoritması tabanlı seyrek aşırı öğrenme makinesi", **Journal of the Faculty of Engineering and Architecture of Gazi University**, 30 (1): 111-117, (2015)

[90] Kaya Y., Tekin R., "Epileptik Nöbetlerin Tespiti için Aşırı Öğrenme Makinesi Tabanlı Uzman Bir Sistem", **Bilişim Teknolojileri Dergisi**, 5 (2): 33-39, (2012)

[91] Shi L., Liu W., Zhang H., Xie Y., Wang D., "A survey of GPU-based medical image computing techniques", **Quant Imaging Med Surg**, 2(3): 188-206, (2012)

[92] Sugimoto Y., Ino F., Hagihara K., "Improving cache locality for GPU-based volume rendering", **Parallel Computing**, 40: 59–69, (2014)

[93] Nam W., Han B., Han J.H., "Macrofeature layout selection for pedestrian localization and its acceleration using GPU", **Computer Vision and Image Understanding**, 120: 46–58, (2014)

[94] Liu B., Telea A.C., Roerdink J.B.T.M., Clapworthy G.J., Williams D., Yang P., Dong F., Codreanu V., Chiarini A., "Parallel centerline extraction on the GPU", **Computers & Graphics**, 41: 72–83, (2014)

[95] Zhang J., You S., Greunwald L., "Parallel online spatial and temporal aggregations on multi-core CPUs and many-core GPUs", **Information Systems**, 44: 134–154, (2014)

[96] Galizia A., D'Agostino D., Clematis A., "An MPI–CUDA library for image processing on HPC architectures", **Journal of Computational and Applied Mathematics**, 273: 414–427, (2015)

[97] Qin L., Sheng B., Lin W., Wu W., Shen R., "GPU-Accelerated Video Background Subtraction Using Gabor Detector", **J. Vis. Commun. Image R.**, 32: 1-9, (2015)

[98] Lastra M., Carabano J., Gutierrez P.D., Benitez J.M., Herrera F., "Fast fingerprint identification using GPUs", **Information Sciences**, 301: 195-214, (2015)

[99] Cappelli R., Ferrara M., Maltoni D., "Large-scale fingerprint identification on GPU , **Information Sciences**, 306: 1-20, (2015)

[100] Hermandez-Juarez D., Chacon A., Espinosa A., Vazquez D., Moure J.C., Lopez M., "Embedded real-time stereo estimation via Semi-Global Matching on the GPU", Procedia Computer Science, ICCS 2016. **The International Conference on Computational Science**, , 80, 143–153, (2016)

[101] Wang H., Zhang N., Creput J., Ruichek Y., Moreau J., "Massively parallel GPU computing for fast stereo correspondence algorithms", **Journal of Systems Architecture**, 65: 46-58, (2016)