# Uzaktan Algılamada Yüksek Başarım

*Mustafa Teke[1,2], Ahmet Erdem[1,2], Alptekin Temizel[2]*

1. HAVELSAN A.Ş
{mteke, aerdem}@havelsan.com.tr

2. Enformatik Enstitüsü
Orta Doğu Teknik Üniversitesi
atemizel@ii.metu.edu.tr

## Öz

*Uydu görüntüleri son yıllardaki teknolojik gelişmeler ve yapılan yatırımlar ile daha kolay elde edilebilir duruma gelmiştir. Artan veri miktarına karşın verileri yüksek performansta kıymetlendirecek sistemler mevcut işlemcilerin saat hızları ile sınırlı kalmaktadır. Uydu görüntüleri genel kullanımdaki fotoğraflardan farklı olarak 4 ve üzeri bantlı ve 16-bit veri yapısı ile saklanmaktadır; görüntü boyutları çözünürlüğe bağlı olarak yüksek olabilmektedir. Bu çalışmada ön işleme, gölge tespiti ve bitki tespiti için kullanılan uzaktan algılama algoritmalarının GPU kullanılarak hızlandırılması ve algoritmaya özel optimizasyon ile ilgili çalışmalar sunulmaktadır. GPU'da yapılan paralelleştirme ve algoritmaya özel iyileştirmeler ile 10 kata kadar performans artışı sağlanabilmektedir.*

## 1. Giriş

Yeryüzünü gözlemlemek için uzaya gönderilen uyduların sayısı arttıkça alınan veri miktarı da artmaktadır. Bir uydu görüntüsünün boyutları bant sayısı ve çözünürlüğüne bağlı olarak çok yüksek olabilmektedir. Örneğin 10000x10000 piksel çözünürlüğünde 4 bant bir uydu görüntüsü 800MB civarında bir boyuta ulaşabilmektedir. Bu büyüklükte verileri günümüz merkezi işlemci (CPU) teknolojisi ile işlemek mümkün olmakta, fakat hızlı işleme için çok sayıda işlemci ünitesini birlikte kullanmak gerekmektedir. Grafik işlemci üniteleri (GPU) CPU teknolojilerine kıyasla çok daha hızlı gelişmektedir ve paralelleştirmeye uygun işlemler genel amaçlı grafik birimleri (GPGPU) ile yapıldığında CPU sistemlerine göre daha yüksek başarım elde edilebilmektedir. Veri ve işlemlerin paralel olduğu görüntü işleme gibi alanlarda GPU paralel hesaplama yöntemlerinin kullanımı işlem süresini kayda değer oranlarda kısaltmaktadır. Bu bildiride, ön işleme, gölge tespiti, bitki tespiti, gölge üreten bölütlerin tespiti gibi uzaktan algılama algoritmalarının paralelleştirilerek GPU üzerinde gerçekleştirilmesi ile ilgili çalışmalarımız sunulacaktır.

GPU'lar 2B/3B grafik oluşturma ve görüntüleme görevlerini yerine getirmek için tasarlanmışlardır. Günümüzde bu grafik üniteleri 1 Teraflop ve üzeri işlem gücü sunmaktadır, buna karşılık CPU'ların işlem güçleri 100 Gigaflop seviyelerindedir. Ayrıca grafik birimleri verileri paralel olarak işlemek için tasarlanmış (örneğin bir sahneyi ekranda çizmek gibi), CPU ise genel amaçlı işlemleri yerine getirmek üzere tasarlanmıştır [1].

GPU ile programlama zorlayıcı bir geliştirme yöntemi olagelmiştir. Fakat günümüzde CUDA (Compute Unified Device Architecture) [2] ve OpenCL [3] gibi geliştirme kütüphanelerin sayesinde uygulama geliştirme süreçlerin kolaylaşmıştır. Bu kütüphaneler sayesinde grafik birimleri çizim benzeri amaçlar dışında kullanılabilmektedir. Grafik birimlerinin genel amaçlı işlemler için kullanılması işlemi general-purpose computation on GPU (GPGPU) olarak adlandırılmaktadır.

Görüntülerin matris (2 ve 3) boyutlu yapısı sayesinde çoğu görüntü işleme algoritması veri seviyesinde paralelleştirilebilmektedir. K. Park *et al* önemli görüntü işleme algoritmalarını CPU ve GPU gerçekleştirimlerini karşılaştırmasını yapmışlardır [4]. GpuCV [5] ve OpenVIDIA [6] görüntü işleme algoritmalarının GPU üzerinde hesaplandığı kütüphanelerdir. Ayrıca OpenCV 2.3 [7] kütüphanesi bir kısım fonksiyonlarının GPU üzerinde çalışan uyarlamalarına sahiptir ve gün geçtikçe artan sayıda fonsiyonun GPU uyarlaması gerçekleştirilmektedir.

Hava ve uydu fotoğraflarının otomatik olarak hızlı bir şekilde işlenmesine ihtiyaç duyulmaktadır. Uzaktan algılama çalışmalarında kullanılan görüntülerdeki gölge bölgelerinin tespiti, bölütleme, 3B sahne geriçatımı ve çakıştırma algoritmalarında performansı düşürdüğü için önemlidir. Ayrıca, gölge bilgisi insan yapısı nesneleri tespit etmek için de kullanılabilmektedir [8]. Sonuç olarak gölge tespiti gölgeleri temizleyerek bağlı algoritmaların performansını arttırdığı gibi hedeflerin tespiti işlemlerinde de ipucu olarak kullanılabilmektedir.

Gölge tespiti algoritmaları genellikle farklı renk uzaylarındaki renk verilerini kullanırlar. Polidorio et al. [9] HSI uzayındaki normalize yeğinlik (I, intensity) ve doygunluk (S, saturation) kullanarak gölge haritası oluşturur. Bu oluşturulan harita sabit bir değer ile eşiklenerek gölge maskesi elde edilir.

Canlı bitkiler güneşten gelen radyasyonu fotosentez işlemi sırasında soğurur, yaprak hücreleri kızılötesi bandındaki radyasyonu ihtiyaç olmadığı için geri yansıtır [10]. Sonuç olarak bitkiler yüksek kızılötesi değerlerine sahiptir; bu sonuç Normalize Bitki İndeksi (Normalized Vegetation Index (NDVI)) ile özetlenebilir:

$$NDVI = \frac{NIR - VIS}{NIR + VIS} \quad (1)$$

NIR kızılötesi, VIS görünür renk değeridir. Bitkisel alanlar 0.2 ile 0.4 arasında değişen oranlara sahip bölgelerdir. Bitki İndeksi değeri sensör kalitesi ve atmosferik etkilerden etkilenebilmektedir. Normalize Bitki İndeksi ilk geliştirilen ve en başarılı uzaktan algılama algoritmalarından birisidir.

Uzaktan algılama verileri ile ilgili başka bir algoritma grubu ise ön işleme algoritmalarıdır. Bu algoritmalar coğrafi bilgi sistemlerinde yoğun bir şekilde kullanılmaktadırlar. Bu algoritmalar gölge veya bitki tespiti gibi algoritmaların performanslarını arttırmanın yanısıra görüntülerin göz ile de daha kolay görülmesini sağlamaktadır. En yaygın kullanılan ön işleme yöntemi histogram eşitleme ve kontrast açmadır.

Kontrast açma, görüntü piksel değerlerinin [0, 255] (12-bit uydu görüntüleri için [0-2047] aralığına genişleme yapılır) aralığına lineer bir şekilde genişletilmesidir. Kontrast açma işleminin zayıf tarafı az görülen piksel değerlerinin, yaygın görülen piksel değerleri ile aynı oranda genişletiliyor olmasıdır. Histogram eşitleme operasyonu ise piksel değerlerinin olasılıksal

dağılımlarına göre iyileştirme yapar ancak piksellerin dağılımını lineer olmayan bir şekilde değiştirdiği için gölge bulma işlemi için kontrast açma işlemi daha uygundur.

Gölge üreten bölütlerin tespiti nesne tespiti için önemli bir ön aşamadır. Literatürde gölge bilgisi bina ve hasar tespitinde kullanılmıştır. Gölge üreten bölütlerin tespiti işlemini hızlandırmak daha sonraki kullanılacak nesne tespiti algoritmaları için önemlidir.

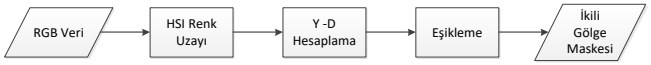
Sırmaçek B., ve Ünsalan C. [11], gölge verilerini nesne tespitinde ipucu olarak kullanmıştır. Irwin R. B., ve McKeown D. M. [12], gölgeyle insan yapısı nesnelerin birbirleriyle olan ilişkilerini kullanarak otomatik olarak insan yapısı nesneleri bulmak üzere yöntemler geliştirmişlerdir. Vu T., Matsouka M., ve Yamazaki F. [13], binaların gölgelerinin yönelim, şekil ve büyüklüklerine göre deprem sonrası hasar tespitinde gölge ve binaların ilişkilerini kullanmıştır. Sırmaçek B., ve Ünsalan C. [14], hasarlı binaların tespitinde gölge ile bina ilişkisini ve gölgelerin şekillerini kullanmıştır.

## 2. Algoritma Detayları ve Gerçekleştirimler

Bu çalışmada gölge tespiti algoritması olarak Polidorio et al. [9] tarafından önerilen yöntem kullanılmıştır. Öncelikli olarak RGB renk uzayı HSI renk uzayına çevrilmiş, daha sonra yeğinlik ve doygunluk farkı kullanılarak gölge haritası oluşturulmuştur:

$$Y - D < K \qquad (3)$$

*K* eşik değeri uydu görüntüleri için -0.2 olarak seçilmiştir. İkili gölge maskesi eşik değeri kullanılarak elde edilmektedir. Algoritma akış diyagramı Şekil 1'de gösterilmektedir.



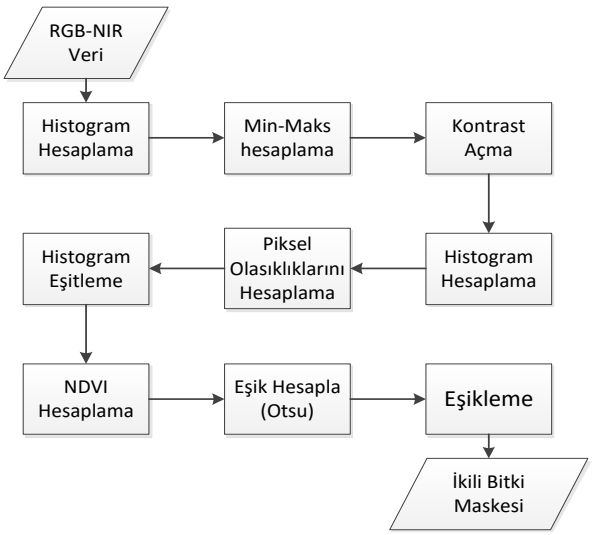

Şekil 1: Gölge Tespiti Algoritması

Bitki tespiti için NDVI hesaplanarak eşik değeri ile ikili maske elde edilmiştir. Bitki algoritması eşik değeri Şekil 2'de gösterilmektedir.


*Şekil 2:* Bitki Tespiti Algoritması

Bitki tespiti algoritmasının gerçekleştirilmesi iki farklı ön işleme algoritması (kontrast açma ve histogram eşitleme) ile birlikte yapılmıştır. İlk olarak kontrast açma işlemi uygulanır, daha sonra normalizasyonu yapılan görüntü üzerine histogram eşitleme ön işlemi tamamlanır. Son olarak NDVI hesaplanarak Otsu'nun yöntemi [15] ile otomatik eşitleme yapılır. Algoritma adımları Şekil 3'de gösterilmektedir.

Gölge üreten bölütlerin tespitinde görüntü meanshift [17] algoritması kullanılarak bölütlere ayrılmıştır. Bu bölütler gölgelerle ilişkilendirilmiştir. Bu amaçla gölge sonuçları ışık yönünün tersi yönde büyütülmüş ve orijinal gölge sonuçları büyütme sonuçlarından çıkarılmıştır. Elde edilen sonuçlar bölütlerle birleştirilmiştir. Üzerindeki gölge miktarı eşik değerinden büyük olan bölütler gölge üreten bölütler olarak seçilmiştir.



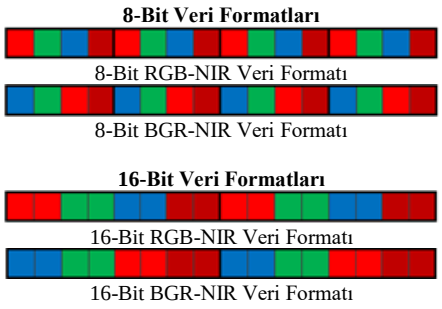

*Şekil 3:* Ön İşleme, Otomatik Eşikleme ve Bitki Tespiti Algoritması

Algoritmalar C++ dilinde OpenCV [7] ve OpenMP (OMP) [15] kullanılarak gerçekleştirilmiştir. OpenMP ile algoritmalar çok izlekli çalışması sağlanmıştır. 4 bant, 16-bit uydu görüntülerinin yüklenmesi GDAL kütüphanesi kullanılarak yapılmıştır.

Testlerde kullanılan algoritmalar veri seviyesinde paralelleştirilmeye uygun algoritmalardır; bu sayede algoritma adımları CUDA (NVIDIA Compute Unified Device Architecture) çekirdek kodlarına çevirebilmiştir.

Bitki tespiti algoritmasını hızlandırmak için ek adımlar kullanılmıştır. Bunlardan ilki ön işleme adımlarında kullanılmayan mavi ve kırmızı bantların çıkarılması olmuştur. Bu sayede yapılan işlem ve hafıza erişiminden zaman kazancı sağlamak mümkündür.

İkinci iyileştirme ise veri sırasını değiştirme işlemidir. Görüntüler RGB-NIR veya BGR-NIR formatında olabilmektedir. NDVI algoritmasında kırmızı ve kızılötesi bantlar kullanıldığı için BGR-NIR bant formatında hafıza erişimi kısalacaktır; R ve NIR bantlarının erişimleri sırasında veri okuma işlemi komşu bloklar arasında yapılacağı için okumadan kaynaklanan gecikmeler azalacaktır. 8-bit ve 16-bit görüntüler için yapılan iyileştirmeler Şekil 4 ile gösterilmektedir.



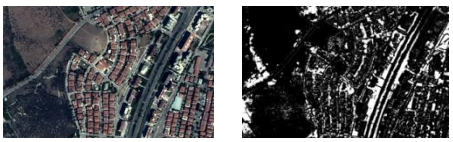

*Şekil 4:* 8 ve 16-bit veriler için değiştirilmiş görüntü formatları

## 3. Test Sonuçları

Testlerde farklı boyutlardaki görüntüler kullanılarak performans ölçümleri yapılmıştır. Bitki tespiti algoritması (Algoritma 2) en

basit algoritma olmakla birlikte ön işleme kullanılarak gerçekleştirilen bitki tespiti algoritması (Algoritma 3) en karmaşık algoritmadır. Testler Intel i7 860 işlemcisi (4 çekirdek 8 izlek) ve NVIDIA GTX 460 (336 CUDA çekirdek) ile yapılmıştır.

Ortalama boyutlardaki bir uydu görüntüsü 5000x5000 piksel olabilmektedir ve hafızada 200MB civarında veri alanı kaplamaktadır. Yapılan testlerde veri boyutunun performansa olan etkisi araştırılmıştır. Bu amaçla çeşitli boyutlardaki 4 bant 16-bit görüntüler kullanılmıştır: küçük (322x265 piksel, 1MB), orta (1535x968, 12MB) ve büyük (4657x4241, 154MB) görüntüler. Bu görüntüler ayrıca 8-bit formatına çevirilerek aynı testler yapılmış ve sonuçlar karşılaştırılmıştır.



(a) (b)

*Şekil 5:* (a) Test görüntüsü, (b) Bitki algoritması gerçekleştirimi sonucu

Şekil 5'te 1535x968 boyutlu görüntü için bitki algoritması gerçekleştirim sonucu görüntülenmektedir.

Tablo 1'de görüntü boyutunun farklı algoritmaların performanslarına olan etkisi gösterilmektedir. Küçük (322x265) boyuttaki görüntü için veri işleme performansı orta ve büyük boyutlardaki görüntülerle karşılaştırıldığında düşük kalmaktadır; grafik biriminde bulunan çekirdekler düşük hacimli veriler için yeterince verimli kullanılamamaktadır. Bazı düşük hacimli veri kullanılan durumlarda grafik birimleri üzerinde yapılan gerçekleştirimler genel amaçlı işlemci gerçekleştirimlerinden daha düşük performans vermektedir. NDVI algoritması gölge tespiti algoritmasından daha çok işlem yapmakta ve daha uzun sürmektedir. Gölge tespiti (Algoritma 1) ve Bitki Tespiti (Algoritma 2) algoritmalarının tek bir çekirdek operasyonu ile kodlanmaları sonucu toplam süre daha az olmaktadır.

*Tablo 1*: Görüntü boyutunun GPU gerçekleştirimlerinde performansa olan etkileri

| | Görüntü Boyutu | Süre (ms) | Mpiksel/s |
|---|---|---|---|
| *Gölge (Algoritma 1)* | 322x265 | 0.38 | 888 |
| | 1535x968 | 1.73 | 3440 |
| | 4657x4241 | 22.51 | 3510 |
| *Bitki (Algoritma 2)* | 322x265 | 0.38 | 907 |
| | 1535x968 | 0.73 | 8094 |
| | 4657x4241 | 7.56 | 10450 |
| *Gölge ve Bitki* | 322x265 | 0.36 | 960 |
| | 1535x968 | 1.81 | 3280 |
| | 4657x4241 | 26.94 | 2933 |

Tablo 2'de hafızaya veri aktarımının performansa olan etkisi görülmektedir. Algoritma karmaşıklığı azaldıkça veri transferinin toplam süreye etkisi artmaktadır. Örneğin en basit algoritma olan bitki tespit algoritması için veri transferi toplam zamanın büyük kısmını kapsamaktadır. *GPU 8 Bit Veri* ile algortimaların sadece GPU'da işleme zamanı gösterilmekte iken *GPU 8 Bit Veri ve Hafıza Erişimi* ile verilerin GPU'ya aktarılma zamanları da hesaplamaya dahil edilmektedir.

Genel amaçlı işlemci gerçekleştirimleri ile yapılan performans karşılaştırmalarında eşit koşullarda ölçüm yapmak için hafıza veri aktarımı yapılan kısımlarda ölçülmelidir. Yapılan testlerde gölge, bitki ve gölge+bitki algoritmaları sırası ile 10.9, 8.2 ve 14.7 kat hızlandırılmışlardır.

*Tablo 2*: Hafızaya veri aktarımının performansa olan etkisi

| | Algoritma | Süre (ms) | Mpiksel/s |
|---|---|---|---|
| *GPU 8 Bit Veri* | Gölge | 22.51 | 3510 |
| | Bitki | 7.56 | 10450 |
| | Gölge+Bitki | 26.94 | 2933 |
| *GPU 8 Bit Veri ve Hafıza Erişimi* | Gölge | 59.88 | 1319 |
| | Bitki | 47.42 | 1666 |
| | Gölge+Bitki | 75.74 | 1043 |
| *CPU 16 Bit Veri* | Gölge | 652.62 | 121 |
| | Bitki | 390.93 | 202 |
| | Gölge+Bitki | 1112.94 | 71 |

Tablo 3'te bitki tespiti algoritmasının ön işleme ve otomatik eşitleme kullanılarak yapılan gerçekleştiriminin performans ölçümü sonuçları verilmektedir. Küçük görüntü boyutu için performans artışı sınırlı kalmaktadır. Görüntü boyutu arttıkça 10.2 kata kadar performans artışı görülmektedir. 8-bit veri ile yapılan testler 16-bit veri kullanılan duruma göre 3 ile 7 kat arasında artış göstermektedir fakat sonuçlarda hassasiyet azalmaktadır.

Kompleks NDVI algoritmasının (Algoritma 3) performansı sadece kırmızı ve kızılötesi bantlarının kullanılması ile arttırılabilir. Tablo 4 ile grafik birimi ile yapılan gerçekleştirimin sonuçları 8-bit ve 16-bit veri için sunulmaktadır. 16-bit veri ile yapılan testlerde daha yüksek veriye ulaşım yapıldığı için süre arttışı daha yüksektir.

Mavi ve yeşil bantların ön işleme adımlarından çıkarılması performansı arttırmıştır fakat kırmızı renk verisi hafıza bloğunun başından, kızılötesi ise sonundan okunmaktadır. Aralıklı veri okumayı azaltmak için veri formatı RGB-NIR yerine BGR-NIR olarak tutulursa bir miktadr daha artış sağlamak mümkün olacaktır. Orta boyutlu görüntü (1535x968) için yapılan testlerde sonuçar Tablo 5 ile verilmektedir. Bant sırasını değiştirmek 8-bit ve 16-bit görüntülerde sırası ile %36.5 ve %4.5 performans artışı sağlamıştır. Sonuçlar grafik birimi gerçekleştirimi ile genel amaçlı işlemcinin çoklu işlek gerçekleştiriminin karşılaştırılması ile elde edilmiştir.

Gölge üreten bölütlerin tespitinde CPU ve GPU performansları karşılaştırılmıştır. Buna göre görüntü boyutu büyüdükçe GPU'nun performans kazancı artmaktadır. Bunun nedeni görüntü boyutu büyüdükçe GPU kapasitesi daha çok kullanılabilmiş olmasıdır. CPU ve GPU karşılaştırma sonuçları Tablo 6: Bunun yanında bölüt sayısı performans değerlerini etkilemektedir.

*Tablo 3*: Kompleks Bitki Tespiti Algoritması (Algoritma 3) için CPU ve GPU gerçekleştirim sonuçları

| | Görüntü Boyutu | Süre (ms) | Mpiksel/s |
|---|---|---|---|
| 8 Bit GPU Veri | 322x265 | 3.04 | 112.28 |
| | 1535x968 | 15.6 | 380.99 |
| | 4657x4241 | 163.2 | 484.08 |
| 16 Bit GPU Veri | 322x265 | 5.06 | 67.45 |
| | 1535x968 | 31.08 | 191.23 |
| | 4657x4241 | 397.45 | 198.77 |
| 8 Bit CPU Veri OMP | 322x265 | 14.62 | 23.35 |
| | 1535x968 | 156.22 | 38.05 |
| | 4657x4241 | 1866.28 | 47.41 |
| 8 Bit CPU Veri | 322x265 | 18.42 | 18.53 |
| | 1535x968 | 311.23 | 19.10 |
| | 4657x4241 | 4004.91 | 19.73 |
| 16 Bit CPU Veri OMP | 322x265 | 10.03 | 34.03 |
| | 1535x968 | 156.08 | 38.08 |
| | 4657x4241 | 1688.06 | 46.80 |
| 16 Bit CPU Veri | 322x265 | 14.03 | 24.33 |
| | 1535x968 | 371.08 | 16.02 |
| | 4657x4241 | 4467.75 | 17.68 |

*Tablo 4*: Kompleks Bitki Tespiti Algoritması(Algoritma 3) için işlemci ve grafik birimi gerçekleştirim sonuçları

| Veri Formatı | Süre (ms) | Mpixels/sec |
|---|---|---|
| 8-Bit (4 bant) | 15.60 | 380.99 |
| 8-Bit (2-bant) | 13.08 | 454.54 |
| 16-Bit (4 bant) | 31.08 | 191.23 |
| 16-Bit (2-bant) | 20.37 | 291.77 |

*Tablo 5*: Bant sırasını değiştirmenin gerçekleştirim performansına etkileri

| Bit Sayısı | Bant Sırası | Süre (ms) | Mpixels/s |
|---|---|---|---|
| 8-Bit | RGB-NIR | 13.08 | 455 |
| 8-Bit | BGR-NIR | 9.57 | 621 |
| 16-Bit | RGB-NIR | 20.37 | 292 |
| 16-Bit | BGR-NIR | 19.49 | 305 |

Gölge üreten bölütlerin sonuçları Şekil 6'da gösterilmiştir. Bu aşamadan sonra bu bölütler nesne tespitinde girdi olarak kullanılabilir.

*Tablo 6:* Gölge Üreten Bölütler Sonucu

| Görüntü Boyutu | Süre (CPU) | Süre(GPU) |
|---|---|---|
| (174*217) | 0.39ms | 0.29ms |
| (290*385) | 0.62ms | 0.40ms |
| (369*414) | 0.74ms | 0.50ms |
| (1535*968) | 28.85ms | 5.98ms |

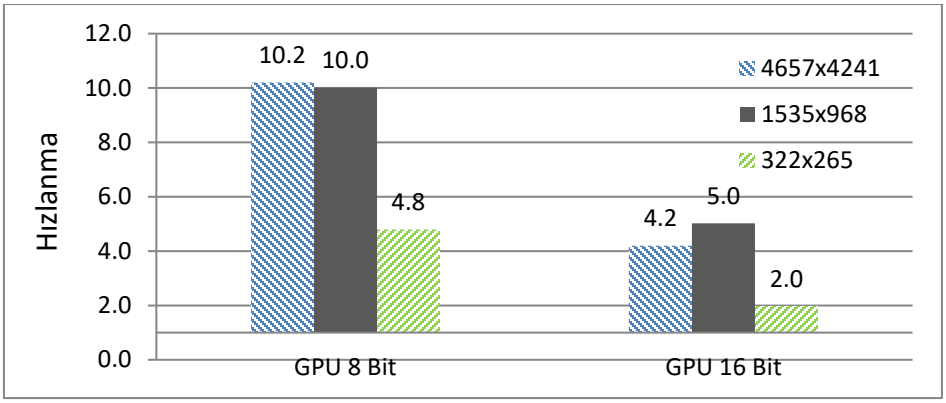

*Şekil 6 a. Orijinal Görüntü b. Bölütleme Sonuçları c. Gölge Sonuçları d. Gölge Üreten Bölüt Sonuçları*

## 4. Sonuçlar

Grafik birimleri genel amaçlı işlemcilerden çok daha hızlı gelişmektedirler ve paralel işlemler için daha uygundurlar. Yaptığımız testlerde görüntü boyutu artışı ile grafik işleme birimlerini daha verimli kullanabildiğimizi gördük. Genel amaçlı işlemci performansı, ön belleğine alınabilecek kadar küçük görüntülerde, daha büyük görüntülere göre daha yüksek olmaktadır. Hassasiyet kaybının önemli olmadığı durumlarda (örneğin 16-bit'ten 8-bit'e yapılan çevirim) veri miktarını küçültmek performans artışı sağlamaktadır. Grafik birimlerinde geçen sürenin büyük bölümü veriye ulaşım sırasında harcanmaktadır. Veri boyutunu 16-bit'ten 8-bit'e çevirdiğimizde performansın 2 kat artması bunu göstermektedir. Ayrıca genel amaçlı işlemci birimi verinin formatının 8-bit veya 16-bit olmasından çok etkilenmemektedir. Şekil 7 ile gösterilen sonuçlarda gösterildiği gibi 4657x4241 piksel boyutlu görüntü için grafik birimi performansı çoklu işlek gerçekleştirim performansının 4.2 katı iken bu oran 8-bit veri için 10.2 kattır.

Yaptığımız diğer bir gözlem NDVI gibi bir algoritma o algoritmaya özel optimizasyonlar ile daha da hızlandırılabilir; kullanılmayan bantların ön işlemeden hariç tutulması ve veri formatının daha hızlı veri erişimine uygun hale getirilmesi gibi.

Sonuç olarak görüntüler doğası gereği paralel uygulamaya elverişli oldukları için (veri paralel) grafik biriminde yapılan işlemler ile hız artışı sağlanmaktadır. Yaptığımız testlerde büyük boyutlu görüntülerini (örneğin 10000x10000, 800MB) giriş seviyesi bir grafik biriminde parçalara ayırmadan işleyemediğimizi gördük Testlerde kullanılan algoritmalarda ön işleme işlemlerinde tüm görüntüden alınan istatistikler kullanılacağı için grafik birimlerinin kısıtlı hafızaları sorun teşkil etmektedir.




*Şekil 7:* Grafik birimi işlemci gerçekleştirilmesinin genel amaçlı işlemci çoklu izlek gerçekleştirilmesine göre hızlanma oranları (Tablo 3).

## 5. Kaynakça


[1] D. B. Kirk and W. W. Hwu , Programming Massively Parallel Processors A Hands-on Approach. Elsevier, 2010.

[2] NVIDIA Corporation, Compute Unified Device Architecture (CUDA), http://developer.nvidia.com/object/cuda.html, 2011.

[3] Khronos Group, Open Computing Language (OpenCL), www.khronos.org/opencl/, 2011

[4] In K. Park, Nitin Singhal, Man H. Lee, Sungdae Cho, Chris Kim, "Design and Performance Evaluation of Image Processing Algorithms on GPUs," IEEE Transactions on Parallel and Distributed Systems, Vol. 99, No. 1., Feb. 2011.

[5] Y. Allusse, P. Horain, A. Agarwal, and C. Saipriyadarshan, "GpuCV: An Opensource Gpu-Accelerated Framework for Image Processing and Computer Vision," in Proc. ACM Int'l Conf. Multimedia, Oct. 2008, pp. 1089-1092.

[6] J. Fung, S. Mann, and C. Aimone, "OpenVIDIA: Parallel GPU Computer Vision," in Proc. ACM Int'l Conf. Multimedia, Nov. 2005, pp. 849-852.

[7] OpenCV Open computer Vision Library, http://opencvlibrary.sourceforge.net/, 2011.

[8] Irvin B. and McKeown J.R. , "Methods for exploiting the relationship between buildings and their shadows in aerial imagery", IEEE Transactions on System, Man and Cybernetics, 19(6), pp. 1564-157, 1989.

[9] A. M. Polidorio, F. C. Flores, N. N. Imai, A. M. G. Tommaselli and C. Franco, "Automatic Shadow Segmentation in Aerial Color Images", in Proc. of the XVI Brazilian Symposium on Computer Graphics and Image Processing, Sao Carlos, Nov. 2003, pp. 270-277.

[10] Tucker, C.J., "Red and Photographic Infrared Linear Combinations for Monitoring Vegetation", Remote Sensing of Environment, 8(2),127-150, 1979.

[11] Sırmaçek B., ve Ünsalan C., "Building Detection from Aerial Images Using Invariant Color Features and Shadow Information", IEEE International Symposium on Computer and Information Sciences, (2008).

[12] Irwin R. B., and McKeown D. M., "Methods for Exploiting the Relationship Between Buildings and Their Shadows in Aerial Imagery", IEEE Transactions on Systems. Man, and Cybernetics, Vol. 19, NO. 6, 1564 - 1575 November/December (1989).

[13] Vu T., Matsouka M., and Yamazaki F., "Shadow analysis in assisting damage detection due to earthquake from quickbird imagery," Proceedings of The 10th International Society For Photogrammetry and Remote Sensing Congress, pp. 607–611, (2004).

[14] Sırmaçek B., ve Ünsalan C., "Damaged Building Detection in Aerial Images using Shadow Information", IEEE 4th International Conference on Recent Advances in Space Technologies (RAST '09) , (2009).

[15] Nobuyuki Otsu, "A threshold selection method from gray-level histograms". IEEE Trans. Sys., Man., Cyber. 9: 62–66, 1979.

[16] OpenMP Website, http://openmp.org/wp/, 2012.

[17] Comanicu D., and Meer P., "Mean Shift Analysis and Applications," The Proceedings of the Seventh IEEE International Conference on Computer Vision,, vol.2, no., pp.1197-1203 vol.2, (1999).