GPGPU YÖNTEMİ İLE FOTOGRAMETRİK UYGULAMALAR

H. Şahin [a, *], S. Külür [b]

[a] Harita Genel Komutanlığı, 06100 Dikimevi, Ankara, Türkiye – hakan.sahin@hgk.msb.gov.tr
[b] İTÜ, İnşaat Fakültesi, 80626 Maslak, İstanbul, Türkiye - kulur@itu.edu.tr

ANAHTAR KELİMELER: GPGPU, CUDA, Akış İşleme, Programlama, Ortorektifikasyon, Görüş Analizi, Görüntü Filtreleme.

ÖZET:

Bilgisayarlar içerisinde yer alan grafik kartları üzerinde bulunan grafik işlemci birimleri (Graphic Processing Units – GPU) on sene öncesine göre, günümüzde son derece gelişme göstermişlerdir. Bu gelişme GPU'ların performanslarının ve yeteneklerinin artışı doğrultusunda olmuştur. Modern GPU'lar sadece çok güçlü grafik motoru olmaktan çıkmışlar ve bilgisayar işlemcilerine (Central Processing Unit-CPU) göre aritmetik işlem yapabilme hızı ve hafıza band genişliği hızı çok daha yüksek olan ve üst seviyede paralel programlanabilir işlemciler halini almışlardır. GPU'ların programlanabilirliğindeki ve yeteneklerindeki hızlı gelişme, yüksek seviyede hesap yapma ihtiyacı olan karmaşık problemlerle uğraşan araştırmacıların ilgisini çekmiştir. Bu ilgi "grafik işlemci birimi üzerinde genel amaçlı hesaplama (General Purpose Computation on Graphic Processing Units - GPGPU)" ve "akış işleme (stream processing)" kavramlarını ortaya çıkarmıştır. Grafik işlemcilerin bilgisayar işlemcilerine bir alternatif olarak gündeme gelmesinin asıl nedeni çok güçlü ve bunun yanında ucuz temin edilebilir donanım olmalarıdır. Bu grafik çipler, sabit uygulama donanımları olmaktan çıkarak günümüzde modern, güçlü ve programlanabilir genel ihtiyaçları karşılayabilecek işlemcilere dönüşmüşlerdir. Özellikle son yıllarda grafik işlemci birimlerinin genel amaçlı hesaplamalarda da kullanılabileceği olgusu araştırmacılar ve geliştirmecileri bu noktaya yöneltmiştir. Buradaki en büyük problem, grafik işlemci birimlerinin mevcut programlama yöntemlerinden farklı bir programlama modelini kullanıyor olmasıdır. Bu nedenle etkili bir GPU programlama, mevcut program algoritmasının, donanımın yapısını ve sınırlamalarını da dikkate alan grafik terimlerini kullanarak tekrardan yazılmasını gerektirmektedir. Halen çok çekirdekli işlemcilerin programlanabilmesi, geleneksel programlama yöntemleriyle gerçekleştirilememektedir. Tipik olay yordamlı programlama yönteminin birden fazla çekirdekli işlemcilerin programlanması için kullanılması mümkün olamamaktadır.

GPU'lar özellikle aynı hesaplama işlem adımının birçok veri elemanına özellikle yüksek aritmetik doğrulukla uygulanmasının gerektiği durumlarda çok etkili ve hızlı sonuçlar ortaya koymaktadırlar. Böylelikle yapılan hesaplama işleminin daha hızlı ve doğru olması sağlanmaktadır. Bilgisayar CPU'ları bir akış kontrolü içerisinde ve belli bir sıra ile her seferinde sadece tek bir hesaplama yaptıkları için GPU'lar ile kıyaslandığında daha yavaş işlem yapmaktadırlar. Bu yapı bilgisayar teknolojisinin kullanıldığı çok çeşitli uygulamalar için değerlendirilebilmektedir.

Yapılan bu çalışma içerisinde, GPGPU yönteminin çeşitli fotogrametrik uygulamalar amacıyla nasıl kullanılabileceği değerlendirilmiştir. Buna yönelik olarak öncelikle çeşitli boyutlardaki sayısal yükseklik modelleri üzerinde görüş analizi algoritması geliştirilerek GPGPU yöntemiyle CUDA (Compute Unified Device Architecture) programlama dili kullanılarak bir program yazılmış ve programın çalıştırılması sonucunda elde edilen sonuçlar ortaya konulmuştur. Sonuçlar değerlendirildiğinde CPU ile kıyaslandığında, GPU'ların ortalama 80 kat daha hızlı sonuç ürettiği görülmüştür. Bir başka uygulama olarak da ortorektifikasyon için kullanılan yöntemlerden birisi olan projektif rektifikasyon, GPGPU yöntemiyle CUDA programlama dili kullanılarak kodlanmış, programın çeşitli boyutlardaki örnek görüntüler üzerinde nasıl sonuçlar verdiği karşılaştırılmalı olarak değerlendirilmiştir. Yapılan bir başka uygulamada ise; fotogrametri içerisinde kullanılan çeşitli görüntü filtreleme tekniklerinin, GPGPU yöntemiyle CUDA programlama diliyle kodlanarak, programların ve kullanılan tekniğin nasıl sonuçlar ürettiği karşılaştırılmalı olarak ortaya konulmaya çalışılmıştır. Böylelikle GPGPU yönteminin özellikle çok yoğun veri üzerinde aynı işlemlerin yapılmasının gerektirdiği durumlarda kullanılabileceği ve böylelikle sonuca çok daha hızlı ulaşılabileceği ortaya konulmaya çalışılmıştır.

* Yazışmadan sorumlu yazar. Birden fazla yazar olduğu iletişim kurulacak uygun kişiyi bilmek için faydalıdır.

PHOTOGRAMMETRIC APPLICATIONS BY USING GPGPU METHOD

H. Sahin [a, *], S. Kulur [b]

[a] General Command of Mapping, 06100 Dikimevi, Ankara, Turkey – hakan.sahin@hgk.msb.gov.tr
[b] ITU, Civil Engineering Faculty, 80626 Maslak, Istanbul, Turkey - kulur@itu.edu.tr

KEYWORDS: GPGPU, CUDA, Stream Processing, Programming, Orthorectification, Line of Sight, Image Filtering.

ABSTRACT:

The graphic processing units (GPU) on the graphic cards used inside of the computers are really developed today compared to the ones of last decade. The development was the increase of the GPUs' performance and capabilities. The modern GPUs are not only powerful graphic engines but also they are high level parallel programmable processors with very fast computing capabilities and high memory bandwidth speed compared to central processing units (CPU). The rapid development of GPUs programmability and capabilities attracted the attentions of researchers dealing with complex problems which need high level calculations. This interest has revealed the concepts of "General Purpose Computation on Graphics Processing Units (GPGPU)" and "stream processing". The graphic processors are powerful hardware which are really cheap and affordable. So the graphic processors became an alternative to computer processors. The graphic chips which were standard application hardware have been transformed into modern, powerful and programmable processors to meet the overall needs. Especially in recent years, the phenomenon of the usage of graphics processing units in general purpose computation have led the researchers and developers to this point. The biggest problem is that the graphics processing units use different programming models unlike current programming methods. Therefore, an efficient GPU programming requires re-coding of the current program algorithm by considering the limitations and the structure of the graphics hardware. Currently, multi-core processors can not be programmed by using traditional programming methods. Event procedure programming method can not be used for programming the multi-core processors.

GPUs are especially effective in finding solution for repetition of the computing steps for many data elements when high accuracy is needed. Thus, it provides the computing process more quickly and accurately. Compared to the GPUs, CPUs which perform just one computing in a time according to the flow control are slower in performance. This structure can be evaluated for various applications of computer technology.

In this study the use of GPGPU method is evaluated for various photogrammetric applications. The focus is given primarily on the analyses made by using digital elevation models of various sizes. A line of sight algorithm is coded by using GPGPU method and CUDA (Compute Unified Device Architecture) programming language. Results provided by this method were compared 80 times faster than the results of CPU programming. In the other application the projective rectification, one of the methods used for orthorectification, is coded by using GPGPU CUDA programming language. Sample images of various sizes, as compared to the results of the program were evaluated. In another application, various image filtering techniques used in photogrammetry, were coded by using GPGPU CUDA programming language. The programs and how the results are produced by this technique were analyzed comparatively. GPGPU method can be used especially in repetition of same computations on highly dense data, thus finding the solution quickly.

1. GİRİŞ

Bilgisayarların ayrılmaz bir parçası olarak kullanılan ve bilgisayar anakartları üzerine takılan grafik kartlarının üzerinde bulunan grafik işlemci birimleri (Graphic Processing Units – GPU) bir on sene öncesi dikkate alındığında, günümüzde son derece gelişme göstermişlerdir. Bu gelişme GPU'ların hızlarının, performanslarının ve yeteneklerinin artışı doğrultusunda olmuştur. Modern GPU'lar sadece çok güçlü grafik motoru olmaktan çıkmışlar ve bilgisayar işlemcilerine (Central Processing Unit-CPU) göre aritmetik işlem yapabilme hızı ve hafıza band genişliği hızı çok daha yüksek olan ve üst seviyede paralel programlanabilir işlemciler halini almışlardır. GPU'ların programlanabilirliğindeki ve yeteneklerindeki hızlı gelişme, yüksek seviyede hesap yapma ihtiyacı olan karmaşık problemlerle uğraşan araştırmacıların ilgisini çekmiştir. Bu ilgi "grafik işlemci birimi üzerinde genel amaçlı hesaplama (General Purpose Computation on Graphic Processing Units - GPGPU)" ve "akış işleme (stream processing)" kavramlarını ortaya çıkarmıştır.

* Corresponding author. This is useful to know for communication with the appropriate person in cases with more than one author

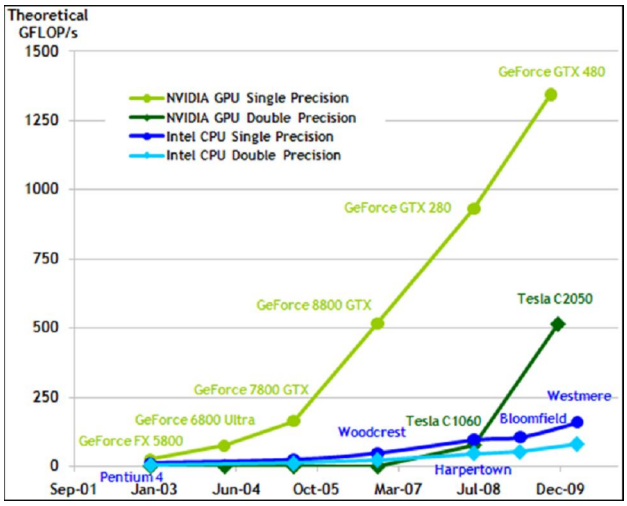
2. KULLANILAN YÖNTEM VE TANIMLAMALAR

2.1 GPGPU ve Akış İşleme

Grafik işlemcilerin gündeme gelmesinin asıl nedeni çok güçlü ve bunun yanında ucuz temin edilebilir donanım olmalarıdır. Bu grafik çipler, sabit uygulama donanımları olmaktan çıkarak günümüzde modern, güçlü ve programlanabilir genel ihtiyaçları karşılayabilecek işlemcilere dönüşmüşlerdir. Özellikle son yıllarda grafik işlemci birimlerinin genel amaçlı hesaplamalarda kullanılabileceği olgusu araştırmacılar ve geliştirmecilerin dikkatini çekmiştir. Buradaki en büyük problem, grafik işlemci birimlerinin mevcut programlama yöntemlerinden farklı bir programlama modelini kullanıyor olmasıdır. Bu nedenle etkili bir GPU programlama, mevcut program algoritmasının, donanımın yapısını ve sınırlamalarını da dikkate alan grafik terimlerini kullanarak tekrardan yazılmasını gerektirmektedir. Mevcut çift çekirdekli işlemcilerin programlanabilmesi, geleneksel programlama yöntemleriyle gerçekleştirilememekte ve tipik olay yordamlı programlama yönteminin birden fazla çekirdekli işlemcilerin programlanması için kullanılması mümkün olamamaktadır.

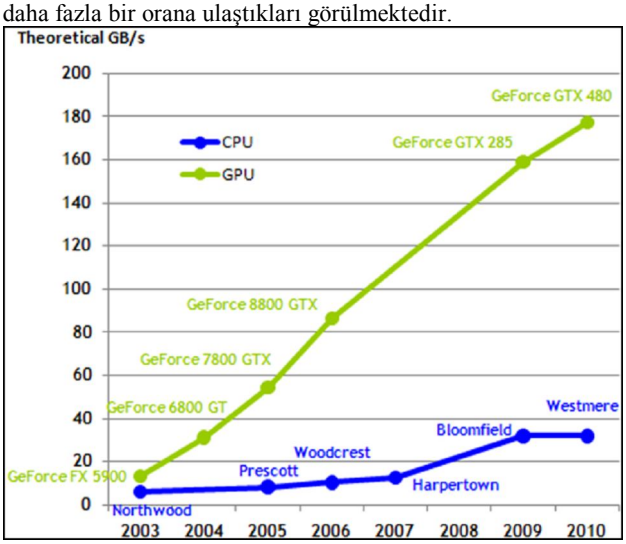
Programlama modeli akış hesaplama (stream computing-processing) terimiyle değişmiştir. Bu model içerisinde, akış içerisindeki her bir elemana uygulanan yoğun hesaplama işlemlerini (kernel functions-çekirdek fonksiyonları) tanımlamak için girdi verileri ve çıktı verileri birer akış olarak nitelendirilmektedir. Grafik kartları üzerinde ise bu akışları hesaplayan ve işleyen çok sayıda işlemci bulunmaktadır. Örneğin günümüzde kullanılan grafik kartlarından birisi olan Nvidia GTX280 serisi grafik kartı üzerinde 240 adet akış işlemcisi bulunmaktadır. Bu da yan yana sıralanmış, birlikte işlem yapabilme kapasitesi olan 240 adet bilgisayar gibi düşünülebilir. Bu akış işlemcileri sayesinde grafik kartları aynı anda birden fazla yoğun işlemi yapabilmektedirler.

GPU'lar CPU'lara göre çok daha fazla paralel hesap yapabilmektedirler. Bu durum Şekil 1'de bir grafikle gösterilmektedir. Burada işlemci hızını "flop" kavramı belirlemektedir. Flop; "saniye başına kayan nokta işlemi" anlamına gelmektedir. Grafik incelendiğinde Nvidia firmasının grafik işlemcilerinin, Intel firmasının bilgisayar işlemcilerine oranla 2009 yılındaki değerlere göre ortalama on kat daha hızlı olduğu görülmektedir.



Şekil 1. Saniyedeki kayan nokta işlem miktarı bakımından GPU ve CPU'ların yıllara göre gelişimi (Nvidia, 2010a).

Hafıza band genişliği kavramı, ekran kartının işlemcisi ile hafıza arasında saniyede aktarılabilen toplam veri miktarı boyutu anlamına gelmektedir. Hafıza veriyolu genişliğinin byte cinsinden değeri ile efektif frekansın çarpılmış hali olarak ifade edilir. Bellekle grafik işlemcisinin haberleşmesinin hızlı olması da grafik kartının performansını artıran bir etkendir. Saniye başına kayan nokta işlem kapasitesinin yıllar dikkate alındığında artış göstermesi paralelinde, hafıza band genişliğinde de bir gelişme olmuştur. Şekil 2 incelendiğinde GPU'ların hafıza band genişliklerinin, CPU'lara oranla 6 kat daha fazla bir orana ulaştıkları görülmektedir.



Şekil 2. GPU ve CPU'ların band genişlikleri (Nvidia, 2010a).

CPU ve GPU'ların kayan nokta işlem kapasiteleri arasındaki farklılığın arkasında yatan asıl neden, GPU'ların yüksek kapasitede hassas paralel hesap yapma ihtiyacı olan grafik renderlama (doku kaplama) işi için özel olarak tasarlanmış olmalarıdır. Bu tasarımdaki farklılık Şekil 3'de gösterilmeye çalışılmıştır. Şekle bakıldığında, GPU'ların CPU'lardan farklı olarak; veriyi önbellekte tutmak ve bir akış kontrolü yapmaktan çok sadece veriyi işlemek için tasarlanmış daha fazla sayıda transistöre sahip olduğu görülmektedir. Böylelikle GPU'ların CPU'lardan farklı olarak birçok aritmetik işlemlerle dolu paralel hesaplamaları yapmak için tasarlandığı anlaşılmaktadır. GPU'lar veri dizinlerini, akış kontrolü yerine, çeşitli sıralı hesaplama iş parçaları şeklinde işlerler.

Şekil 3. CPU ve GPU'ların genel yapısı ve sahip oldukları transistör sayısındaki farklılık (Nvidia, 2010a).

GPU'lar özellikle aynı hesaplama işlem adımının birçok veri elemanına özellikle yüksek aritmetik doğrulukla uygulanmasının gerektiği durumlarda çok etkili sonuçlar ortaya

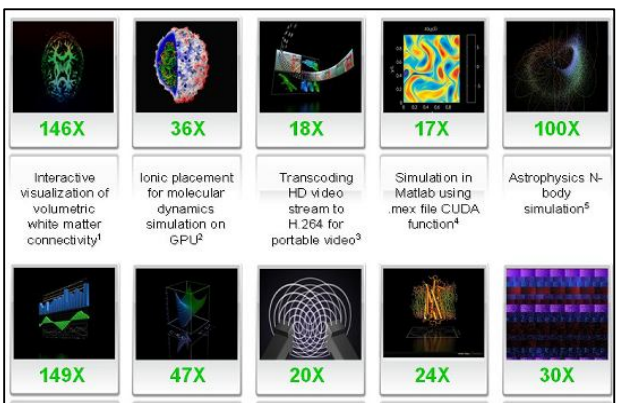
koymaktadır. Bunun en önemli nedeni aynı programın her bir veri elemanı için çalıştırılması ve akış kontrolüne çok az ihtiyaç duyulmasıdır. Çok sayıdaki işlemci, yani transistörlerin yardımıyla küçük veri kümelerinin her biri için program çalıştırılarak hesap yapılmakta, sonuçta da daha yüksek aritmetik doğruluk elde edilerek çok büyük verinin önbellekte tutulmasının önüne geçilip, hafızaya ulaşmadaki gecikmeler de önlenerek işlem süresinden kazanılmaktadır. Verinin paralel işlenmesinde, veri elemanları paralel iş parçalarına ayrılır. Birçok uygulama için bu yöntem işlem ve hesaplama hızını artırmak için kullanılabilir. Örneğin üç boyutlu (3B) renderlama işleminde, çok büyük boyutlardaki pikseller ve verteksler birer paralel iş parçası haline getirilir ve bu şekilde GPU'lar tarafından hesaplanır ve işlenirler (Yılmaz, 2010).

2.2 CUDA (Birleşik Hesaplama Aygıt Mimarisi – Compute Unified Device Architecture)

Uzun zaman önce geliştiriciler paralel hesaplama işleri için GPU'ları kullanmayı denemişlerdir. İlk başlardaki bu kullanım girişimleri (rasterizing ve Z-buffering gibi) çok ilkeldir ve donanım fonksiyonlarını tam anlamıyla kullanabilmek için sınırlı kalmıştır. Fakat gölgelendirme işlemleri matris hesaplamalarını hızlandırmıştır. 2003 yılında yapılan SIGGRAPH konferansında hemen hemen hiç katılım olmadan geçen ve "GPU computing" için ayrılmış "GPGPU" adında bir oturum olmuştur. Burada en iyi bilinen başlık "BrookGPU" adıyla anılan akış programlama dili olmuştur. Bu programlama dilinin yayınlanmasından önce popüler olan iki yazılım geliştirme uygulaması vardır; Direct3D ve OpenGL. Fakat bunlarla sınırlı sayıda GPU uygulamaları geliştirilebilmektedir. Sonrasında Brook projesi ile GPU'ların paralel işleyiciler olarak kullanılabilmelerini ve C diliyle programlanabilmelerini olanaklı kılmıştır. Stanford Üniversitesi tarafından geliştirilen bu proje, iki farklı grafik kartı tasarımcısı ve üreticisi olan NVIDIA ve ATI firmalarının dikkatlerini çekmiştir. Sonrasında ise Brook'u geliştiren bazı insanlar NVIDIA firmasına katılmışlar ve paralel hesaplama stratejisini yeni bir pazarlama birimi olarak sunmaya başlamışlardır. Böylelikle grafik donanımı doğrudan kullanabilen bir yapı ortaya çıkmıştır ve adına NVIDIA CUDA denilmiştir.

CUDA teknolojisi C programlama diline dayanan, paralel hesap yapabilmek için GPU komutlarını ve video hafızayı kontrol edebilen, NVIDIA tarafından geliştirilen yazılım-donanım hesaplama mimarisidir. CUDA mimarisi, Nvidia GeForce 8 ve daha yeni model grafik kartlara uygulanabilmektedir. CUDA ile GPU programlama, daha önceki GPGPU çözümlerine göre oldukça esnek ve kolaydır.

Paralel GPU hesaplama işleminin gücü, yapılan çeşitli uygulamalarla ortaya konulmaya başlanmıştır. Bu yöntemle bazı uygulamalar yapılarak, GPU'ların çok hızlı CPU'larla kıyaslandığında 5-30 kat daha hızlı işlem yapabildiği ortaya çıkmıştır. NVIDIA firmasının yaptığı araştırmalara göre elde edilen sonuçlar Şekil 4'de görüldüğü gibidir.



Şekil 4. NVIDIA tarafından yapılan uygulama sonuçları.

Şekil 4 incelendiğinde, çok şaşırtıcı işlem hızı farklarının oluştuğu görülmektedir. Günümüzde GPU hesaplamasını kullanan uygulamaları şu şekilde sıralayabiliriz: Görüntü ve sinyal işleme, fiziksel simülasyon, bilgisayarlı matematik, bilgisayarlı biyoloji, finansal hesaplamalar, veritabanları, gaz ve akışkanlar dinamiği, kriptografi, uygulamalı x-ışını terapisi, astronomi, ses işleme, bio-informatik, biyolojik simülasyonlar, bilgisayarlı görüş, veri madenciliği, moleküler dinamik, manyetik rezonans görüntüleme (MRI), sinir ağları, oşinografik araştırmalar, partikül fiziği, kuantum kimyası, vizualizasyon, radarlar, yapay zeka, uydu veri analizi, video konferans vb. Bu uygulamalara ilişkin detaylı bilgi "www.nvidia.com/cuda" adresinden edinilebilir.

3. UYGULAMALAR

GPGPU yönteminin fotogrametrik yöntemler için de kullanılabileceği ve yöntemin etkinliğinin anlaşılabilmesi amacıyla çalışma içerisinde çeşitli uygulamalar gerçekleştirilmiştir.

3.1. CUDA İle Görüş Analizi

Özellikle arazinin gerçekçi 3B modellerin bilgisayar aracılığıyla gösteriminde eğimin, eğim yönünün ve belirli bir ışık kaynağına göre gölgelendirme değerinin hesaplanması gerekir. Ayrıca arazi ile ilgili çalışmalar yapan ve altlık olarak bir sayısal yükseklik modelini ya da sayısal arazi modelini kullanan her operatör, kullandıkları yazılımlar yardımıyla görüş analizi yapabilmektedir. Görüş analizi temel anlamda gözlemcinin bulunduğu noktadan gözlemlediği alan içerisinde nereleri görebildiğini analiz etmektir. Görüş analizi "bakı analizi", "eğim analizi" şeklinde çeşitlendirilebilir.

Görüş analizinde öncelikle hesaplanması gereken eğimdir. Bu amaca yönelik olarak Şekil 5'deki gibi 3x3 lük bir arama penceresi, yükseklik verisi olan raster grid verisi üzerinde her bir hücre için dolaştırılır. Böylelikle hücrenin çevresindeki 8 komşu hücreye göre ortada yer alan hücrenin eğim değeri hesaplanır.

a	b	c
d	e	f
g	h	i

Şekil 5. 3x3 boyutunda arama penceresi.

Arama penceresinde yer alan "e" hücresinin "x" ve "y" yönündeki değerleri şu şekilde hesaplanır:

$$[dz/dx] = ((c + 2f + i) - (a + 2d + g)) / (8 * hucrebuyuklugu) \quad (1)$$

$$[dz/dy] = ((g + 2h + i) - (a + 2b + c)) / (8 * hucrebuyuklugu)$$

Arazinin durumuna göre "e" hücresinin eğimi radyan cinsinden;

$$Egim\_rad = ATAN (z\_factor * \sqrt{ ([dz/dx]^2 + [dz/dy]^2) }) \quad (2)$$

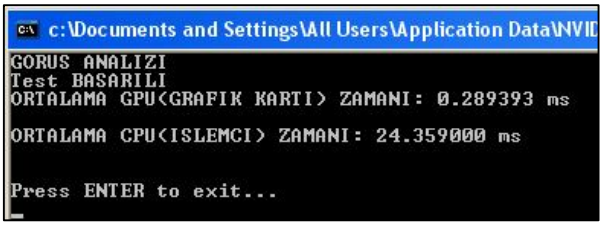
Ayrıca eğimin değiştiği tarafa "eğim yönü" denilir. Bu yön radyan cinsinden 0 ile 2π aralığında değişir. Bu hesaplama için ise ilgili algoritma Şekil 6'daki gibi verilebilir. Bu elde edilen eğim ve eğim yönü ile aynı zamanda gölgelendirme değeri de hesaplanabilmektedir.

```
if [dz/dx] ≠0:

  Aspect_rad = atan2 ([dz/dy], -[dz/dx])
  if Aspect_rad < 0 then
    Aspect_rad = 2π + Aspect_rad

if [dz/dx] = 0
  if [dz/dy] > 0 then
    Aspect_rad = π/2
  else if [dz/dy] < 0 then
     Aspect_rad = 2π - π/2
  else
        Aspect_rad = Aspect_rad
```

Şekil 6. Eğim yönü hesap algoritması.

Görüş analizi problemi dikkate alındığında aynı hesaplama yönteminin sırayla bütün hücrelere uygulandığı görülmektedir. Bu durum da aslında karşımıza çoklu işlemi çıkarmaktadır. Böyle bir durumda CUDA programlama dili kullanılarak problem iş parçalarına bölünüp dağıtık bir şekilde, birçok işlemciye aynı anda verilip, sonuç hesaplamalar elde edilebilirse, çok hızlı sonuçlar elde edilebileceği ortadadır. Buna yönelik olarak görüş analizi problemi kodlanmıştır.

Sonuçta program hem CUDA mimarisini kullanarak grafik kartı üzerindeki işlemcileri (GPU) kullanarak işlem yapmakta, hem de mevcut CPU'yu kullanarak aynı işlemi tekrarlamaktadır. Her iki farklı yöntem için de, işlem süresini ekranda vermektedir.



Şekil 7. Görüş analizi programının arayüzü.

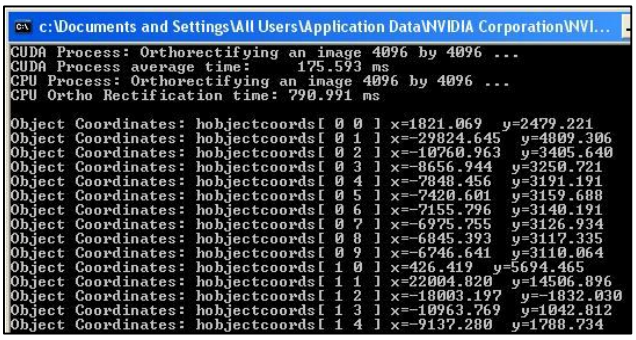
Şekil 7 incelendiğinde, yapılan hesaplamanın GPU süresi 0,29 milisaniye iken, aynı işlem CPU üzerinde yapıldığında 24,36 milisaniye olmaktadır. Aradaki fark hesaplandığında, GPU ile yapılan işlemin CPU ile yapılan işleme göre 84 kat hızlı olduğu görülmektedir. Bu yapılan hesaplamada 512x512 boyutunda bir raster grid veri kullanılmıştır. Raster verinin boyutu küçülürse şayet, CPU nun performansı iyileşmektedir. Fakat bunun yanında GPU'nun performansı aynı kalmaktadır (Tablo 1). Bu da çok yoğun ve tekrarlı hesap gerektiren problemlerde GPU'nun daha iyi sonuçlar ürettiğini ortaya koymaktadır.

Raster Hücre Boyutu	GPU zamanı (milisaniye)	CPU zamanı (milisaniye)	CPU / GPU (oran)
512 x 512	0,29	24,36	84
256 x 256	0,28	5,05	18
128 x 128	0,28	1,59	5,68
64 x 64	0,28	0,41	1,46
32 x 32	0,28	0,11	0,39

Tablo 1. GPU ve CPU ile yapılan hesaplama sonuçları.

3.2. CUDA İle Projektif Rektifikasyon

Ortorektifikasyon amacıyla kullanılabilecek rektifikasyon yöntemlerinden bir olan projektif rektifikasyon yöntemi çalışma içerisinde bir örnek uygulama için uygulanmıştır. Örnek uygulamada bir düzlem alanda çekilen 4096x4096 piksellik bir örnek görüntü değerlendirilmiştir. Örnek görüntü üzerinde obje ve resim koordinatları bilinen dört adet nokta seçilmiştir. Bu noktalar yardımı ile denklem sistemi çözülmüş ve denklem katsayıları hesaplanmıştır. Bu aşamanın ardından katsayıları belirlenen denklem yardımı ile görüntü üzerindeki her bir pikselin obje uzayındaki koordinatları hesaplanmıştır. Bu yapılan işlemler için CUDA programlama dili ile bir program yazılmış ve programın hem CPU, hem de GPU'yu kullanarak aynı işlemleri yapması sağlanarak, işlem süreleri karşılaştırılmıştır. Kodlanan program çalıştırıldığında elde edilen ekran görüntüsü Şekil 8'deki gibidir.



Şekil 8. Projektif rektifikasyon programının çıktı görüntüsü.

Yapılan çalışmada uygulanan projektif rektifikasyon yöntemi, görüntüyü oluşturan her bir piksel için tekrarlanmaktadır. Böyle bir durumda CUDA programlama dili kullanılarak

problem iş parçalarına bölünüp dağıtık bir şekilde, birçok grafik işlemciye aynı anda verilip, sonuç hesaplamalar elde edilmektedir.

Sonuçta program hem CUDA mimarisini kullanarak grafik kartı üzerindeki işlemcileri kullanarak işlem yapmakta, hem de mevcut CPU'yu kullanarak aynı işlemi tekrarlamaktadır. Her iki farklı yöntem için de işlem süresi Şekil 8'de görülmektedir.

Projektif rektifikasyon programının çıktı görüntüsü olan Şekil 8 incelendiğinde, yapılan hesaplamanın GPU süresi 175,593 milisaniye iken, aynı işlem CPU üzerinde yapıldığında 790,991 milisaniye olmaktadır. Aradaki bu fark hesaplandığında, GPU ile yapılan işlemin CPU ile yapılan işleme göre yaklaşık 4,5 kat hızlı olduğu görülmektedir. Bu yapılan uygulama içerisinde 4096x4096 boyutunda bir görüntü kullanılmıştır. Görüntünün boyutu küçülürse şayet, CPU'nun performansı iyileşmektedir (Tablo 2). Bu da çok yoğun ve tekrarlı hesap gerektiren problemlerde GPU'nun daha hızlı sonuçlar ürettiğini ortaya koymaktadır.

Görüntü boyutu (piksel)	GPU Zamanı (milisaniye)	CPU Zamanı (milisaniye)	CPU / GPU (oran)
1024 x 1024	14,09	48,917	3,47
2048 x 2048	75,994	190,626	2,51
4096 x 4096	175,593	790,991	4,50

Tablo 2. GPU ve CPU ile yapılan hesaplama sonuçları.

3.3. CUDA İle Gauss Bulanık (Gaussian Blur) Filtreleme

Gauss bulanık filtreleme, özellikle görüntülerin yumuşatılması için kullanılan bir filtreleme yöntemidir. Bu filtrenin uygulanması ile görüntülerdeki gürültü ve detaylar azaltılabilir. Bu yöntem içerisinde Gauss normal dağılım fonksiyonundan faydalanılır. Yöntem olarak ortalama filtreye benzer.

Bir boyutlu Gauss fonksiyonu şu şekilde verilebilir:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \quad (3)$$

Ayrıca iki boyutlu Gauss fonksiyonu ise:

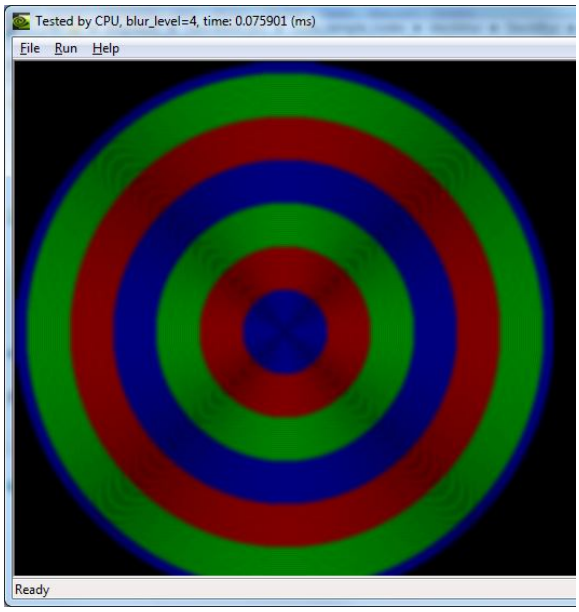
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (4)$$

Denklemiyle ifade edilir.

Burada x = yatay eksende merkezden mesafesi
y = dikey eksende merkezden mesafesi
σ = Gauss dağılımının standart sapması

İki boyutlu olarak bu denklem uygulandığında merkez noktadan Gauss normal dağılımlı kıvrımlar oluşur. Bu dağılımdan elde edilen değerler, orijinal görüntüye uygulanacak olan kıvrılma (convolution) matrisini oluşturur. Her bir pikselin yeni değeri, bu pikselin komşularının ağırlıklı ortalamasına göre yeniden belirlenir.

Bu yöntemin CUDA programlama dili ile kodlanması sonucu elde edilen programın arayüzü Şekil 9'da görülmektedir.



Şekil 9. Gauss bulanık filtreleme programının arayüzü.

Yazılım içerisinde bulanıklık seviyesi değiştirilebilmektedir. Aynı zamanda yapılan işlemin süresi bakımından GPU ve CPU karşılaştırması da yapılabilmektedir. Kodlanan programdan elde edilen sonuçlar incelendiğinde GPU ile yapılan bulanıklaştırma işleminin, CPU ile yapılan işleme oranla yaklaşık 450 kat kadar daha hızlı yapıldığı görülmektedir (Tablo 3).

Bulanıklık Seviyesi	GPU zamanı (milisaniye)	CPU zamanı (milisaniye)	CPU / GPU (oran)
1	0,000234	0,088141	376,67
4	0,000224	0,075901	338,84
7	0,000226	0,088474	391,48
10	0,000224	0,101787	454,41
13	0,000227	0,095546	420,91
16	0,000226	0,102246	452,41

Tablo 3. GPU ve CPU ile yapılan filtreleme sonuçları.

4. SONUÇLAR

Yapılan bu çalışma çerisinde özellikle fotogrametri amacıyla kullanılan bazı uygulamalar GPGPU yöntemi kullanılarak CUDA programlama dili yardımıyla kodlanarak, derlenmiştir. Elde edilen sonuçlar değerlendirildiğinde, yöntemin fotogrametrinin görüntü işlemeyi gerektirdiği ve aynı işlem adımlarının her bir piksel için tekrarlandığı durumlarda ve ayrıca hesap yoğun işlem adımlarında çok etkili ve hızlı sonuçlar verdiği görülmektedir. Yapılan bu uygulamaların sayısı ve çeşitliliği arttırılabilir. Hızın ve hızlı karar verme sürecinin etkili olduğu, hesaplama işleminin çok yoğun olarak kullanıldığı problemler için benzer yöntemler uygulanabilir.

Günümüzde çeşitli platformlar üzerinden elde edilen görüntülerin çok hızlı bir şekilde ve doğru olarak georeferanslandırılması, ayrıca anlık olarak karar vermenin gerektirdiği orman yangını, deprem gibi felaketlerin yaşandığı süreçlerde bu görüntülerin hızlı bir şekilde kullanılabilmesi önemli bir ihtiyaç ve gerekliliktir. Buna yönelik olarak, görüntülerin ortorektifikasyonu sürecinde bu yöntemin kullanılabileceği değerlendirilmektedir.

5. KAYNAKLAR

Kraus, K., 2007. *Fotogrametri Cilt 1 Fotoğraflardan ve Lazer Tarama Verilerinden Geometrik Bilgiler*. İstanbul Teknik Üniversitesi, Nobel Yayın Dağıtım, 1.Basım.

Mercedes Marqu´es, Gregorio Quintana-Ort´ı, Enrique S. Quintana-Ort´ı, Robert van de Geijn. 2009. Using graphics processors to accelerate the solution of out-of-core linear systems *8th IEEE International Symposium on Parallel and Distributed Computing*, Lisbon.

Novak, K. 1992. Rectification of Digital Imagery, *Photogrammetric Engineering and Remote Sensing*, 339-344.

Nvidia, 2009. CUDA Architecture, Introduction and Overview, Nvidia Corp., California, USA.

Nvidia, 2010a. OpenCL Programming Guide for the CUDA Architecture, Nvidia Corp., California, USA.

Nvidia, 2010b. CUDA C Programming Guide, Nvidia Corp. California, USA.

Ylmaz, E., 2010. *Massive Crowg Simulation with Parallel Processing*, Doktora Tezi, Bilişim Sistemleri Bölümü, ODTÜ, Ankara.