İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

GRAFİK İŞLEMCİ BİRİMİ ÜZERİNDE GENEL AMAÇLI HESAPLAMA YÖNTEMİ İLE GÖRÜNTÜLERİN GERÇEK ZAMANLI ORTOREKTİFİKASYONU

DOKTORA TEZİ

Hakan ŞAHİN

Geomatik Mühendisliği Anabilim Dalı

Geomatik Mühendisliği Programı

MART 2016

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

GRAFİK İŞLEMCİ BİRİMİ ÜZERİNDE GENEL AMAÇLI HESAPLAMA YÖNTEMİ İLE GÖRÜNTÜLERİN GERÇEK ZAMANLI ORTOREKTİFİKASYONU

DOKTORA TEZİ

Hakan ŞAHİN
(501042604)

Geomatik Mühendisliği Anabilim Dalı

Geomatik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Mehmet Sıtkı KÜLÜR

MART 2016

İTÜ, Fen Bilimleri Enstitüsü'nün 501042604 numaralı Doktora Öğrencisi Hakan ŞAHİN, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "GRAFİK İŞLEMCİ BİRİMİ ÜZERİNDE GENEL AMAÇLI HESAPLAMA YÖNTEMİ İLE GÖRÜNTÜLERİN GERÇEK ZAMANLI ORTOREKTİFİKASYONU" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur

Tez Danışmanı :	**Prof. Dr. Mehmet Sıtkı KÜLÜR** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Mehmet Orhan ALTAN** İstanbul Teknik Üniversitesi	.....
	Prof. Dr. Zübeyde ALKIŞ Yıldız Teknik Üniversitesi	.....
	Prof. Dr. Ferruh YILDIZ Selçuk Üniversitesi	.....
	Doç. Dr. Zaide DURAN İstanbul Teknik Üniversitesi	.....

Teslim Tarihi : 16 Ekim 2015
Savunma Tarihi : 04 Mart 2016

Eşime ve Kızıma,

ÖNSÖZ

Doktora tez çalışmama aslında, çok uzun bir süre önce başladım. Fakat, meslek hayatımda verdiğim bir kararla, Zonguldak Karaelmas Üniversitesinde sürdürdüğüm akademisyenliği bırakarak, meslek hayatımı bir neferi olmaktan büyük gurur duyduğum Türk Silahlı Kuvvetleri bünyesindeki Harita Genel Komutanlığında devam ettirmeye başladım. Bu geçiş dönemi biraz uzun ve yorucu oldu. Bu noktada bu geçiş sebebiyle tez konumu değiştirerek yeniden bu tez çalışması içerisinde anlatılmaya çalışılan konu üzerinde çalışmaya başladım ve sonuçlandırdım.

Bu aşamalarda bana her zaman her konuda sonsuz desteğini veren, yılmadan sıkılmadan yaşadığım problemleri çözen Danışman Hocam Sayın Prof.Dr.Mehmet Sıtkı KÜLÜR'e, büyük bir sabır göstererek tez çalışmasının her aşamasında fikirleri ile beni yönlendiren Sayın Prof.Dr.Orhan ALTAN ve Sayın Prof.Dr.Zübeyde ALKIŞ'a teşekkürü bir borç bilir, saygılarımı sunarım.

Tez çalışmamda çeşitli şekillerde destek gördüğüm İstanbul Teknik Üniversitesi'ne, Zonguldak Karaelmas Üniversitesi'ne, Harita Genel Komutanlığı'na ve personeline, birlikte mesai paylaştığım tüm arkadaşlarıma teşekkür ederim.

Ayrıca çalışmada fikirleriyle birçok konuda destek veren dostum Yrd.Doç.Dr.Mehmet Güven KOÇAK'a, emekli Dr.Müh.Bnb.Erdal YILMAZ'a ve Dr.Müh.Bnb.Mehmet ERBAŞ'a teşekkür ederim.

Bugünlere gelmemde her türlü maddi ve manevi desteği esirgemeyen, üzerimde sonsuz emekleri olan anne ve babama içten şükranlarımı sunarım.

Eşim Yük.İnş.Müh.Fatma ŞAHİN'e de bu zamana kadar gösterdiği anlayış, yardım ve bitmek bilmeyen desteği için sonsuz ve en derin sevgilerimi sunar, teşekkür ederim. Son olarak hayatımın talih kuşu olan Kızım Hüma'dan da, ondan çaldığım birlikte geçirmemiz gereken zamanlar için beni affetmesini ister, ileride büyüdüğünde beni anlayışla karşılayacağını umarım.

Ekim 2015

Hakan ŞAHİN
(Mühendis Subay)

İÇİNDEKİLER

Sayfa

ÖNSÖZ…… **vii**
İÇİNDEKİLER **ix**
KISALTMALAR **xi**
SEMBOLLER **xiii**
ÇİZELGE LİSTESİ **xv**
ŞEKİL LİSTESİ **xvii**
ÖZET…… **xix**
SUMMARY **xxiii**
1. GİRİŞ…… **1**
1.1 Tezin Amacı ve Kapsamı 3
2. ORTOREKTİFİKASYON **5**
2.1 Polinomal Dönüşüm 7
2.2 Projektif Dönüşüm 8
2.3 Sensör Model Dönüşümü 8
2.4 Rasyonel Fonksiyon Model Dönüşümü 9
2.5 Diferansiyel Ortorektifikasyon 9
2.5.1 Görüntü Koordinat Sisteminden Resim Koordinat Sistemine Dönüşüm 14
2.5.2 Resim Koordinat Sisteminden Kamera Koordinat Sistemine Dönüşüm 15
2.5.3 Kamera Koordinat Sisteminden IMU Koordinat Sistemine Dönüşüm 15
2.5.4 IMU Koordinat Sisteminden Yerel Coğrafi Koord. Sistemine Dönüşüm 16
2.5.5 Kesişim noktasının hesaplanması 17
2.5.5.1 Yerel Coğrafi Koord. Sisteminden ECEF Koord. Sist. Dönüşüm 17
2.5.5.2 Atmosferik Refraksiyon Etkisi 21
2.5.5.3 ECEF Koord. Sisteminden Yerel Coğrafi Koord. Sist. Dönüşüm 22
2.5.6 Rölyef Kayma Düzeltmesi 23
2.5.7 Coğrafi Koordinat Sisteminden UTM Koordinat Sistemine Dönüşüm 25
2.5.8 Yeniden Örnekleme 27
3. PARALEL İŞLEM VE NVIDIA CUDA TEKNOLOJİSİ **29**
3.1 Paralel İşlemin Tarihçesi ve Merkezi İşlem Birimi 29
3.2 GPGPU ve Akış İşleme 31
3.3 CUDA Teknolojisi ve Genel Amaçlı Hesaplama 36
4. UYGULAMALAR **47**
4.1 Görüş Analizi Uygulaması 49
4.2 Gauss Bulanık (Gaussian Blur) Filtreleme Uygulaması 51
4.3 Uygulamalarda Kullanılan Görüntülerin Teknik Özellikleri 53
4.4 Projektif Dönüşüm Uygulaması 54
4.5 Diferansiyel Ortorektifikasyon Uygulaması 56
4.6 Uygulama Sonuçlarının Değerlendirilmesi 58
5. SONUÇ VE ÖNERİLER **61**
5.1 Sonuçlar 61

5.2 Öneriler 63
KAYNAKLAR 65
ÖZGEÇMİŞ 69

KISALTMALAR

İHA : İnsansız Hava Aracı
IMU : Inertial Measurement Unit
GPS : Küresel Konumlama Sistemi
LIDAR : Lazer Tarayıcı Sistemler
SAR : Yapay Açıklıklı Radar
InSAR : Interferometrik SAR
INS : Inertial Navigation Systems
VISAT : Video Cameras an Inertial System and Satellite GPS Receivers
GPU : Grafik İşlemci Birimi
CPU : Merkezi İşlem Birimi
GPGPU : Grafik İşlemci Birimi Üzerinde Genel Amaçlı Hesaplama
YKN : Yer Kontrol Noktası
CUDA : Birleşik Hesaplama Aygıt Mimarisi
CCD : Charge Coupled Device
ECEF : Yer Merkezli Yer Sabit
SYM : Sayısal Yükseklik Modeli
UTM : Universal Transvers Mercator
2B : İki Boyutlu
3B : Üç Boyutlu
API : Uygulama Programlama Arayüzü
SFU : Özel Fonksiyonlar Birimi
SM : Akış İşlemcileri
SP : Skaler İşlemciler
SIMT : Single Instruction Multiple Thread
MRI : Manyetik Rezonans Görüntüleme
RAM : Rastgele Erişilebilir Bellek

SEMBOLLER

(X_P, Y_P, Z_P) : Referans koordinat sistemindeki koordinatlar
(x_p, y_p) : Fotoğraf koordinat sistemindeki koordinatlar
f : Odak uzaklığı
α : Ölçek katsayısı
(x_0, y_0) : Resim ana noktasının resim koordinatları
(X_0, Y_0, Z_0) : Resim çekme merkezinin koordinatları
$(\omega, \varphi, \kappa)$: Dönüklük matrisi elemanları
c : Resim çekme makinesi sabiti
x, y : Objenin resim uzayındaki nokta koordinatları
a, b : Polinom katsayıları
n : Polinomun derecesi
(r_n, c_n) : Piksellerin resim uzayındaki normalleştirilmiş satır, sütun indeksleri
(X, Y, Z) : Obje nokta koordinatları
(X_n, Y_n, Z_n) : Obje noktasının obje uzayındaki normalleştirilmiş koordinatları
$a_{ijk}, b_{ijk}, c_{ijk}$: Rasyonel fonksiyon katsayıları
x'', y'' : Digital görüntüdeki kartezyen piksel koordinatları
$width, height$: Görüntünün genişliği ve yüksekliği
$\Delta x, \Delta y$: Resim ana noktasının x ve y görüntü koordinatları
x', y' : Kamera koordinat sistemindeki koordinatlar
k_1 ve k_2 : Radyal distorsiyon parametreleri
p_1 ve p_2 : Teğetsel distorsiyon parametreleri
$\overset{c}{D}$: Kamera koordinat sistemine bağlı yön vektörü
$\overset{B}{D}$: Resim noktasının IMU koordinat sistemindeki yön vektörü
R_1, R_2, R_3 : Dönüklük matrisleri
φ_A, λ_A : Yerel coğrafi koordinat sisteminin merkezinin enlem ve boylamı
s : Kamera ile referans elipsoidi üzerinde kesişim noktası arası mesafe
a, b : Elipsoidin yarıeksen bileşenleri
n : Ortam refraksiyon indeksi
θ_1, θ_2 : Işığın iki ortam arasındaki sınıra giriş, çıkış açıları.
N . Refraktivite
h : Ortalama deniz seviyesi yüksekliği
p : Atmosferik basınç
T : Atmosfer sıcaklığı
e : Su buharı basıncı
ϕ, λ : Enlem, boylam
α, z : Azimut ve zenit açıları
M : Meridyen mesafesi
N : Elipsoid normali uzunluğu

ÇİZELGE LİSTESİ

Sayfa

Çizelge 4.1 : Görüş analizinde, GPU ve CPU ile yapılan hesaplama hızı sonuçları. **51**
Çizelge 4.2 : GPU ve CPU ile yapılan Gauss bulanık filtreleme hızı sonuçları..... **52**
Çizelge 4.3 : UltracamX Vexcel sayısal kameranın teknik özellikleri. **53**
Çizelge 4.4 : GPU ve CPU ile yapılan uygulamanın hız testi sonuçları..... **56**
Çizelge 4.5 : GPU ve CPU ile yapılan uygulamanın hız testi sonuçları..... **57**
Çizelge 4.6 : GPU ve CPU ile yapılan görüntü üzerinden bulut kaldırılması uygulamasının hız testi sonuçları. **58**
Çizelge 4.7 : GPU ve CPU ile yapılan hiperspektral görüntü sıkıştırma uygulamasının hız testi sonuçları. **59**

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1 : Ortogonal ve perspektif projeksiyon gösterimi. **6**
Şekil 2.2 : Ortofoto üretiminin prensibi. **12**
Şekil 2.3 : Diferansiyel ortorektifikasyon iş akışı. **13**
Şekil 2.4 : Kamera koordinat sistemi ve IMU koordinat sistemi gösterimi. **16**
Şekil 2.5 : Kesişim noktasına rölyef kayma düzeltmesinin iteratif olarak getirilmesi **24**
Şekil 3.1 : Saniyedeki kayan nokta işlem miktarı bakımından GPU ve CPU'ların yıllara göre gelişimi. **34**
Şekil 3.2 : GPU ve CPU'ların band genişlikleri. **35**
Şekil 3.3 : CPU ve GPU'ların genel yapısı ve sahip oldukları transistör sayısındaki farklılık. **36**
Şekil 3.4 : Kernel iş parçacığının tanımlanması. **39**
Şekil 3.5 : CUDA iş parçacığı hiyerarşisi. **40**
Şekil 3.6 : Bloklara ayrılan kernel tanımlaması. **41**
Şekil 3.7 : Akış çoklu işlemcisi. **41**
Şekil 3.8 : Tek komutlu çoklu iş parçacığı mimarisi. **42**
Şekil 3.9 : GPU hafıza mimarisi. **43**
Şekil 3.10 : CUDA işlem akışı **45**
Şekil 4.1 : NVIDIA firması tarafından yapılan uygulamalarda elde edilen bazı sonuçlar. **47**
Şekil 4.2 : 3x3 boyutunda arama penceresi. **49**
Şekil 4.3 : Eğim yönü hesabı algoritması. **50**
Şekil 4.4 : Görüş analizi uygulamasının gösterim arayüzü. **50**
Şekil 4.5 : Gauss bulanık filtreleme uygulamasının arayüz örnekleri. **52**
Şekil 4.6 : Uygulamalarda kullanılan görüntülere örnek. **54**
Şekil 4.7 : Projektif dönüşüm ile ortorektifikasyon uygulamasının çıktı arayüzü. ... **55**
Şekil 4.8 : Ortorektifikasyon sonucu elde edilen görüntü örneği. **57**

GRAFİK İŞLEMCİ BİRİMİ ÜZERİNDE GENEL AMAÇLI HESAPLAMA YÖNTEMİ İLE GÖRÜNTÜLERİN GERÇEK ZAMANLI ORTOREKTİFİKASYONU

ÖZET

Bilgisayarların grafik kartları üzerindeki grafik işlemci birimleri (Graphic Processing Units – GPU) on sene öncesine göre, özellikle performans ve yeteneklerinin artışı doğrultusunda büyük gelişme göstermiştir. Modern GPU'lar sadece çok güçlü grafik motoru olmaktan çıkarak, 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 ise; 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 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.

Çeşitli platformlardan elde edilen görüntülerden harita üretimi ve özellikle görüntü değerlendirme sonuçlarının hızla elde edilmesi gereken felaketler ve doğal afetler gibi kriz durumları, askeri açıdan hedef istihbaratı, hızlı bir biçimde hedef konumunun tespiti gibi işlemleri yapmadan önce yapılması gereken en önemli işlem adımı; görüntülerin ortorektifikasyonudur. Özellikle insansız hava araçlarından çok

hızlı olarak hedef tespit edebilmek ve bu hedefleri de doğru koordinatlarla belirleyebilmek için elde edilen görüntülerin ortorektifiye edilmesi ihtiyacı bulunmaktadır.

Sayısal hava kameraları ve insansız hava araçları gibi çeşitli sensör ve platformlardan elde edilen görüntülerin boyutları oldukça yüksektir. Haliyle bu görüntülerin ortorektifikasyonu için çok güçlü bilgisayarlara ihtiyaç vardır. Yapılan çeşitli çalışmalar incelendiğinde görüntülerin anlık georektifikasyonu için çok sayıda güçlü donanıma sahip bilgisayarın kullanıldığı görülmektedir. Bu türden sistemlerin normal uçak platformları için oluşturulabileceği, maliyet göz önüne alınmazsa mümkün olabilir.

Fakat insansız hava araçları gibi yapı itibariyle normal uçaklardan çok daha küçük olan ve görüntü alımı gerçekleştirerek çok hızlı ve doğru sonuç elde edilmesi gereken platformlarda ise böyle çoklu bilgisayar sistemlerinin kurulabilmesi pek olanaklı değildir. Aynı zamanda bu sistemlerde platform üzerindeki kamera, navigasyon sistemi gibi faydalı yük olarak ifade edilen donanımın hafif olması da önemlidir.

Mevcut sayısal hava kameraları ve insansız hava araçlarındaki görüntüleme sistemlerinin üzerinde GPS-IMU (Global Positioning System-Inertial Measurement Unit) sistemi vardır. Haliyle kamera sistemi ile senkronize olarak yapılan ölçümler ve kamera kalibrasyon değerleri ile doğrudan ortorektifikasyon yapılabilir. İşte bu doğrudan ortorektifikasyon noktasında karşımıza GPGPU yöntemi çıkmaktadır.

Ortorektifikasyon işlemi GPGPU yöntemi ile CUDA (Compute Unified Device Architecture-Birleşik Hesaplama Aygıt Mimarisi) programlama dili kullanılarak yeniden programlanarak, daha kısa sürede ve daha ucuz maliyetli donanımlarla ortorektifiye edilmiş görüntülerin elde edilebileceği değerlendirilmiştir.

Bu amaçla tez kapsamında öncelikle literatür araştırması yapılmıştır. Sonrasında kullanılacak yöntemlerin matematiksel modelleri ortaya konulmuştur. Ardından kullanılacak yöntem açıklanmıştır. Bu aşamanın ardından da uygulama aşamasına geçilerek çeşitli uygulamalar gerçekleştirilmiş ve sonuçlar elde edilmiştir.

Çalışma içerisinde yapılan uygulamalardan görüleceği gibi, 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.

Yapılan uygulamalarda 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. Özellikle ortorektifikasyon amacıyla yapılan uygulamalarda aynı donanımla CPU'ya oranla 7 kat hız farklarına ulaşılmıştır.

Sonuçlar biraz daha detaylı incelendiğinde görüntü boyutu büyüdükçe CPU'nun performansının düştüğü, GPU'nun ve GPGPU yönteminin performansının arttığı 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 yöntemlerin uygulanabilir olduğu görülmektedir.

Yapılan bu çalışma ile daha önce çeşitli platformlardan elde edilen görüntülerin ortorektifikasyonu için kullanılmamış olan bir yöntem, donanım altyapısı kullanılmış ve programlama algoritması uygulanmıştır. Çalışma içerisinde gerçekleştirilen uygulamalar ile GPGPU yöntemi kullanılarak görüntülerin çok hızlı ve donanım olarak çok daha ekonomik bir şekilde ortorektifikasyonunun yapılabileceği değerlendirilmektedir. Böylece, bu çalışmayla halen anlık karar verme ve hedef tespiti için çok önemli bir unsur olan zaman faktörünün en aza indirilmesi için bir yöntem ortaya konulmuş ve bu yöntemin geliştirilmesi için de bir başlangıç yapılmıştır.

REAL TIME ORTHORECTIFICATION OF IMAGES BY GENERAL PURPOSE COMPUTATION ON GRAPHICAL PROCESSING UNITS METHOD

SUMMARY

The graphic processing units (GPU) on the graphic cards integral parts of computers are really developed today according to the last ten years. The development was the increase of the GPUs performance and capabilities. The modern GPUs are not only became powerful graphic engines and also they are high level parallel programmable processors with very fast computing capabilities and high memory bandwidth speed comparing to central processing units (CPU). The rapid development of GPUs programmability and capabilities attracted the researchers dealing with complex problems who need highly level calculation. 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 is 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 has 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.

General purpose parallel programming can use GPUs not only for graphics but also for removing the burden of the non-graphic computational workload which is traditionally handled by a CPU. Significant computational speedups have been achieved by various researchers from different disciplines using general purpose parallel programming. Although GPU-based non-graphics computation is well suited to data-parallel tasks such as image processing kernels and matrix operations, it is also possible to accelerate many other applications by adapting existing algorithms to the general purpose parallel programming. Therefore it seems reasonable to exploit

tremendous computing power of GPUs for orthorectification, since computational power is an important concern.

Real time processing of imagery data which is coming from several sensors and platforms like airborne and unmanned air vehicles is really important today. For rapid evaluating data coming from unmanned air vehicles (UAV) in military applications, for supporting rescue and security forces, and also for obtaining surveys in disaster scenarios or mass events an airborne real time image processing system is required. So the need is speed for processing orthorectification the imagery data to find the target and get true coordinates of targets from this quick orthorectified imagery.

The imagery data which is coming from several sensors and platforms size is really huge. Naturally, very powerful computers are needed to orthorectify of these images. When the various studies are analyzed, it can be seen that numerous powerful computers are used to orthorectify these imagery in real time. Such systems can be created for normal aircraft platforms mey be possible if considering the cost. However, in unmanned air vehicles, which are much smaller than regular aircrafts, it is not possible to establish that kind of multiple computer systems. At the same time it is important that the payloads on the platform as camera, navigation system etc. must be light hardware.

Existing digital aerial cameras and unmanned air vehicles imaging systems have GPS-IMU (Global Positioning System-Inertial Measurement Unit) systems. So it is possible to orthorectify images with camera calibration parameters in real time. With this direct orthorectification oppurtunity we encounter GPGPU method and it can be possible that orthorectification procedure can be done by GPGPU and CUDA (Compute Unified Device Architecture) programming language. So we can orthorectify images with cheaper hardware in a short time.

For this purpose in this thesis firstly a literatur search has been done. After that, mathematical models of the methods have demonstrated. Next it is described methods to be used. After this step the method has been implemented and taken the application results.

It can be seen from the studies in this thesis, the effective GPU programming requires the re-writing existing program algorithm using graphical terms again considering to hardware structure and limitations. Today, the multicore processors can not be programming using traditional programming methods. So the usage of typical event programming procedure can not be possible for programming the multicore processors.

Programming model changed to stream computing and processing. In this new model for identifying the kernel functions, that apply intensive calculation each element in the flow, all the input and output data qualified as stream. There are lots of processors on the GPU that process these streams.

For example Nvidia GTX580 series card has 512 unit stream processors (CUDA processors). So we can consider such as 512 computers stay side by side. The graphic cards can do multiple intensive processes with these stream processors at the same time.

Data-parallel processing maps data elements to parallel processing threads. Many applications that process large data sets can use a data-parallel programming model to speed up the computations. In 3D rendering, large sets of pixels and vertices are

mapped to parallel threads. Similarly, image and media processing applications such as post-processing of rendered images, video encoding and decoding, image scaling, stereo vision, and pattern recognition can map image blocks and pixels to parallel processing threads. In fact, many algorithms outside the field of image rendering and processing are accelerated by data-parallel processing, from general signal processing or physics simulation to computational finance or computational biology.

In this study covers how general purpose parallel programming and computational power of the GPUs and GPGPU method can be used in photogrammetric orthorectification applications especially direct georeferencing and projective rectification. These two methods coded with CUDA programming language.

The results obtained are evaluated; the method is really suitable for image processing and photogrammetry especially if we do the same calculations to per image pixels. Also it is suitable for intensive calculation procedures. GPGPU and CUDA programming method make the calculation really fast. We can increase the number of applications which can be adapted to photogrammetry and image processing that require intensive computation and speed. Especially with orthorectification procedure with GPU is 7 times faster than CPU implementation and speedup is 7 times. The results are analyzed in detail when the image size is slightly bigger, GPGPU method and GPU performance is increase and CPU performance is decrease.

In this study, a method which has not used before for orthorectification of images obtained from various platforms was implemented using programming algorithm and hardware infrastructure. By using this GPGPU method it can be possible to orthorectify images very fast and with economical hardwares. So this study is a start point for improve this method and increase the time factor for making instant decisions from images.

1. GİRİŞ

Çeşitli platformlardan elde edilen görüntüler harita üretiminde, özellikle görüntülerin değerlendirme sonuçlarının hızla elde edilmesi ve karar vericilere aktarılmasının gerektiği felaketlerde, orman yangınları ve depremler gibi doğal afetlerde yaşanan kriz durumlarında, askeri açıdan hedef istihbaratı ve hedef konumunun hızlıca tespitinde sıklıkla kullanılmaktadırlar. Fakat bu görüntülerin anlamlı hale gelmesi ve belirtilen uygulamalarda kullanılabilmesi için öncelikli olarak yapılması gereken işlem adımı; bu görüntülerin ortorektifikasyonudur. Özellikle son zamanlarda yaygınlaşarak çeşitli kurum ve kuruluşlar tarafından birçok uygulamada kullanılmaya başlanan insansız hava araçları (İHA) yardımıyla elde edilen görüntüler üzerinden çok hızlı karar vermek, çıkarımlar yapmak ve çeşitli hedef tespiti yaparak bu hedefleri de doğru koordinatlarla tarifleyebilmek için elde edilen görüntülerin yöneltilmesi yani ortorektifiye edilmesi ihtiyacı bulunmaktadır.

Sayısal hava kameraları ve insansız hava araçları gibi çeşitli sensör ve platformlardan elde edilen görüntülerin dosya boyutları oldukça yüksektir. Haliyle bu görüntülerin ortorektifikasyonu için çok güçlü bilgisayarlara ihtiyaç vardır. Thomas ve diğ. (2008) tarafından yapılan bir çalışma incelendiğinde görüntülerin anlık ortorektifikasyonu için toplam beş adet çok güçlü donanıma sahip bilgisayarın kullanıldığı görülmektedir. Böyle bir sistemin normal uçak platformları için oluşturulabileceği, maliyet göz önüne alınmazsa mümkün olabilir. Fakat insansız hava araçları gibi yapı itibariyle normal uçaklardan çok daha küçük olan ve görüntü alımı gerçekleştirerek çok hızlı ve doğru sonuç elde edilmesi gereken platformlarda ise böyle çoklu bilgisayar sistemlerinin kurulabilmesi pek olanaklı değildir. Aynı zamanda bu sistemlerde platform üzerindeki kamera, navigasyon sistemi gibi faydalı yük olarak ifade edilen donanımın hafif olması da son derece önemli bir faktördür. İnsansız hava araçları üzerine eklenen her bir parça sisteme ağırlık olarak etki etmekte ve bu da sistemi olumsuz yönde etkilemektedir.

Çeşitli platformlar üzerinde yer alabilen tarayıcı sistemlerle elde edilen görüntülerin yöneltilmesi amacıyla her tarama çizgisi için konum ve dönüklük verilerine ihtiyaç

duyulmaktadır. Haliyle bu sistemlerden elde edilen görüntülerin yöneltilmesi probleminde klasik dolaylı yöntem olarak isimlendirilen yaklaşımın kullanımı çok sayıdaki kontrol noktası ihtiyacı sebebiyle hemen hemen imkânsızdır (Schwarz ve diğ, 1993). Bu sebeple bu tür görüntülerin yöneltilmesi için çalışmalar yer kontrol noktası kullanmaksızın 6 adet dış yöneltme bilinmeyeninin doğrudan belirlenmesi üzerine yoğunlaşmıştır. "Inertial Measurement Unit" (IMU) ve "Differential Global Positioning System" (DGPS)'in birlikte kullanılması ile elde edilen GPS/IMU sistemi yardımı ile dış yöneltme elemanları doğrudan belirlenebilmektedir (Yastıklı, 2003).

"Doğrudan Yöneltme" (Direct Georeferencing) olarak isimlendirilen ve GPS/IMU sistemleri yardımı ile dış yöneltme elemanlarının doğrudan belirlendiği bu teknoloji; klasik film kameralar, lazer tarayıcı sistemler (LIDAR), yapay açıklıklı radar (SAR), Interferometrik SAR (InSAR), sayısal dizi tarayıcılar, çok bantlı tarayıcılar ve sayısal kameralarla görüntü kaydeden platformlardaki uzaktan algılama sistemlerinde kullanılmaktadır (Skaloud, 2002).

GPS/IMU sisteminin temeli navigasyon sistemlerine dayanmaktadır. Özellikle "Inertial Navigation Systems" (INS), askeri ve bilimsel uygulamaların önemli bir parçası olmuştur (Farrel ve Barth, 1999). INS sistemleri, IMU sistemini ölçme birimi olarak içermekte ve buna ek olarak konum ve yönlendirme fonksiyonları genellikle bir yazılımla gerçekleştirilmektedir (Colomina, 1999).

INS sistemi ilk kez 1949 yılında C.S.Draper tarafından geliştirilmiştir (Schherzinger, 2001). Daha sonraki yıllarda, Amerika Birleşik Devletleri Hava Kuvvetleri askeri navigasyon uygulamalarında kullanılmak üzere standart GPS/INS üretimini gerçekleştirmiş ve 1970'li yıllarda INS bir ölçme aleti olarak görülmeye başlanmıştır. Bu yıllarda uçaklardaki uzaktan algılama sistemlerinde dönüklük açılarının tespiti için INS kullanılmıştır. Sonraki yıllarda teknolojinin gelişimi ile lazer jiroskoplar üretilmiş ve INS hantal ve ağır sistemler olmaktan çıkmıştır. Bugünkü anlamda IMU'nun ilk üretim amacı silah sistemlerinin ve özellikle kısa menzilli füze sistemlerinin yönlendirilmesiydi. Sonuç olarak, GPS ve IMU'nun birlikte kullanılması ile uzaktan algılama uygulamalarında doğrudan yöneltme yaklaşımının geliştirilmesi mümkün olmuştur (Yastıklı, 2003).

İlk deneysel çalışmalara Ohio State Üniversitesi'nde 1980'li yılların sonunda başlanarak, yerde hareket halindeki araç üzerinde bulunan algılama sistemi yardımıyla doğrudan görüntü elde edilmesi üzerinde çalışmalar yapılmıştır. Ardından 1990'lı yılların başında Calgary Üniversitesi Coğrafi Bilgi Sistemi uygulamaları için karada hareketli ölçme sistemi VISAT (**V**ideo cameras, an **I**nertial System, and **SAT**ellite GPS receivers) 'ı geliştirmiştir. Bu deneysel çalışmalarla birlikte doğrudan yöneltme yaklaşımı, hareket halindeki algılayıcının konum ve dönüklük verilerinde oluşan dış yöneltme elemanlarının belirlenmesi üzerine yoğunlaşmıştır (Schwarz ve diğ, 1993; Schwarz 1995). Günümüze kadar yapılan deneysel çalışmalar ve test projeleri, GPS/IMU verilerinin sinyal işleme teknikleri kullanılarak iyileştirilmesi ve bu verilerin Kalman Filtreleme yöntemi ile birleştirilmesi, klasik fotogrametrik kameralar kullanılarak doğrudan yöneltme yaklaşımı, sayısal kameralar kullanılarak doğrudan yöneltme yaklaşımı ve uçakta kullanılan tarayıcı sistemler ile doğrudan yöneltme yaklaşımı konularında yapılmıştır (Yastıklı, 2003).

Amerika Birleşik Devletlerinde 11 Eylül 2001 yılında meydana gelen terörist saldırı sonrası, arama-kurtarma çalışmalarını yönlendirmek amacıyla hızlıca bölgenin haritalarının üretimi için bu yöntemden faydalanılmıştır (Yastıklı, 2003). Yapılan bu çalışma; özellikle hızlı ve acil müdahalelerin gerektiği afet durumunda, afet bölgesi ile ilgili hızlıca yorum yapabilmek ve karar verme sürecini hızlandırmak amacıyla, farklı sensör ve platformlardan elde edilen veri ve görüntülerin hiç yer kontrol noktası olmadan GPS/IMU verileriyle doğrudan yöneltilerek kullanılması ve böylelikle yöntemin geçerliliği açısından çok iyi bir örnek teşkil etmektedir.

1.1 Tezin Amacı ve Kapsamı

Bilgisayarların grafik kartları üzerinde yer alan grafik işlemci birimleri (Graphic Processing Units – GPU) on sene öncesine göre, özellikle performans ve yetenek artışı doğrultusunda çok büyük gelişme göstermişlerdir. Modern GPU'lar sadece çok güçlü grafik motoru olmaktan çıkarak, 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 programlanabilmesi 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 ise; 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.

GPU'lar 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.

Bu kapsamda; GPU'ların genel amaçlı paralel programlama ve hesaplama gücünü kullanarak GPGPU yöntemiyle sayısal hava kameraları, insansız hava araçları gibi çok çeşitli platformlardan elde edilebilecek görüntülerin hızlı bir şekilde gerçek zamanlı ortorektifikasyonunun yapılabilmesi için, yöntem detaylı bir şekilde incelenecek, bir program algoritması ortaya çıkarılarak uygulanabilirliği değerlendirilecektir. Böylece daha önce görüntülerin ortorektifikasyonu için kullanılmamış olan bir yöntem ve programlama mantığı uygulanmaya çalışılacak, ortaya çıkacak sonuç ile yüksek kapasitedeki farklı platformlardan elde edilen görüntülerin çok hızlı ve donanım olarak çok daha ekonomik bir yöntemle ortorektifikasyonunun yapılıp yapılamayacağı ortaya konulmaya çalışılacaktır. Yapılan çalışmayla, halen anlık karar verme ve hedef tespiti için çok önemli bir unsur olan zaman faktörü en aza indirilmeye çalışılacaktır.

2. ORTOREKTİFİKASYON

Yöneltme, algılama sistemiyle elde edilen görüntünün koordinat sistemi ile sonuç ürünün sunulacağı nesne koordinat sistemi arasındaki dönüşüm işlemi olarak tanımlanabilir. Hava fotogrametrisinde yöneltme problemi, resim koordinat sistemindeki resim koordinatlarından referans koordinat sistemine dönüşüm problemi olarak tanımlanabilir. Referans koordinat sistemi bu durumda genellikle ülke koordinat sistemidir. Bu dönüşüm problemi (2.1) denklemi ile formüle edilebilir.

$$\begin{pmatrix} X_P \\ Y_P \\ Z_P \end{pmatrix}_r = \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix}_r + \alpha R_f^r \begin{pmatrix} x_p \\ y_p \\ -f \end{pmatrix}_f \tag{2.1}$$

Bu denklemde, (X_P, Y_P, Z_P) referans koordinat sistemindeki koordinatları, (x_p, y_p) P'ye karşılık gelen bir P' noktasının fotoğraf koordinat sistemindeki koordinatlarını, f odak uzaklığını, α ölçek katsayısını ve R_f^r iki koordinat sistemi arasındaki dönüşüm matrisini ifade etmektedir. Bu eşitliğin çözümü ile izdüşüm denklemleri elde edilir. Bu denklemlerin çözümü için, fotogrametrik kameranın odak uzaklığı ve asal noktanın konumundan (f, x_0, y_0) oluşan iç yöneltme parametreleri ve resim çekme noktasının konumu (X_0, Y_0, Z_0) ve dönüklük açılarından $(\omega, \varphi, \kappa)$ oluşan dış yöneltme parametrelerinin bilinmesi gerekir. Dış yöneltme parametreleri, hava fotogrametrisinde resim bloğundaki tüm resimler için birlikte fotogrametrik nirengi yöntemi ile ve en küçük kareler yaklaşımı ile bulunur.

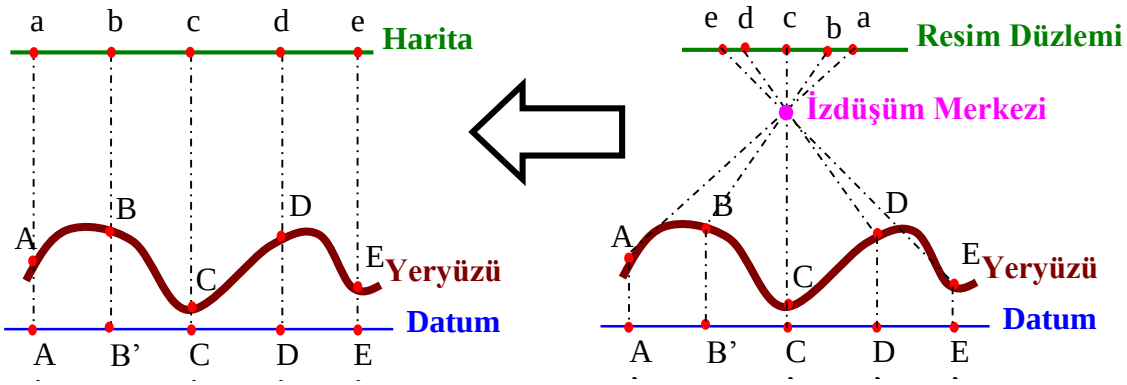
Buradaki temel prensip, kullanılan algılayıcı tipine bağlı olarak kayıt anında konum ve dönüklük bilgilerinin referans koordinat sisteminde belirlenmesidir. Kullanılan navigasyon sistemleri, konum ve dönüklük verilerinden oluşan bu bilgileri yeterli doğrulukta sağladığı takdirde bu amaç için kullanılabilir. Navigasyon sistemlerinin bu amaçla kullanılabilmesi için aşağıdaki üç şartı sağlaması gerekir (Skaloud, 1999):

- Konum ve dönüklük verileri için navigasyon koordinat sistemi ile görüntü koordinat sistemi arasındaki ötelemenin yeterli doğrulukta belirlenmesi,

- Belirlenen bu ötelemenin sabit olması ve değişimlerin modellenebilmesi
- Bu amaçla kullanılan algılayıcıların yeterli doğrulukta senkronize edilmesidir.

Burada kullanılan yeterli doğruluk ifadesi, doğrudan ölçülen dış yöneltme elemanlarının doğruluğunun bağlı olduğu navigasyon sisteminin doğruluğu yanında algılayıcılar arasındaki ötelemelerin belirlenmesi ve senkronizasyon doğruluğunu da içermektedir (Yastıklı, 2003).

Dijital resimler ve uydu görüntüleri, resim çekme makinaları ve sensörlerin eğikliği ve arazinin rölyef etkisi nedeniyle detayları gerçek konumlarında göstermezler. Ortorektifikasyon, resmin merkezi izdüşümünü, yeryüzünün ortogonal bir görünümü haline dönüştürür. Ortorektifikasyon, işlenmemiş ham haldeki resim veya görüntüyü doğru ortogonal projeksiyona dönüştürmek için kullanılan bir işlemdir (Şekil 2.1). Ortorektifikasyon işlemi sonucunda elde edilen sonuç ürün bir ortofotodur. Ortorektifikasyon yapılmazsa şayet, resim ölçeklenemez ve üzerinden herhangi bir ölçüm işlemi yapılamaz.



- Ortogonal projeksiyon
- Sabit ölçek
- Rölyef hatası giderilmiş

- Perspektif projeksiyon
- Sabit olmayan ölçek
- Rölyef hatası mevcut

Şekil 2.1 : Ortogonal ve perspektif projeksiyon gösterimi.

Bir görüntünün ortorektifikasyonu için bir dönüşüm modeli gereklidir. Bu model resim çekme anında birçok sebepten kaynaklanan resim distorsiyonunu yansıtabilmelidir. Bunlar şu şekilde sıralanabilir:

- Sensör dönüklüğü,
- Topoğrafik rölyef etkisi,
- Dünyanın şekli ve dönüşü,

- Sensör yörüngesi ve davranışındaki farklılıklar,
- Sensöre bağlı olan sistematik hatalar.

Sensör dönüklüğü ile ilgili geometrik parametreler çekim esnasında, sensör modeli, yer kontrol noktaları ve platformun yörünge veya uçuş verisi (konum, hız, dönüklük) sayesinde elde edilebilmektedir.

Genel olarak yöneltme yaklaşımları iki sınıfta değerlendirilmektedir. Bunlar:

- Parametrik yaklaşımlar,
- Parametrik olamayan yaklaşımlar (Hemmleb ve Wiedemann, 1997).

Parametrik yaklaşımda iç ve dış yöneltme parametrelerine ihtiyaç duyulmaktadır. Parametrik olmayan yaklaşımda ise sadece kontrol noktaları gereklidir. Parametrik olmayan yaklaşımlar polinomal ve projektif dönüşümleri içermektedir. Bu konu ile ilgili ayrıntılı bir çalışma Novak (1992) de bulunabilir.

2.1 Polinomal Dönüşüm

Birçok görüntü işleme sisteminde de yer alan en basit yöntemdir. Basitçe yüzeyi bir polinomal fonksiyonla ifade edip, kontrol noktalarıyla polinomu gerçekleyen bir yöntemdir. Bu yöntemle sadece eğim etkisi düzeltilebilir ve hem uydu görüntülerine hem de hava resimlerine uygulanabilir. Eldeki mevcut yer kontrol noktalarının sayısına ve istenilen doğruluğa göre polinomun derecesi belirlenir. Rosenholm ve Akerman (1998) tarafından yapılan çalışmalar, düşük dereceli polinomların uydu görüntüleri için piksel altı doğruluğu sağlayabildiğini ortaya koymuştur. Novak (1992) polinomal yöneltmenin kolay uygulanabilir bir yöntem olduğunu, fakat bununla birlikte rölyef etkisinin düzeltilmesini sağlayamadığını vurgulamaktadır. Hemmleb ve Wiedemann (1997) yüksek dereceli polinom kullanmanın hem yer kontrol nokta sayısını çok fazla artırması nedeniyle, hem de hesaplamada düzensizlikler yapabileceğinden dolayı tehlikeli olduğunu söylemektedirler. Polinomal dönüşüm denklemleri aşağıdaki gibi verilebilir:

$$r = a_{j=0}^{n} a_{j=0}^{n-i} a_{ij} x^i y^j \tag{2.2}$$

$$c = a_{j=0}^{n} a_{j=0}^{n-i} b_{ij} x^i y^j \tag{2.3}$$

Bu denklemlerdeki *r* ve *c* girdi resmin piksel koordinatları (satır ve sütun); *x* ve *y* ise sonuç olarak hesaplanan görüntünün koordinatlarıdır. *a* ve *b* polinom katsayıları, *n* ise polinomun derecesini ifade etmektedir.

2.2 Projektif Dönüşüm

Projektif dönüşüm uygulanabilmesi için resim düzlemi ile izdüşüm düzlemi arasındaki geometrik dönüşüme ihtiyaç vardır. Projektif dönüşüm denklemindeki sekiz bilinmeyenin çözülebilmesi için obje düzleminde koordinatları bilinen en az dört kontrol noktası gereklidir. Projektif dönüşüm daha çok, özellikle arazi yüzeyinin düz olduğu alanlar, bina cepheleri gibi detayların yer aldığı hava resimlerinin yöneltilmesi için uygulanabilirdir ve rölyef etkisini ortadan kaldırmamaktadır (Novak, 1992). Projektif dönüşüm için denklemler aşağıdaki gibi verilebilir:

$$r = \frac{b_{11}x + b_{21}y + b_{31}}{b_{12}x + b_{23}y + 1} \tag{2.4}$$

$$c = \frac{b_{12}x + b_{22}y + b_{32}}{b_{13}x + b_{23}y + 1} \tag{2.5}$$

Bu denklemlerdeki *r* ve *c*, girdi resmin piksel koordinatları (satır ve sütun); *x* ve *y* ise sonuç olarak hesaplanan görüntünün koordinatlarıdır. b_{11}, b_{12}, b_{13}, b_{21}, b_{22}, b_{23}, b_{31}, b_{32} ve b_{33} katsayılardır.

2.3 Sensör Model Dönüşümü

Sensör modelleri resim uzayı ve obje uzayı arasında fonksiyonel bir ilişki kurmayı gerektirirler. Sensör modelleri tipik olarak iki kategoride sınıflandırılabilir; fiziksel modeller ve genelleştirilmiş modeller. Sensör modelinin seçimi tümüyle istenilen performans ve duyarlılığa, ayrıca resim çekme makinesi ve platformun kontrol bilgisinin varlığına bağlıdır (Tao ve Hu, 2001). Fiziksel bir sensör modeli, fiziksel resim çekme iş akışını gösterir. Parametreler sensörün konumunu ve dönüklüğünü obje uzay koordinat sisteminde tanımlamayı gerektirir. Kolinearite denklemleri gibi fiziksel modeller, analitik triyangülasyon dengelemesi için ve yüksek modelleme duyarlılığı için çok uygundur.

Çerçeve, Pushbroom, Panaromik, SAR vb gibi birçok tipte sensör mevcuttur. Fiziksel modeller, sensöre bağlı olan ve her bir sensör için farklı olan modellerdir. Fiziksel sensör modelleri matematiksel açıdan daha karmaşık yapıdadırlar ve daha uzun süren hesaplama zamanına ihtiyaç duyarlar.

Genelleştirilmiş sensör modelleri ve sensör tipleri çekici bir şekilde gelişmektedir. Genelleştirilmiş bir sensör modelinde resim ve obje uzayı arasındaki dönüşüm fiziksel resim çekme işlemini oluşturmadan, bazı genel fonksiyonlarla tanımlanmaktadır.

2.4 Rasyonel Fonksiyon Model Dönüşümü

Rasyonel fonksiyon model sensör modeli obje ve resim uzayı arasındaki geometrik ilişkiyi tanımlamaktadır. Obje nokta koordinatları (X, Y, Z), resim piksel koordinatları (r, c)'na bağlıdır, ya da tam tersi 78 adet rasyonel polinomal katsayılar kullanılır. Yeryüzünden resime dönüşüm için polinomal oranlar aşağıda verilen denklemlerde olduğu gibi ifade edilebilir (Croitoru vd., 2004):

$$r_n = \frac{p_1(X_n, Y_n, Z_n)}{p_2(X_n, Y_n, Z_n)} = \frac{a_{i=0}^{m1} a_{j=0}^{m2} a_{k=0}^{m3} a_{ijk} X_n^i X_n^j X_n^k}{a_{i=0}^{m1} a_{j=0}^{m2} a_{k=0}^{m3} b_{ijk} X_n^i X_n^j X_n^k} \tag{2.6}$$

$$c_n = \frac{p_3(X_n, Y_n, Z_n)}{p_4(X_n, Y_n, Z_n)} = \frac{a_{i=0}^{m1} a_{j=0}^{m2} a_{k=0}^{m3} c_{ijk} X_n^i X_n^j X_n^k}{a_{i=0}^{m1} a_{j=0}^{m2} a_{k=0}^{m3} d_{ijk} X_n^i X_n^j X_n^k} \tag{2.7}$$

Bu denklemlerdeki (r_n,c_n) piksellerin resim uzayındaki normalleştirilmiş satır ve sütun indeksleridir; (X_n, Y_n, Z_n) obje noktasının obje uzayındaki normalleştirilmiş koordinat değerleridir; a_{ijk}, b_{ijk}, c_{ijk} ise rasyonel fonksiyon katsayılarıdır.

2.5 Diferansiyel Ortorektifikasyon

Günümüzde GPS ve IMU'nun birlikte kullanılması ile oluşan GPS/IMU sistemlerinin fotogrametri ve uzaktan algılamada kullanımı son derece yaygınlaşmıştır. GPS/IMU, stereo model oluşturulmasında ve resim yöneltmede fotogrametrik nirengi gereksinimi için büyük bir esneklik getirmektedir. Dış yöneltme elemanları (X_0, Y_0, Z_0, ω, φ, κ), blok köşelerinde 4 adet yer kontrol noktası ve bindirme alanlarında homojen dağılmış bağlama noktaları kullanılarak

fotogrametrik nirengi işlemi ve ışın demetleri ile blok dengeleme yapılarak belirlenmektedir. Yer kontrol nokta (YKN) sayısının azaltılması, fotogrametrik çalışmaların temel amaçları arasındadır. INS (Inertial Navigation System) ve DGPS (Differential Global Positioning System)'in birlikte kullanılmasıyla oluşturulan GPS/INS sistemi kullanılarak uçuş sırasında gerçekleştirilen ölçümler sonucunda dış yöneltme elemanları doğrudan hesaplanarak elde edilebilmektedir. Doğrudan yöneltme ile oluşturulan ortogörüntüler, hızlı bir şekilde elde edilebildiğinden birçok uygulamada altlık olarak kullanılmaktadır. Doğrudan yöneltme yöntemi, görüntü üzerindeki piksel konumu ve bunlara karşılık gelen yeryüzündeki konumları arasındaki eş doğrusallık (kolinearite) eşitliklerine dayanmaktadır. GPS/INS entegrasyonu ile elde edilen dış yöneltme parametrelerinin doğrudan kullanılabilmesi, ortofoto oluşturma işlemini zaman açısından oldukça kısaltacak ve maliyeti azaltacaktır. Doğrudan GPS/INS parametrelerinin kullanılması istenilen konum doğruluna bağlı olarak YKN ihtiyacını azaltmakta ve blok dengeleme işlemine ihtiyaç duyulmamaktadır. Doğrudan yöneltme, bir görüntünün piksel bazında ortogonal projeksiyona dönüşümüdür.

Diferansiyel ortorektifikasyon yönteminde temel husus, resim üzerindeki gri değerlerinin ortofoto üzerindeki her bir piksele atanmasıdır. Novak (1992) doğrudan yöneltmenin rölyef hatasını ve resim çekme makinası distorsiyon hatasının her ikisini birden en iyi değerlerle düzeltebildiğine, hava resimleri ve uydu görüntüleri için uygulanabilir olduğuna vurgu yapmıştır.

Diferansiyel ortorektifikasyon denklemi şu şekilde verilebilir:

$$x = x_0 - c\frac{r_{11}(X - X_0) + r_{12}(Y - Y_0) + r_{13}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)} \tag{2.8}$$

$$y = y_0 - c\frac{r_{12}(X - X_0) + r_{22}(Y - Y_0) + r_{32}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)} \tag{2.9}$$

Bu denklemlerde:

- (x, y) objenin resim uzayındaki nokta koordinatları,
- (x_0, y_0) resim ana noktasının resim koordinatları,
- c resim çekme makinesi sabiti,

- (X_0, Y_0, Z_0) resim çekme merkezinin koordinatları,
- $(\omega, \varphi, \kappa)$ resim ile obje koordinat sistemi arasındaki dönüklük matrisinin elemanları olarak tanımlanmaktadır.

Diferansiyel ortorektifikasyon işleminin adımları şu şekilde verilebilir (Mayr ve Heipke, 1988):

- Ortofoto düzlemi üzerinde sabit aralıklı bir grid tanımlanır,
- Ortofoto düzleminde bulunan her bir grid elemanı (X, Y) için, enterpolasyonla $Z(X, Y)$ yükseklik değeri belirlenir,
- İç yöneltme parametreleri ve dış yöneltme parametreleri her ikisi birden kullanılarak, kolinearite denklemi ile resim noktası (x, y) koordinatları bulunur.
- Yeniden örnekleme teknikleri kullanılarak (x, y) ye karşılık gelen gri değeri bulunur.
- Yukarıdaki iş akışı ortofoto düzleminde bulunan her bir piksel için tekrarlanır.

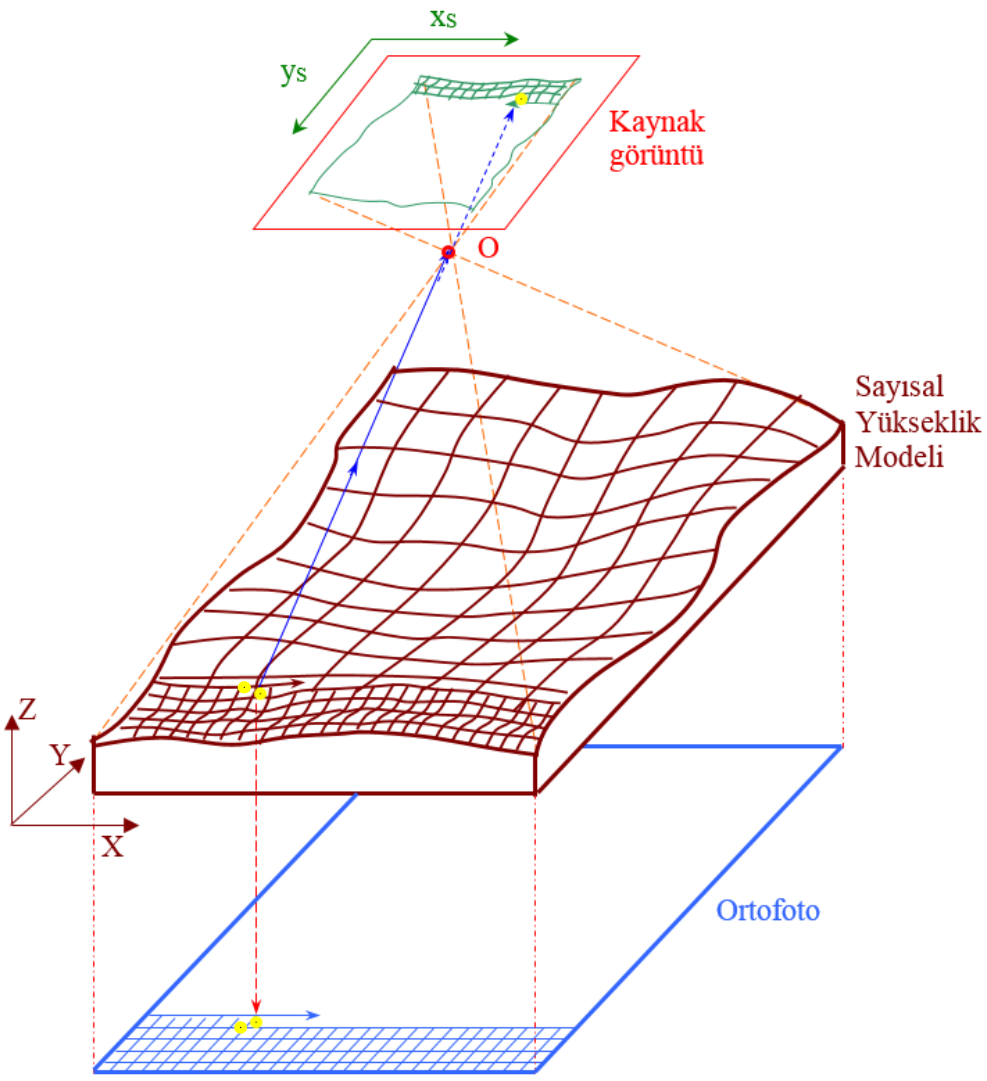
İşlem adımlarında yer alan en son maddeye dikkat edilecek olursa; bu madde, bütün bu iş akışının her bir piksel için ayrı ayrı tekrarlandığını söylemektedir. Bu da aslında mevcut problemin ve uygulamanın ileriki bölümde açıklanacak olan CUDA (Birleşik Hesaplama Aygıt Mimarisi - Compute Unified Device Architecture) ile programlama için uygun olduğunu göstermektedir. Şayet bu tekrar işlemi parçalara bölünüp aynı anda birçok işlemciye yaptırılabilirse, sonucun doğal olarak daha hızlı elde edilmesi beklenecektir.

Diferansiyel ortorektifikasyon, kaynak görüntünün ortogonal bir projeksiyonla piksel bazında referans düzlemine izdüşümü olarak kısaca ifade edilebilir (Şekil 2.2). Görüntüdeki pikseller ve yeryüzü üzerinde karşılık geldiği konumlar arasındaki ilişki iki yaklaşımla belirlenebilir:

- Doğrudan yaklaşım (kaynak görüntüden → ortofotoya)
- Dolaylı yaklaşım (ortofotodan → kaynak görüntüye)

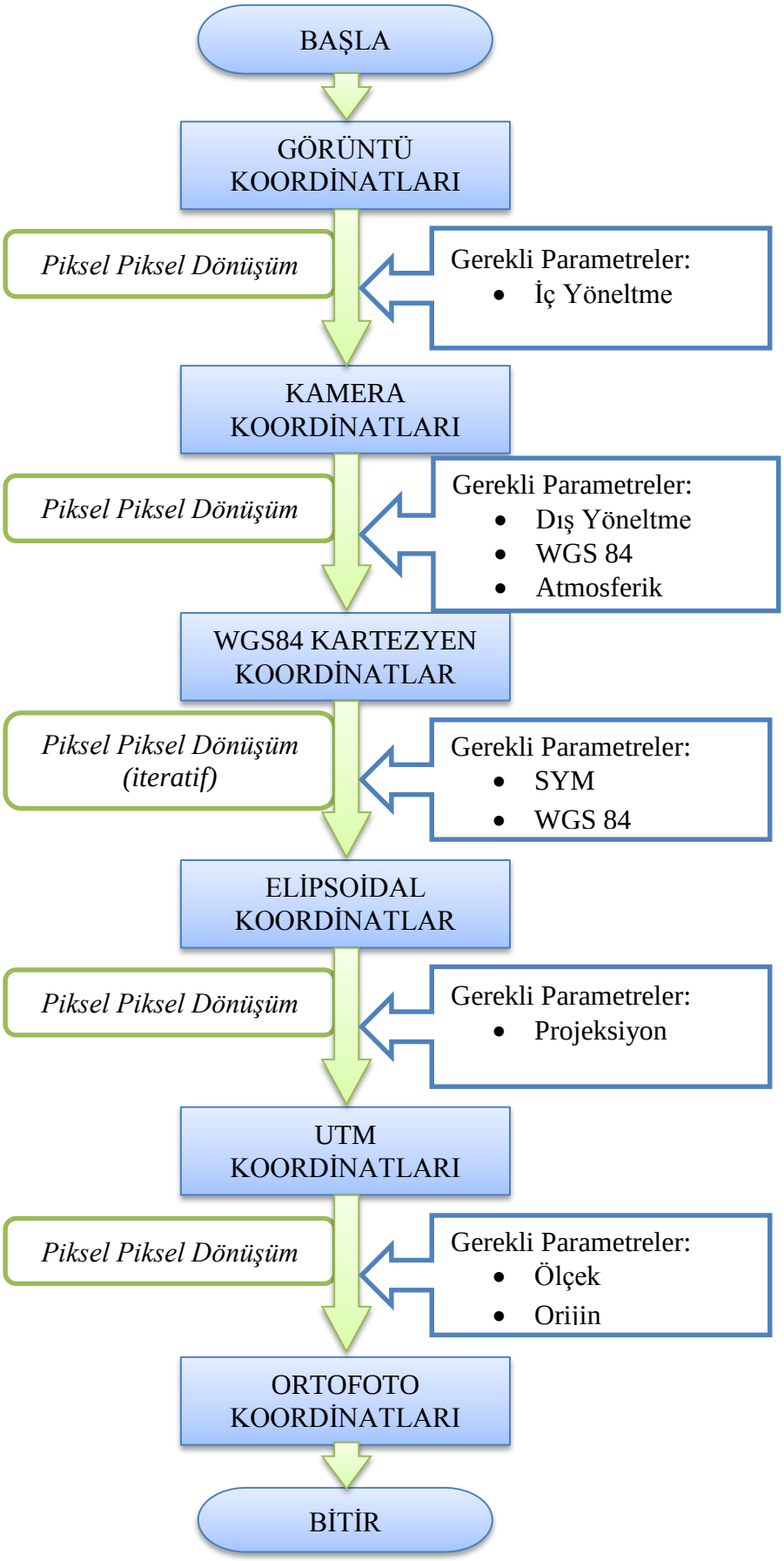
Doğrudan yaklaşımda kaynak görüntüdeki piksel koordinatları kullanılarak izdüşüm denklemleri ile yeryüzünde karşılık geldiği koordinatlar hesaplanır ve yeniden

örnekleme ile ortofoto elde edilir. Dolaylı yaklaşımda ise ortofotonun sınırları belirlenir ve bu belirlenen sınırlar boyutunda bir grid ağı oluşturulur. Belirlenen grid ağındaki koordinatlara karşılık gelen kaynak görüntüdeki pikseller ters dönüşüm ile belirlenir.



Şekil 2.2 : Ortofoto üretiminin prensibi.

Diferansiyel ortorektifikasyon yöntemi resim üzerindeki piksel konumları ve bunlara ait yeryüzündeki konumları arasındaki ilişkiyi belirleyen kolinearite eşitliklerine dayanmaktadır. Diferansiyel ortorektifikasyon için tasarlanan algoritmada öncelikle iç yöneltme elemanları ve her bir piksel için elde edilen üç boyutlu görüntü vektörü (yön vektörü) dış yöneltme elemanları kullanılarak referans elipsoidi üzerine izdüşürülür, sayısal yükseklik modeli kullanılarak elipsoid üzerindeki yeni konumu hesaplanır ve yeniden örnekleme yöntemi ile de ortofoto elde edilir. Algoritmanın genel akış diyagramı Şekil 2.3'deki gibi verilebilir.



Şekil 2.3 : Diferansiyel ortorektifikasyon iş akışı (Karslıoğlu vd, 2005).

2.5.1 Görüntü Koordinat Sisteminden Resim Koordinat Sistemine Dönüşüm

Algoritmanın ilk adımı piksel cinsinden görüntü koordinat sisteminde (C_{IM}) ifade edilen görüntü koordinatlarının, mm cinsinden tanımlanan resim koordinat sistemine (C_P) dönüşümü ile başlar. Bu dönüşümde koordinat sisteminin merkezi sol üst köşeden, CCD dizisinin tam ortasına, CCD dizinin boyutunun yarısı kadar taşınır ve aynı zamanda CCD merkezinin ana noktasının Δx ve Δy kadar düzeltmesi de yapılır. Bu dönüşüm için aşağıdaki denklemlerden faydalanılır.

$$x' = \left(x'' - \frac{width}{2} \right) * c - \Delta x \tag{2.10}$$

$$y' = \left(-y'' + \frac{heigth}{2} \right) * c - \Delta y \tag{2.11}$$

Bu denklemlerde:

- x'' ve y'' digital görüntüdeki kartezyen piksel koordinatlarıdır (piksel),
- *width* ve *height* görüntünün genişliği ve yüksekliğidir (piksel),
- Δx ve Δy resim ana noktasının x ve y görüntü koordinatlarıdır (mm),
- c ise görüntü ölçeğidir (mm/piksel),
- x' ve y' resim koordinat sistemindeki koordinatlar (mm).

Bu denklemler sonucunda elde edilen koordinatlar aslında olması gereken doğru koordinatlar değildir. Bunun nedeni ise kamera sistemindeki distorsiyon hatasıdır. Bu hatanın giderilebilmesi için polinomal bir denklemden faydalanılır. Bu polinomal denklemler aşağıdaki gösterilmiştir. Böylelikle düzeltilmiş x ve y kamera koordinat sistemindeki koordinat değerlerine ulaşılır.

$$x = x' * \left(1 - k_1 * r^2 - k_2 * r^4 - 2 * p_1 * x' - 2 * p_2 * y' - \frac{p_1 * r^2}{x'} \right) \tag{2.12}$$

$$y = y' * \left(1 - k_1 * r^2 - k_2 * r^4 - 2 * p_2 * y' - 2 * p_1 * x' - \frac{p_2 * r^2}{y'} \right) \tag{2.13}$$

$$z = -f \tag{2.14}$$

$$r = \sqrt{x'^2 + y'^2} \tag{2.15}$$

Bu denklemlerde:

- r ilgili pikselin CCD dizini üzerindeki resim ana noktasından olan mesafesi (mm),
- x' ve y' kamera koordinat sistemindeki koordinatlar (mm),
- x,y ve z değerleri distorsiyon hatası giderilmiş görüntü koordinatları (mm),
- k_1 ve k_2 radyal distorsiyon parametreleri,
- p_1 ve p_2 teğetsel distorsiyon parametreleri,
- f kameranın odak uzaklığı.

2.5.2 Resim Koordinat Sisteminden Kamera Koordinat Sistemine Dönüşüm

Resim koordinat sisteminden kamera koordinat sistemine dönüşüm, kamera odak uzaklığı ve resim koordinat sistemindeki piksellerin koordinatları kullanılarak yapılabilir. Koordinat sisteminin merkezi resim ana noktası olmuştur. Bu sebepten dolayı görüntü üzerindeki tüm noktaların z koordinatı $-f$ değerini almaktadır. Her bir piksel için kamera koordinat sistemine bağlı yön vektörü aşağıdaki denklem ile hesaplanabilmektedir.

$$\overset{C}{D} = [x, y, z]' / \sqrt{x^2 + y^2 + z^2} \tag{2.16}$$

Bu denklemde:

- x,y değerleri düzeltilmiş resim koordinat sistemindeki piksellerin koordinatlarıdır,
- z değeri ise kamera odak uzaklığının negatif değeridir.

2.5.3 Kamera Koordinat Sisteminden IMU Koordinat Sistemine Dönüşüm

Kamera koordinat sistemindeki koordinatlar, IMU koordinat sistemine dönüştürülmelidir (Şekil 2.4). Bu amaca yönelik olarak kamera koordinat sistemi X ekseninde 180°, Z ekseninde 90° döndürülmelidir. Bu ise aşağıdaki denklemle gerçeklenmektedir.

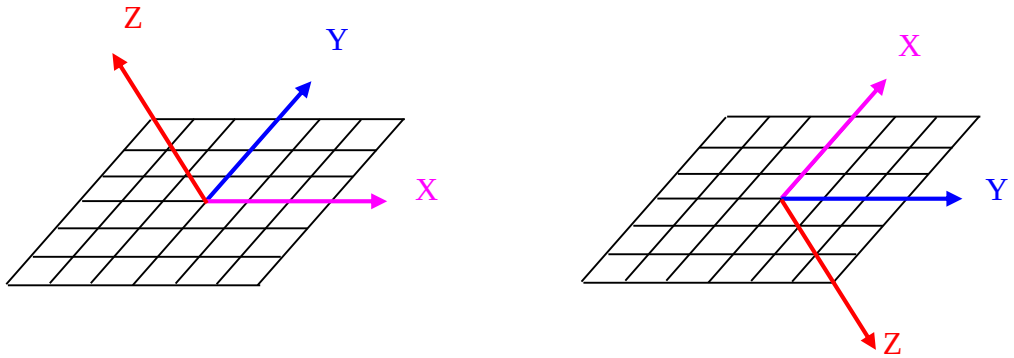
$$\overset{B}{D} = R_3(-90°)R_1(180°)\overset{C}{D} \tag{2.17}$$

$$R_1(180°) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{2.18}$$

$$R_3(-90°) = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.19}$$

Denklemdeki:

- $\overset{B}{D}$ değeri, ilgili kamera koordinat sistemindeki resim noktasının, IMU koordinat sistemindeki yön vektörünü ifade etmektedir,
- R_1 ve R_3 dönüklük matrisleridir.



Kamera koordinat sistemi (C_C) IMU koordinat sistemi (C_B)

Şekil 2.4 : Kamera koordinat sistemi ve IMU koordinat sistemi gösterimi.

2.5.4 IMU Koordinat Sisteminden Yerel Coğrafi Koord. Sistemine Dönüşüm

INS/IMU sisteminden elde edilen dönüklük açıları (roll, pitch, yaw) Yerel Coğrafi Koordinat Sistemindedir. Bu nedenle IMU koordinat sistemine döndürülen yön vektörü, Yerel Coğrafi Koordinat Sistemine dönüştürülür. Bunun için (5) denklemi kullanılır.

$$\overset{L}{D} = \overset{L}{R}_B \overset{B}{D} \tag{2.20}$$

$$R_B^L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\Phi & \sin\Phi \\ 0 & -\sin\Phi & \cos\Phi \end{bmatrix} \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} \cos\Psi & \sin\Psi & 0 \\ -\sin\Psi & \cos\Psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.21)$$

Bu denklemlerde:

- Aşağı eksende saat yönünde dönmesi ile oluşan heading Ψ açısı,
- Bir kez doğu ekseninde saat yönünde dönmesi ile oluşan pitch θ açısı,
- İki kez saat yönünde kuzey ekseninde dönmesi ile oluşan roll Φ açısı,

ile ifade edilmektedir.

2.5.5 Kesişim noktasının hesaplanması

Yukarıda yer alan denklemlerin uygulanması ile diferansiyel ortorektifikasyonun ilk adımı da tamamlanmış olmaktadır. Bu aşamada ise Yerel Koordinat Sisteminde belirlenen yön vektörü referans elipsoidi (WGS 84) üzerine izdüşürülür ve bu yön vektörünün referans elipsoidi ile kesiştiği noktanın kartezyen koordinatları hesaplanır. Bu hesaplama işlemini yapabilmek amacıyla, öncelikle Yerel Coğrafi Koordinat Sisteminde olan yön vektörü, ECEF (Earth Centered Earth Fixed - Yer Merkezli Yer Sabit) koordinat sistemine dönüştürülmelidir. Böylelikle ECEF Koordinat Sistemindeki yön vektörünü içeren kesişim denklemi ve kamera konumu oluşturulabilir.

2.5.5.1 Yerel Coğrafi Koord. Sisteminden ECEF Koord. Sistemine Dönüşüm

Bu dönüşüm işlemi için yazılabilecek denklem şu şekildedir:

$$\overset{E}{D} = \overset{E}{R}_L \overset{L}{D} \quad (2.22)$$

Her iki koordinat sistemi arasındaki ilişki aşağıdaki denklemde olduğu gibi iki ardışık dönüklükle ifade edilebilir:

$$R_L = R_3(-\lambda_A) R_2(\varphi_A - 90) R_3(-90) \quad (2.23)$$

Bu denklemde yer alan:

- φ_A ve λ_A Yerel Coğrafi Koordinat Sisteminin merkezinin enlem ve boylam değerleridir.
- R_3 ve R_2 ise; Z ve Y eksenlerindeki saat yönünde dönüklüklerdir.

Denklem açılacak olursa dönüklük matrisi denklemi aşağıdaki gibi olur.

$$R_L = \begin{bmatrix} -\sin\lambda_A & -\sin\varphi_A\cos\lambda_A & \cos\varphi_A\cos\lambda_A \\ \cos\varphi_A & -\sin\varphi_A\sin\lambda_A & \cos\varphi_A\sin\lambda_A \\ 0 & \cos\varphi_A & \sin\varphi_A \end{bmatrix} \tag{2.24}$$

(2.23) denkleminde değerler yerine konulursa ilgili görüntü noktasının ECEF koordinat sistemindeki yön vektörü $\overset{E}{D}$ elde edilmiş olur. Böylelikle yön vektörü, ilgili algılayıcı olan CCD dizininden çıkıp kamera odak noktasından geçmekte ve referans elipsoidi olan WGS84 elipsoidini kesmektedir. Algoritmanın ikinci adımını da tamamlamak için bu kesişim noktasının da hesaplanması gerekmektedir. ECEF koordinat sisteminde kameranın konumu bilinmekte ve şayet referans elipsoidi üzerindeki kesişim noktası P_0'ın koordinatları $[X_0,Y_0,Z_0]$ ise, aşağıdaki eşitlik yazılabilir:

$$\begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} = \begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{bmatrix} + s * \begin{bmatrix} \overset{E_X}{D} \\ \overset{E_Y}{D} \\ \overset{E_Z}{D} \end{bmatrix} \tag{2.25}$$

Bu denklemde:

- X_0,Y_0,Z_0 ECEF koordinat sisteminde kesişim noktasının koordinatları,
- X_{cam},Y_{cam},Z_{cam} ECEF koordinat sisteminde kameranın koordinatları,
- s kamera ile referans elipsoidi üzerindeki kesişim noktası arasındaki mesafe,
- $\overset{E_X}{D}$, $\overset{E_Y}{D}$, $\overset{E_Z}{D}$ yön vektörünün ECEF koordinat sisteminde bileşenleridir.

Bu eşitlik incelendiğinde, X_0,Y_0,Z_0 ve s değerlerinin bilinmeyen olduğu görülmektedir. Fakat toplamda sadece 3 adet denklem kurulabilmektedir. Bununla birlikte referans elipsoidi üzerinde P_0 noktasının olduğu dikkate alındığında, bu nokta için aşağıdaki gibi bir elipsoidal yüzey denklemi yazılabilir.

Bu denklem:

$$\frac{X_0 + Y_0}{a^2} + \frac{Z_0}{b^2} = 1 \tag{2.26}$$

Bu denklemde:

- *a* WGS 84 elipsoidinin büyük yarıeksen uzunluğu,
- *b* WGS 84 elipsoidinin küçük yarıeksen uzunluğu ifade etmektedir.

Bu denklemler ışığında sistemdeki bilinmeyenler aşağıdaki denklem ile çözülebilir.

$$\left[\frac{\overset{E}{D_x^2} + \overset{E}{D_y^2}}{a^2} + \frac{\overset{E}{D_z^2}}{b^2}\right] * s^2 + 2 * \left[\frac{\overset{E}{D_x} * X_{cam} + \overset{E}{D_y} Y_{cam}}{a^2} + \frac{\overset{E}{D_z} Z_{cam}}{b^2}\right] * s + \frac{X_{cam}^2 + Y_{cam}^2}{a^2} + \frac{Z_{cam}^2}{b^2} - 1 = 0 \tag{2.27}$$

Bu denklemin çözülmesi ile iki faklı *s* değeri hesaplanmaktadır. Bu sonuçlardan kısa olanı bizim aradığımız cevaptır, çünkü büyük olanı elipsoid yüzeyinin diğer tarafından olan mesafeyi verecektir.

s değerini hesaplamak için kullanılacak denklemler şu şekilde verilebilir:

$$s_{1,2} = \frac{-\beta \pm \sqrt{\beta^2 - 4\alpha\gamma}}{2\alpha} \tag{2.28}$$

(2.27) denklemi içerisinde kullanılacak olan s denklemi aşağıdaki gibidir:

$$s = \frac{-\beta - \sqrt{\beta^2 - 4\alpha\gamma}}{2\alpha} \tag{2.28}$$

Bu denklemde yer alan değerler aşağıdaki denklemlerle ifade edilmektedir:

$$\alpha = \frac{\overset{E}{D_x^2} + \overset{E}{D_y^2}}{a^2} + \frac{\overset{E}{D_z^2}}{b^2} \tag{2.29}$$

$$\beta = 2 * \left[\frac{\overset{E}{D_x} * X_{cam} + \overset{E}{D_y} Y_{cam}}{a^2} + \frac{\overset{E}{D_z} Z_{cam}}{b^2} \right] \tag{2.30}$$

$$\gamma = \frac{X_{cam}^2 + Y_{cam}^2}{a^2} + \frac{Z_{cam}^2}{b^2} - 1 \tag{2.31}$$

Hesaplanan bu *s* mesafesinin denklem (2.27) içerisinde yerine konulması ile kesişim noktasının kartezyen koordinatlarına ulaşılabilir. Burada algoritmanın ikinci adımı bitmiş gibi gözükse de maalesef bir başka hata kaynağı karşımıza çıkmaktadır. Işık ışını atmosfer boyunca çeşitli katmanlardan geçmekte ve bu da ışının yansımasına ve yolunun düzgün bir hat olmamasına sebep olmaktadır. Bu nedenle kesişim noktasının koordinatları bu hatalardan giderilerek düzeltilmelidir. Bu düzeltme yön vektörünün zenit açısına getirilmelidir. Yön vektörünün zenit açısının hesaplanabilmesi için, kartezyen koordinat sistemi ECEF'den tekrar Yerel Coğrafi Koordinat Sistemine dönüştürülmesi gerekmektedir. İlgili noktanın jeodezik koordinatlarıyla bu dönüşüm için gerekli olan dönüklük matrisi hesaplanabilir. Bu nedenle kartezyen koordinatların jeodezik koordinatlara dönüşüm ihtiyacı doğmaktadır. Kartezyen koordinatlar, elipsoidal koordinatlara aşağıdaki denklemler yardımıyla iteratif bir yöntemle dönüştürülebilir.

$$\phi_i = \tan^{-1}\left(\frac{Z}{\sqrt{X^2 + Y^2}} \left(1 - e^2 \frac{N_i}{N_i + h_i} \right)^{-1} \right) \tag{2.32}$$

$$\lambda = \tan^{-1}\left(\frac{Y}{X} \right) \tag{2.33}$$

$$h_i = \frac{\sqrt{X^2 + Y^2}}{\cos \phi_i} - N_i \tag{2.34}$$

Bu iteratif yöntemde öncelikle *h* değerini sıfır alıp, ϕ yi hesaplayıp, *h* değerini düzeltmek iyi bir yaklaşım olmaktadır. İterasyon *h* değerleri arasındaki fark belirlenen bir eşik değerinden daha küçük oluncaya kadar devam edecektir.

2.5.5.2 Atmosferik Refraksiyon Etkisi

Snell yasasına göre ışık ışınları çeşitli ortamlardan geçerken ortamın refraksiyon indisine göre kırılmaya uğrarlar ve düz bir yol izleyemezler. Bir ortama giriş ve çıkış açıları için şu eşitlik verilebilir:

$$n_1 \sin\theta_1 = n_2 \sin\theta_2 \tag{2.35}$$

Bu denklemde:

- n_1 ilk ortamın refraksiyon indeksi,
- n_2 ikinci ortamın refraksiyon indeksi,
- θ_1 ışığın iki ortam arasındaki sınıra giriş açısı,
- θ_2 ışığın iki ortam arasındaki sınırdan çıkış açısıdır.

Işığın ortama giriş açısı, zenit açısı ile aynı olacaktır.

$$N = (n-1)10^6 \tag{2.36}$$

Buradaki N refraktivitedir ve görünür ışık için şu şekilde hesaplanabilir:

$$\frac{\partial N}{\partial h} = -78\frac{p}{T^2}\left(0{,}034 + \frac{\partial T}{\partial h}\right) - \frac{11}{T}\frac{\partial e}{\partial h} \tag{2.37}$$

Bu denklemde:

- h ortalama deniz seviyesi yüksekliği
- p atmosferik basınç (hPa)
- T atmosfer sıcaklığı (Kelvin)
- e su buhar basıncı (Pa)

(2.37) denklemindeki parametreler sabit olmadığından, her atmosfer katmanı için yeniden hesaplanmalıdır. Refraktivitenin hesaplanması için atmosfer 1km kalınlığında katmanlara ayrılmıştır. Her bir katman için refraktivite sabit olarak belirlenebilmektedir.

Atmosferik basınç p, aşağıda yer alan barometrik formül yardımı ile belirlenebilmektedir:

$$p = p_0 e^{\frac{-Mgh}{RT}} \tag{2.37}$$

Burada:

- h yükseklik (m),
- p_0 yer seviyesindeki basınç,
- M 1mol havanın kütlesi (0,029kg mol^{-1}),
- R gaz sabiti (8,314 J K^{-1} mol^{-1}),
- g_0 yerin gravitesi (9,81 m s^{-2}) dir.

2.5.5.3 ECEF Koord. Sisteminden Yerel Coğrafi Koord. Sistemine Dönüşüm

Yukarıda belirtildiği gibi atmosferik düzeltmelerin uygulanabilmesi için hesaplanan yön vektörünün zenit açısına ihtiyaç duyulmaktadır. Bu nedenle de kartezyen koordinat sisteminde (ECEF) hesaplanan yön vektörünün zenit açısının ve azimut açısının hesaplanabilmesi için, Yerel Coğrafi Koordinat Sistemine dönüştürülmesi gerekmektedir.

Bu amaca yönelik olarak öncelikle üçüncü eksen Z'ye göre sistem λ derece kadar döndürülmeli, ardından ikinci eksen Y eksenine göre de $90\text{-}\phi$ derece kadar döndürülmelidir. Lokal sistemden sol el sistemine dönmek için de birinci eksen X'e göre -1 ile çarpılmalıdır. Sonuçta elde edilen dönüklük matrisi şu şekilde olmaktadır:

$$\overset{L}{R}_E = Q_1 R_2(90-\phi) R_3(\lambda) \tag{2.38}$$

Bu denklemde:

- ϕ yeryüzündeki kesişim noktasının enlemi,
- λ yeryüzündeki kesişim noktasının boylamıdır.

(2.38) denklemindeki matris çarpımı yapılırsa denklem şu hale gelir:

$$\overset{L}{R}_E = \begin{bmatrix} -\sin\phi\cos\lambda & -\sin\phi\sin\lambda & \cos\phi \\ -\sin\lambda & \cos\lambda & 0 \\ \cos\phi\cos\lambda & \cos\phi\sin\lambda & \sin\phi \end{bmatrix} \tag{2.39}$$

Bu matris içerisinde $sin(90\text{-}\phi)$ yerine $cos(\phi)$ ve $cos(90\text{-}\phi)$ değeri de $sin(\phi)$ ile yer değiştirilirse:

$\overset{L}{D} = \overset{L}{R}_E \overset{E}{D}$ şeklinde verilen (2.22) denklemi matris formunda şu şekilde yazılabilir:

$$\begin{bmatrix} \overset{L}{D}_x \\ \overset{L}{D}_y \\ \overset{L}{D}_z \end{bmatrix} = \begin{bmatrix} -\sin\phi\cos\lambda & -\sin\phi\sin\lambda & \cos\phi \\ -\sin\lambda & \cos\lambda & 0 \\ \cos\phi\cos\lambda & \cos\phi\sin\lambda & \sin\phi \end{bmatrix} \begin{bmatrix} \overset{E}{D}_x \\ \overset{E}{D}_y \\ \overset{E}{D}_z \end{bmatrix} \tag{2.40}$$

Böylelikle yerel koordinat sistemindeki yön vektörüne ait α azimut açısı ve z zenit açıları aşağıdaki denklemlerle hesaplanabilir:

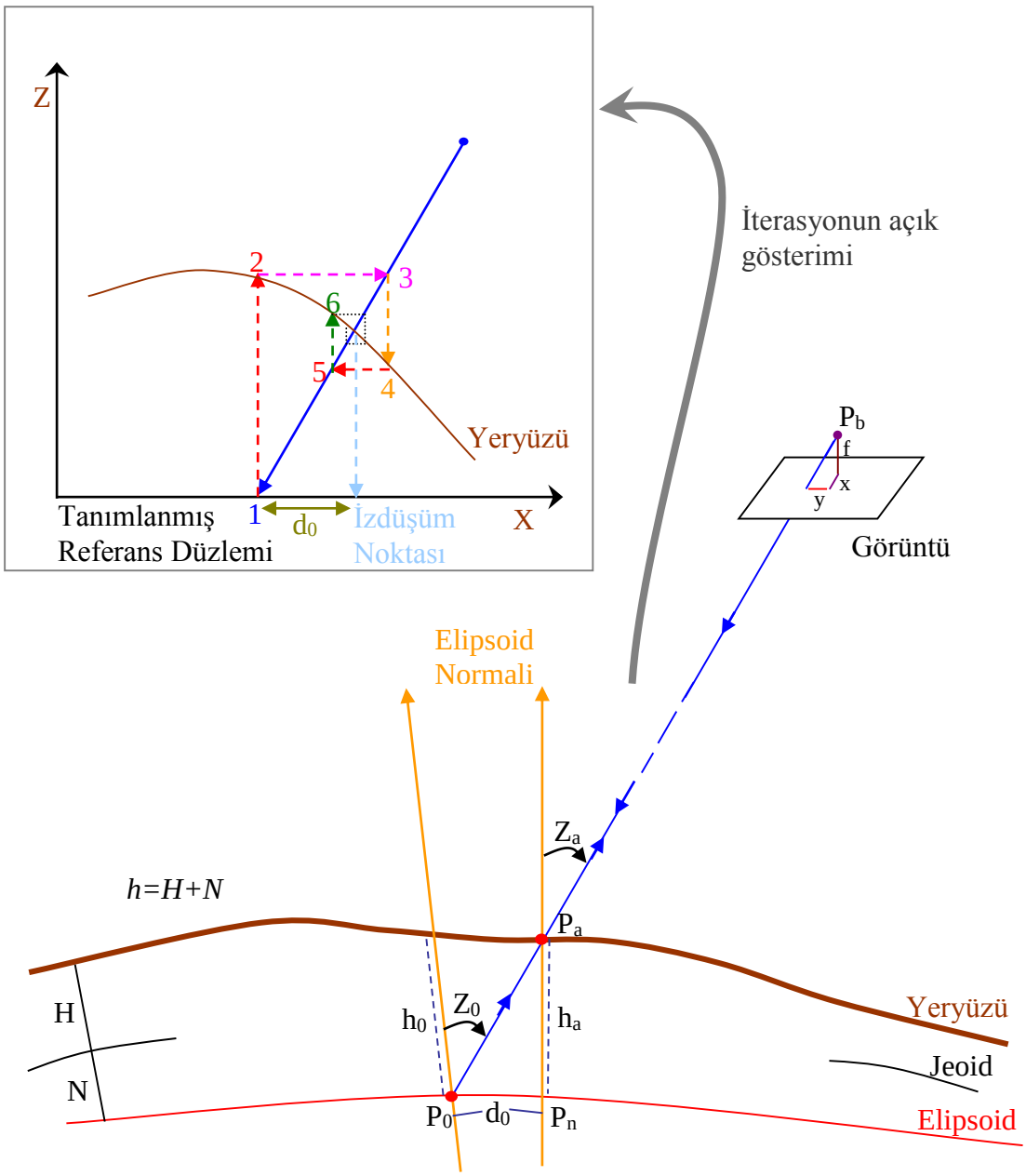
$$\alpha = \tan^{-1}\left(\frac{\overset{L}{D}_y}{\overset{L}{D}_x}\right) \tag{2.41}$$

$$z = \tan^{-1}\left(\frac{\sqrt{\overset{L}{D}_x^{\,2} + \overset{L}{D}_y^{\,2}}}{\overset{L}{D}_z}\right) \tag{2.42}$$

2.5.6 Rölyef Kayma Düzeltmesi

Yön vektörünün kesişim noktası artık tam olarak WGS 84 referans elipsoidi üzerinde bulunmaktadır. Başka bir deyişle bu noktadaki elipsoid yüksekliği sıfırdır. Görüntü üzerinde bulunan piksellerin konumu yükseklik farklarından kaynaklanan rölyef kaymasından dolayı değişir. Bu hatanın giderilebilmesi için, yeryüzündeki noktaların gerçek yükseklik değerlerinin bilinmesi gerekir. Bu nedenle tekil görüntülerde rölyef kaymasının düzeltilebilmesi için sayısal yükseklik modeli (SYM) kullanılır.

Öncelikle mevcut kesişim noktasının yükseklik değeri SYM üzerinden okunur. Ardından bir rölyef kayma düzeltmesi referans elipsoidi üzerinde jeodezik konumu için yapılır, düzeltilmiş konumun yükseklik değeri yeniden SYM üzerinden okunur ve yeniden bir düzeltme yapılarak her iki yükseklik değeri arasındaki fark belli bir eşik değerine gelene kadar bu işlem iteratif olarak tekrarlanır (Şekil 2.5).



Şekil 2.5 : Kesişim noktasına rölyef kayma düzeltmesinin iteratif olarak getirilmesi.

WGS84 referans elipsoidi ile yön vektörünün kesişim noktası Şekil 2.5'de P_0 ile ifade edilir. Bu noktanın yüksekliği jeodezik koordinatlar yardımıyla SYM üzerinden okunur. Azimut, zenit açısı ve elipsoidal parametrelerin hesaplamada kullanılmasıyla kesişim noktasının jeodezik koordinatları düzeltilir. Düzeltme formülü (2.43) denklemi ile verilmiştir. İlk iterasyonun ardından mevcut nokta P_1 olarak ifade edilir ve jeodezik kordinatları bir öncekinden farklıdır. Haliyle bu yeni noktadaki elipsoid normali de farklı olacaktır. Yön vektörü yeniden ECEF Kartezyen Koordinat Sisteminden, Yerel Coğrafi Koordinat Sistemine döndürülerek düzeltilmiş zenit ve

azimut açıları hesaplanır ve SYM üzerinden yeni bir yükseklik değeri okunur. Bu iterasyon, [h_1-h_0] farkı belirlenen eşik değerinin altında kalana kadar devam eder.

$$\Delta h_n = h_n(\lambda_{n-1}, \phi_{n-1}) - h_{n-1}(\lambda_{n-1}, \phi_{n-1}) \tag{2.43}$$

$$d_n = \Delta h_n \tan(z) \tag{2.44}$$

$$\lambda_n = \lambda_{n-1} + d_n \sin(\alpha) * \left(\frac{V}{c}\right)_{n-1} \left(\frac{1}{\cos(\phi_{n-1})}\right) \tag{2.45}$$

$$\phi_n = \phi_{n-1} + d_n \cos(\alpha_{n-1}) * \left(\frac{V^3}{c}\right)_{n-1} \tag{2.46}$$

Bu denklemlerde;

$$c = \frac{a^2}{b} \tag{2.47}$$

$$V_n = \sqrt{1 - \frac{a^2 - b^2}{b^2} \cos^2(\phi_o)_n} \tag{2.48}$$

ile ifade edilir. Eşik değer $|\Delta h_n| < \varepsilon$ olduğunda;

$$\lambda_a = \lambda_n, \phi_a = \phi_n, h_a = h_n \tag{2.49}$$

elde edilir. Böylelikle n sayıda iterasyondan sonra noktanın koordinatları λ_n, ϕ_n, h_n olarak hesaplanmış olur.

2.5.7 Coğrafi Koordinat Sisteminden UTM Koordinat Sistemine Dönüşüm

Her bir pikselin konumu için jeodezik koordinatların hesaplanmasından sonra, her bir koordinatın Universal Transverse Mercator (UTM) koordinatlarına dönüşümü ihtiyacı ortaya çıkar. UTM projeksiyonuna dönüşüm için tanımlanması gereken bazı sabitler vardır:

- k_0 merkezi meridyen boyunca ölçek faktörü,
- ϕ_0 Grid merkezinin enlemi,
- λ_0 Grid merkezinin boylamı,

- E_0, N_0 false easting ve false northing değerleri

Grid merkezinin boylamı genelde merkezi meridyenin boylamı olarak belirlenir.

UTM dönüşümleri genelde limitli sayıdaki terimleri içeren seriye açılımla yapılır. Burada asıl anahtar parametre, ekvatordan belirli bir ϕ enlemine kadar olan meridyen mesafesi M'dir. Bu parametrenin hesaplanması aşağıda yer alan denklem gibi bir dizi seriye açılımla yapılabilir:

$$M = a\left[\begin{array}{l}\left(1-\frac{e^2}{4}-\frac{3e^4}{64}-\frac{5e^6}{256}\right)\phi-\left(\frac{3e^2}{8}+\frac{3e^4}{32}+\frac{45e^6}{1024}\right)\sin(2\phi)+\\ \left(\frac{15e^4}{256}+\frac{45e^6}{1024}\sin(4\phi)\right)-\frac{35e^6}{3072}\sin(6\phi)\end{array}\right] \tag{2.50}$$

Bu denklemde;

- M ekvatordan belirli bir ϕ enlemine kadar olan meridyen mesafesi,
- a büyük yarı eksen,
- e eksentrisite,
- ϕ enlemdir (radyan).

Bu aşamadan sonra noktanın enlem ve boylam değerleri, X ve Y UTM koordinatlarına dönüştürülebilir.

$$T = \tan^2\phi \tag{2.51}$$

$$C = e'^2\cos^2(\phi) \tag{2.52}$$

$$A = (\lambda-\lambda_0)\cos\phi \tag{2.53}$$

Bu denklemlerde;

- λ ve λ_0 radyan cinsindendir,
- e' ikinci eksentrisite,
- λ_0 grid merkezinin boylamı (merkezi meridyen)

$$X = k_0 N\left[A + (1 - T + C)\frac{A^3}{6} + (5 - 18T + T^2 + 72C - 58e'^2)\frac{A^5}{120} \right] + E_0 \qquad (2.54)$$

$$Y = k_0 \left[M - M_0 + N\tan\phi \left[\frac{A^2}{2} + (5 - T + 9C + 4C^2)\frac{A^4}{24} + (61 - 58T + T^2 + 600C - 330e'^2)\frac{A^6}{720} \right] \right] + N_0 \qquad (2.55)$$

Bu denklemlerde;

- k_0 merkezi meridyenin ölçek faktörü,
- e' ikinci eksentrisite,
- E_0, N_0 false easting ve false northing değerleri
- M değeri ϕ enlemindeki meridyen mesafesi,
- M_0 değeri ϕ_0 enlemindeki meridyen mesafesi,
- N ϕ enlemindeki elipsoid normalinin uzunluğudur.

2.5.8 Yeniden Örnekleme

Yeryüzündeki noktanın jeodezik koordinatlardan UTM koordinatlara dönüştürülmesinden sonra sıra, ortofoto görüntünün oluşturulmasına gelmiştir. Hesaplanan UTM koordinatları düzensiz ve boşluklu bir yapıya sahiptir. Bu nedenle parlaklık değerlerini de dikkate alarak yeniden düzenli şekil ve boşluğa sahip bir UTM gridi ile yeniden tanımlanmaları gerekir.

Oluşturulan bu grid içerisindeki koordinatlara karşılık gelen piksellerin renk değerlerinden yeniden örnekleme yapılarak sayısal görüntü oluşturulur. Bu amaca hizmet eden birçok teknik vardır. Bunlardan en yaygın olarak kullanılanları, en yakın komşuluk enterpolasyonu, bilineer enterpolasyon ve bikübik enterpolasyondur. Örneğin bilineer enterpolasyonda parlaklık değeri bilinmeyen pikselin çevresindeki 2x2 lik bilinen pikseller hesaplama için kullanılmaktadır. Bu dört pikselin ağırlıklı ortalaması ile aranan sonuç pikselin gri değerine ulaşılır.

Yeniden örnekleme için kullanılmak üzere herhangi bir enterpolasyon yöntemini seçmek için kullanılabilecek teorik bir ölçüt bulunmadığından, ortorektifikasyon

işlemi için geliştirilen yazılımların çeşitli enterpolasyon tekniklerini içerecek yapıda olması gerekir.

3. PARALEL İŞLEM VE NVIDIA CUDA TEKNOLOJİSİ

Paralel hesaplama ve programlama, yakın bir geçmişe kadar bilgisayar bilimi altında uzmanlık gerektiren çok özel bir alan olarak karşımıza çıkmaktaydı. Bu algı son yıllarda derinlemesine değişerek, bilgisayar dünyası içerisindeki her programcının paralel programlama üzerine eğitimli olması, bilgisayar biliminde tam olarak etkili olabilmesi için bir gereklilik haline dönüşmüştür.

3.1 Paralel İşlemin Tarihçesi ve Merkezi İşlem Birimi

Son yıllarda bilgisayar endüstrisi paralel programlamaya hızlı bir geçiş yapmıştır. 2010 yılından sonra üretilen hemen hemen tüm bilgisayarlar çok çekirdekli merkezi işlem birimine (Central Processing Unit-CPU) sahip olarak üretilmeye başlanmıştır. Geçmiş tarihlerde süper bilgisayarlar olarak ifade edilen bilgisayarların üzerinde bulunan işlemci kapasiteleri ve çekirdek sayıları, günümüzde daha fazla sayıda artarak ve güçlenerek, bilgisayarlardan tablet bilgisayarlara, akıllı cep telefonlarına ve hatta taşınabilir müzik çalarlara kadar birçok elektronik aygıtta kullanılır hale gelmiştir. Böylelikle paralel işlem yapma yeteneği süper bilgisayarların ve iş istasyonlarının tekelinden çıkarak, elimizin altındaki bilgisayarlar, akıllı cep telefonları, taşınabilir müzik çalarlar gibi birçok elektronik aygıtın sahip olduğu bir yetenek halini almıştır. Böylelikle yazılım geliştirenler de kullanıcıların her geçen gün artan değişik taleplerine cevap verebilmek için paralel hesaplama platformları ve teknolojileri üzerine kendilerini geliştirmektedirler. Günümüzde artık komut arayüzleri demode, çoklu işlemli grafik arayüzler moda, sadece telefon görüşmeleri yapılan cep telefonları demode, aynı anda müzik dinlenebilen, internete girilebilen, GPS servislerini kullanan akıllı telefonlar moda olmuştur. Bugünkü noktaya da aslında elektronik ve paralel işlem teknolojisinin gelişmesi ile gelinmiştir.

Otuz yıldır, hesaplama araçlarının aygıtlarının geliştirilmesi için kullanılan en önemli yöntem; işlemcinin saat hızının artırılması olmuştur. 1980'lerde ilk kişisel bilgisayarlar kullanılmaya başlandığında, CPU'ların saat hızları yaklaşık 1MHz'di. Yaklaşık otuz yıl sonra, birçok masaüstü bilgisayarın işlemci hızı 1GHz ile 4Ghz

arasında bir değere ulaştı ki bu da ilk kişisel bilgisayarlara göre 1000 kattan daha fazla bir saat hızı demektir. CPU saat hızının arttırılması, bilgisayarların performansını artırmak için kullanılan tek yöntem olmamasına rağmen, performans geliştirmesi için her zaman kullanılacak güvenilir bir kaynak olmuştur.

Son yıllarda, artan işlem gücünün bu geleneksel kaynağına alternatifler araması konusunda üreticiler zorlanmıştır. Entegre devrelerin üretimindeki çeşitli temel fabrikasyon kısıtlamalar sebebiyle, mevcut mimariler üzerinde işlemci saat hızlarını artırarak ek güç artırımı artık pek mümkün olmamaktadır. Güç ve ısınmadan kaynaklanan kısıtlamalar sebebiyle, işlemcilerin temelini oluşturan transistörlerin boyutlarında fiziki sınırlara gelinmiştir. Bu nedenle araştırmacılar ve üreticiler başka alternatifler aramaya başlamışlardır.

Bilgisayar tüketicisi dünyası dışında, süper bilgisayarlar yıllardır benzer yollarla güçlü performanslarını korumuşlardır. Süper bilgisayarlarda kullanılan işlemcilerin performansı kişisel bilgisayarların CPU'larındaki gelişme gibi astronomik olarak artmıştır. Bununla birlikte, tek bir işlemcinin performansındaki dramatik gelişmeye ek olarak, süper bilgisayar üreticileri performanstaki güçlü atılımı işlemci sayılarını artırarak yapmışlardır. Bu sebeple en hızlı süper bilgisayarların on binlerce veya yüz binlerce birbiri ardına çalışan işlemci çekirdeğine sahip olması nadir bir durum değildir.

Üreticiler, kişisel bilgisayarlar için ek bir işlem gücü arayışı içerisindeyken, süper bilgisayarlarda olduğu gibi tek bir işlemcinin saat hızını artırmak için çalışmaktansa, birden fazla işlemci çekirdeğini kişisel bilgisayarlar içerisine koymanın daha etkili bir çözüm olacağını anlayarak, bu yolla kişisel bilgisayarların performanslarını geliştirmeyi tercih ettiler.

2005 yılında, pazardaki artan rekabet ve birkaç alternatifin oluşması nedeniyle, önde gelen CPU üreticileri bir yerine iki çekirdekli işlemcileri satışa sunmaya başladı. İzleyen yıllarda, üç, dört, altı, sekiz, oniki, onaltı çekirdekli işlemcilerle bu gelişme devam etti. Bu durum özellikle tüketici açısından bilgisayar pazarında büyük bir değişim ve ilerleme yaratarak, çok çekirdekli devrim olarak nitelendirilmektedir.

Günümüzde, tek çekirdekli işlemciye sahip bir masaüstü bilgisayar bulmak gerçekten çok zor bir hale gelmiş ve çok düşük güçteki işlemciler bile en az iki çekirdekli üretilir olmuşlardır.

3.2 GPGPU ve Akış İşleme

CPU'ların gelişmesi esnasında, grafik işlemede de bir devrim yaşanmıştır. 1980'lerin sonu 1990'ların başında, Microsoft Windows gibi grafik ağırlıklı işletim sistemlerinin popüler hale gelmesi, yeni tip bir işlemci için bir pazar yaratmıştır. 1990'ların başında insanlar kişisel bilgisayarları için iki boyutlu (2B) ekran hızlandırıcıları satın almaya başlamışlardır. Bu ekran hızlandırıcıları donanım destekli bitmap işlemleri ile grafik işletim sistemlerinin görüntülenmesi ve kullanılabilmesine yardımcı olmuştur.

Aynı zamanlarda, profesyonel bilgisayar dünyasında, Silicon Graphics adında bir şirket, devlet ve savunma sanayi uygulamaları, bilimsel ve teknik görselleştirmeler, çarpıcı sinema efektleri oluşturmak gibi konularda üç boyutlu (3B) grafiklerin kullanımını artırmak ve popülerleştirmek için 1980'li yıllarını harcamıştır. 1992 yılında Silicongraphics kendi donanımı üzerinde kullandığı programlama arayüzü olan OpenGL kütüphanesini herkesin kullanımına açmış ve özellikle OpenGL'in 3B uygulamaların yazılması ve geliştirilmesinde kullanılacak platform bağımsız, standart bir altyapı olmasını sağlamaya çalışmıştır.

1990'ların ortalarında, kullanıcıların 3B'lu grafikleri içeren taleplerindeki artış paralelinde iki önemli gelişme olmuştur. Bunlardan birisi; Doom, DukeNukem 3D, Quake gibi kişi odaklı oyunların (First Person Games-Birinci Kişi Oyunları) geliştirilmesinin oyun dünyasında daha gerçekçi 3B ortamlar yaratılmasında yeni yöntem arayışlarına önayak olmasıdır. Bir diğeri ise; Nvidia, Ati, 3dfx Interactive gibi şirketlerin grafik hızlandırıcıların geliştirilmesine başlanmış olmasıdır. Bu gelişmeler gelecek yıllarda 3B'lı grafiklerin bir teknoloji olarak ön plana çıkışını güçlendirmiştir.

Nvidia'nın Geforce 256 serisi ekran kartları kullanıcıların grafik donanım yeteneklerini ileri derecede artırdı. Bu seri ile ilk defa 3B grafiklerdeki dönüşüm ve ışıklandırma hesapları doğrudan grafik işlemci üzerinde yapılabilmiştir. Böylelikle daha görsel ve ilginç uygulamaların gerçekleştirilebilmesi potansiyeli artmıştır. OpenGL grafik iş akışının temel bir parçası olan dönüşüm ve ışıklandırmanın doğrudan grafik işlemciler üzerinde hesaplanması ile Geforce 256, grafik işakışının doğrudan ekran kartları üzerine uygulanması yönünde ilerleyişin doğal bir başlangıcı olmuştur.

Paralel hesaplama açısından, Nvidia'nın GeForce3 serisini 2001 yılında çıkarması tartışmasız GPU teknolojisindeki en önemli atılım olmuştur. GeForce 3 serisi, Microsoft'un DirectX 8.0 standardını uygulayan hesaplama endüstrisindeki ilk çiptir. Bu standart programlanabilir vertex ve programlanabilir piksel gölgelendirme adımlarının her ikisini birden içeren uyumlu donanımı gerektirmektedir. Bu sayede ilk defa geliştiriciler, GPU üzerinde gerçekleştirdikleri kesin hesaplamalar üzerinde bazı kontrollere sahip olmuşlardır.

GPU'lar basit OpenGL ve DirectX desteği olan grafik donanımları olmalarından çok programlanabilir olmaları açısından birçok araştırmacının ilgisini çekmiştir. GPU hesaplamanın ilk zamanlarındaki genel yaklaşım olağanüstü karmaşıktı. Çünkü OpenGL ve DirectX gibi API'ler (uygulama programlama arayüzü - application programming interface) halen GPU üzerinde hesaplama yapabilmek için kullanılabilecek, çeşitli kısıtlamaları olan ve aslında grafik programlama için tasarlanan araçlardır. Bu nedenle de araştırmacılar problemlerini çözmek için genel amaçlı hesaplamaları grafik API'nin anlayacağı şekilde geleneksel grafik işleme problemi olan renderlama işlemine benzetmenin yöntemlerini aramışlardır.

Aslında, 2000'lerin başındaki GPU'lar, piksel gölgelendirici (piksel shader) olarak bilinen programlanabilir aritmetik birimleri kullanarak ekran üzerindeki her bir pikselin rengini üretmek için tasarlanmışlardır. Genel olarak bir piksel gölgelendirici pikselin ekran üzerindeki (x, y) konumunu ve ek bazı bilgileri girdi olarak birleştirip hesap yaparak pikselin sonuç rengini üretmektedir. Bahsedilen ek bilgiler; girdi renkleri, doku (texture) koordinatları veya çalıştırıldığında gölgelendirmeye etki edecek diğer özellikler olabilmektedir. Fakat girdi renkleri ve dokularda uygulanan aritmetik tümüyle programcı tarafından kontrol edilebilmektedir. Bu noktada araştırmacılar "girdi renklerinin" aslında herhangi bir "veri" olabileceğini keşfetmişlerdir.

Böylelikle girdiler renkten farklı olarak sayısal bir veri olarak nitelendirildiğinden, programcılar piksel gölgelendiricileri bu veri üzerinde istenilen hesaplamaları yapmak için programlayabilmişlerdir. Elde edilen sonuçlar aynı sonuç piksel renkleri gibi GPU'ya geri dönmüş ve böylelikle bu renkler basitçe programcının girdilerini vererek GPU'ya yaptırdığı hesaplama işleminin sonuçları olmuşlardır. Bu veri araştırmacılar tarafından geri okunabilmiştir. Özünde GPU, renderlama işlemi

olmayan işlemleri sanki renderlama işlemiymiş gibi görerek renderlama işlemi yapıyormuş gibi yapmıştır.

GPU'ların yüksek aritmetik çıktısından elde edilen sonuçlar, GPU hesaplamanın parlak bir gelecek vaat ettiğini göstermiştir. Ancak halen programlama modeli geliştiriciler için çok kısıtlayıcı bir yapıya sahipti. Özellikle girdi verilerinin, sadece renk biçiminde ve doku birimleri üzerinden programlar tarafından alınabilmesi çok sıkı bir kaynak kısıtı olarak ön plana çıkmıştır. Programcıların, sonuçları hafızada nereye nasıl yazacağı konusunda ciddi sınırlamaları olmuştur. Bu sebepten dolayı algoritmaların hafıza üzerindeki rasgele konumları kullanmaya izin verecek şekilde yazılması, kodun GPU üzerinde çalışabilmesini engellemiştir. Daha da ötesinde kayan noktalı veri ile ne kadar doğru hesaplama yapacağını tahmin etmek neredeyse imkânsız olmuş ve şayet kayan noktalı veri kullanabilseydi, çoğu bilimsel hesaplama GPU ile yapılabilecekti. Sonuç olarak program yanlış hesaplar ürettiğinde, hatanın nereden kaynaklandığını ortaya çıkarabilecek hata ayıklama yöntemi GPU için yoktu.

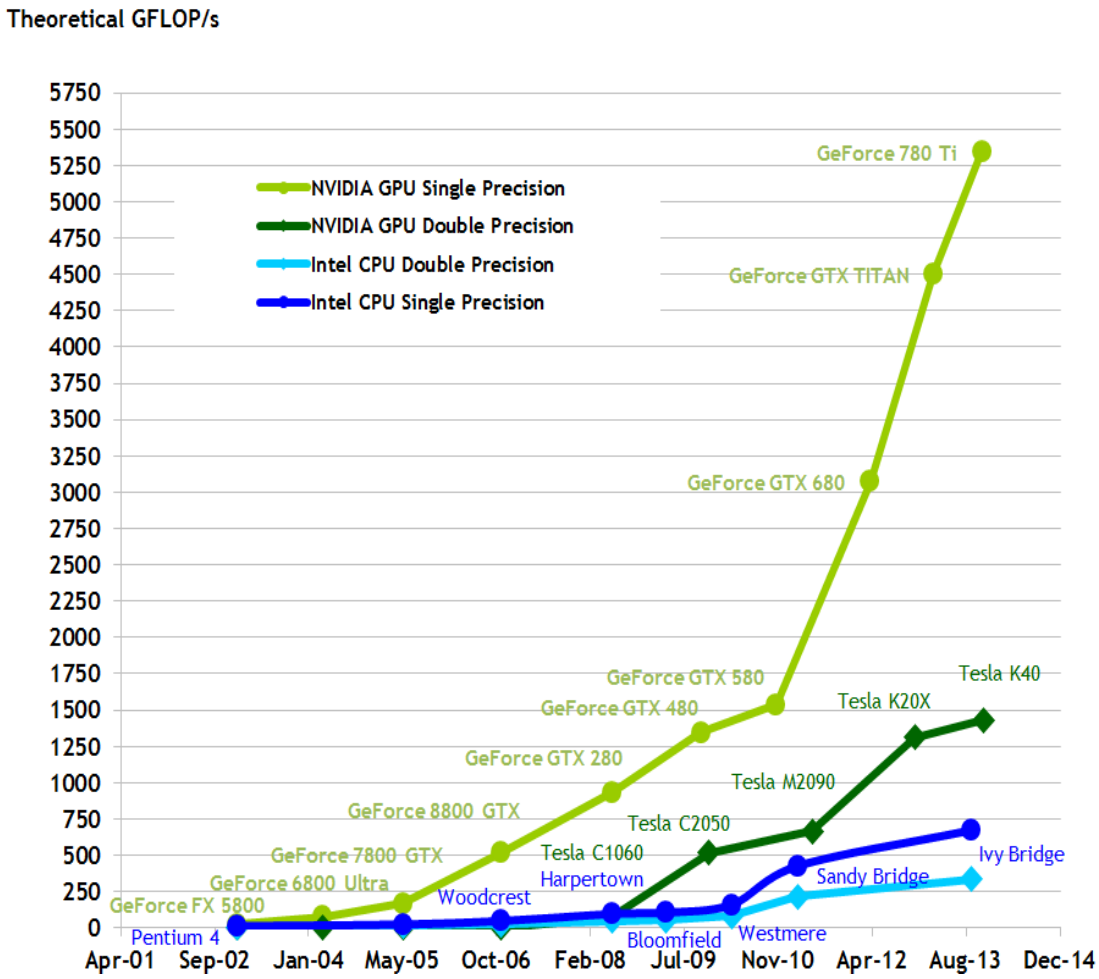
Bütün bu sınırlamaların yanında bir de GPU üzerinde genel amaçlı hesaplama yapmak için OpenGL ya da DirectX öğrenmek gerekiyordu. Bu noktada sadece verinin grafik dokular gibi saklanması değil aynı zamanda gölgelendirme dilleri gibi özel grafik programlama dillerini bilerek, OpenGL veya DirectX fonksiyonları ile hesaplama yapılması da gerekiyordu.

Aslında hesaplama, CPU üzerindeki "merkezi işlemden", CPU ve GPU üzerindeki "birlikte işleme" doğru bir evrim geçirmektedir. 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.

GPGPU için uygulanacak 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

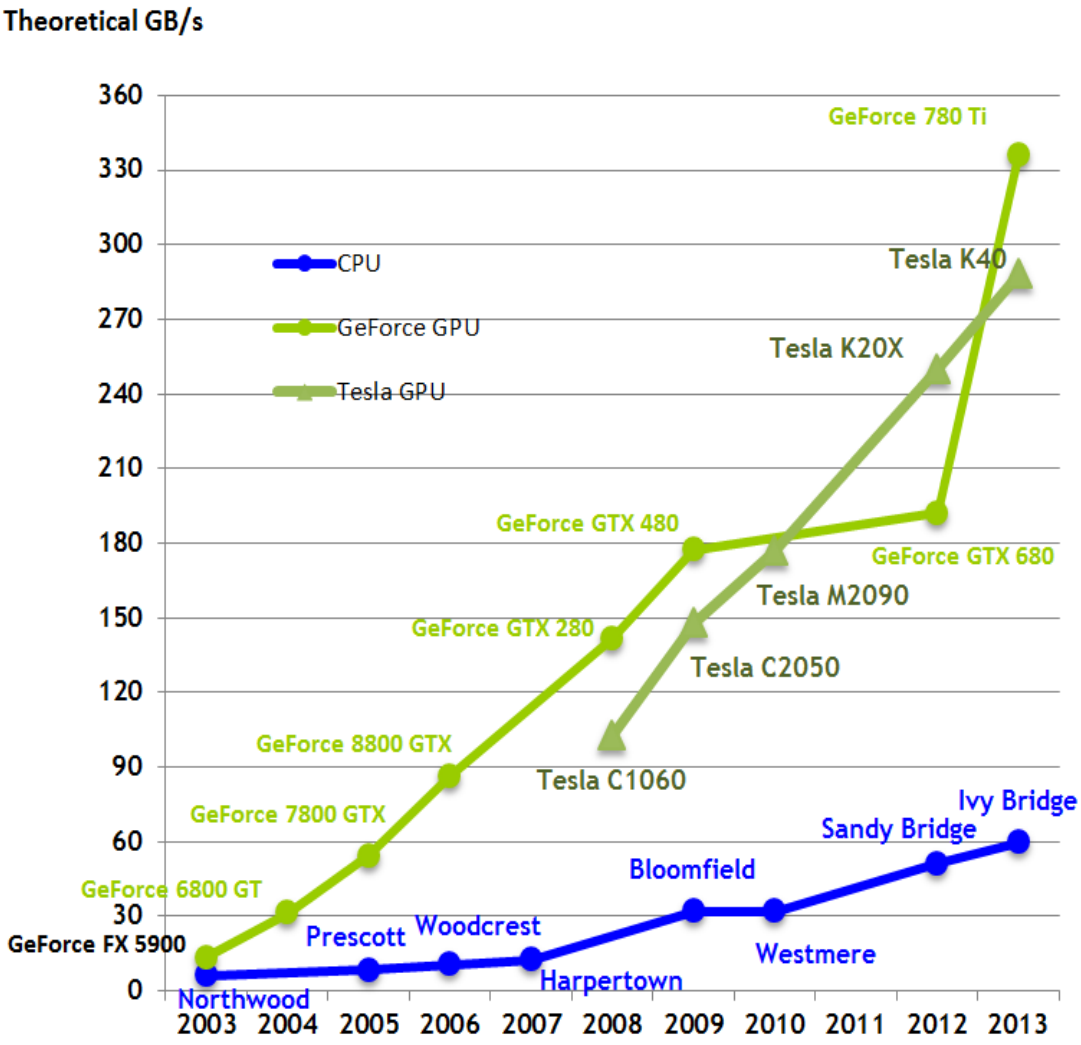
çok sayıda işlemci bulunmaktadır. Örneğin günümüzde kullanılan grafik kartlarından birisi olan Nvidia GTX780 serisi grafik kartı üzerinde 2304 adet akış işlemcisi bulunmaktadır. Bu da yan yana sıralanmış, birlikte işlem yapabilme kapasitesi olan 2304 adet tek çekirdekli işlemcisi olan bilgisayarlar kümesi 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 3.1'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 2014 yılındaki değerlere göre ortalama on kat daha hızlı olduğu görülmektedir.



Şekil 3.1 : Saniyedeki kayan nokta işlem miktarı bakımından GPU ve CPU'ların yıllara göre gelişimi (Nvidia, 2015).

Hafıza band genişliği kavramı, ekran kartının işlemcisi ile hafızası 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 3.2 incelendiğinde, GPU'ların hafıza band genişliklerinin, CPU'lara oranla 4 kat daha fazla bir orana ulaştıkları görülmektedir.



Şekil 3.2 : GPU ve CPU'ların band genişlikleri (Nvidia, 2015).

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 3.3'de gösterilmeye çalışılmıştır.



Şekil 3.3 : CPU ve GPU'ların genel yapısı ve sahip oldukları transistör sayısındaki farklılık (Nvidia, 2015).

Şekil 3.3'e 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.

3.3 CUDA Teknolojisi ve Genel Amaçlı Hesaplama

Yapılacak bir hesaplamada, tekrarlanan hesaplama işlem adımının birçok veri elemanına, yüksek aritmetik doğrulukla uygulanmasının gerektiği hallerde GPU'lar çok etkili sonuçlar vermektedir. 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). Benzer şekilde görüntü ölçekleme, stereo görüş ve görüntüden şekil tanıma gibi renderlanmış görüntülerin ve videoların işlenmesinde görüntü bloklar ve pikseller şeklinde iş parçaları haline getirilebilirler. Aslında görüntü işleme dışında sinyal işlemeden fiziksel simülasyona, finansal hesaplamadan

biyolojik hesaplamalara kadar birçok algoritma paralel programlama ve paralel işlemeyi kullanarak hızlanmışlardır.

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 (Birleşik Hesaplama Aygıt Mimarisi – Compute Unified Device Architecture) denilmiştir.

2006 yılının Kasım ayında NVIDIA, endüstrideki ilk DirectX10 destekli GPU'su GeForce8800 GTX'i duyurdu. Bu ekran kartı aynı zamanda NVIDIA CUDA teknolojisine sahip ilk ekran kartı idi. Fakat duyuruları daha önceden yapılmış olmasına rağmen NVIDIA, yeni teknolojisi CUDA'yı ancak 2007 yılının Şubat ayında kamuoyuna açıklayarak tanıtmıştır. Bu teknoloji geniş bir kullanıcı kitlesinin gereksinimlerini karşılamak için geliştirilmiştir. En önemli ihtiyaç GPU'ların kolayca programlanabilirliğinin sağlanmasıdır. Basitlik ve sadelik, GPU ile paralel programlamanın kolaylığını ve daha fazla disiplin içerisinde kullanılabilirliğini sağlamak için gerekli görülmüştür. CUDA'dan önce, GPU paralel programlama grafik API(Application Programming Interface-Uygulama Programlama Arayüzü)'lerinin gölgelendirme modelleri ile sınırlıydı. Böylelikle sadece vertex ve

parça gölgelendirmesinin doğasına uygun olan problemler GPU paralel programlama yöntemi kullanılarak hesaplanabilmiştir. Ayrıca, dokulandırma ve GPU gereken 3B işlemler için sadece kayan noktalı sayıların kullanılması GPU hesaplama konusunun popülaritesini sınırlandırmıştır. NVIDIA firması GPU paralel programlamayı kolay ve pratik hale getirebilmek için C programlama dilinin minimum eklentileri ile kullanımını önermiştir. Diğer önemli bir konu ise CPU ve GPU kaynaklarının birarada kullanılabilirliğini sağlayan heterojen bir hesaplama modelinin ortaya konulması olmuştur. CUDA, programcıların kod ve veriyi alt parçalara bölmesine olanak tanıyarak, CPU/GPU mimarisi ve ilgili programlama tekniklerine uygun hale getirebilmesini sağlamıştır. Böyle bir ayrım aslında GPU ve CPU'nun her ikisinin de kendi hafızasına sahip olmasından kaynaklanmaktadır. Bu anlamda aynı zamanda mevcut tanımlamaların CPU'dan GPU'ya aktarımı mümkün olabilmektedir (Yılmaz, 2010). Kısaca 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 ile GPU programlama, daha önceki GPGPU çözümlerine göre oldukça esnek ve kolaydır.

CUDA mimarisi, NVIDIA GeForce 8x serisi ve GeForce, Quadro, Tesla gibi daha yeni model tüm grafik kartlarında kullanılabilmektedir. Veri-paralel ve iş parçası-paralel mimari ölçeklenebilir bir yapıya sahiptir. Yeniden fazladan bir çaba sarfetmeden mevcut çözümü GPU üzerinde daha fazla iş parçacığı işleme kapasitesiyle uygulamak ve çalıştırmak mümkündür. Başka bir ifadeyle NVIDIA 8 serisi için yazılan bir kod, başka bir kodlamaya gerek duymadan NVIDIA Quadro için de çalışabilecek ve olduğu gibi kullanılarak daha hızlı sonuç alınabilecektir.

İyi veri paralelleştirmesi ve iş parçacığı paralelleştirmesi için NVIDIA tarafından üç özet çıkarım yapılmıştır. CUDA programlama dilini kullanarak kod yazan programcıların hayatını kolaylaştıran bu çıkarımların listesi aşağıdaki gibi verilebilir:

- İş Parçası Grup Hiyerarşisi (Thread Group Hierarchy): İş parçaları bloklar halinde paketlenir ve bu paketlenen bloklar da tek bir grid olarak paketlenir.
- Paylaşılan Hafızalar (Shared Memories): CUDA iş parçalarının çok farklı ihtiyaçları görmek için tasarlanmış altı farklı hafızayı kullanmasına olanak tanır.

- Bariyer Senkronizasyonu (Barrier Synchronization): İş parçalarının küçük bir blok halinde eşitlenmesini ve ilgili hesaplamanın diğer parçalarının daha ileriye gitmeden bitmesini bekleyen bir iş parçacığı olmasını sağlamaktadır.

CUDA için C programlama dili, GPU üzerinde çalışabilen fonksiyonların C dilini kullanarak yazılabilmesini sağlamaktadır. Bu fonksiyonlar "çekirdekler-kernels" olarak ifade edilmektedir ve bu kerneller her bir iş parçacığı için paralel olarak yeniden üretilerek işleme alınmaktadırlar. Bu yönüyle bir defa çalışan geleneksel seri programlama fonksiyonlarından farklıdırlar. Bunu açıklayan bir örnek Şekil 3.4'de NxN boyutunda A ve B matrisinin çarpımını yapıp sonuçları C matrisine yazan bir kod parçası ile gösterilmektedir. Şekil 3.4'de de görüldüğü üzere iş parçacıkları için bloklar oluşturulmuştur. Bu bloklar bir, iki ya da üç boyutlu tanımlanabilmektedir.

```
// Kernel definition
__global__ void MatAdd(float A[N][N], float B[N][N],
                       float C[N][N])
{
    int i = threadIdx.x;
    int j = threadIdx.y;
    C[i][j] = A[i][j] + B[i][j];
}

int main()
{
    ...
    // Kernel invocation with one block of N * N * 1 threads
    int numBlocks = 1;
    dim3 threadsPerBlock(N, N);
    MatAdd<<<numBlocks, threadsPerBlock>>>(A, B, C);
    ...
}
```

Şekil 3.4 : Kernel iş parçacığının tanımlanması.

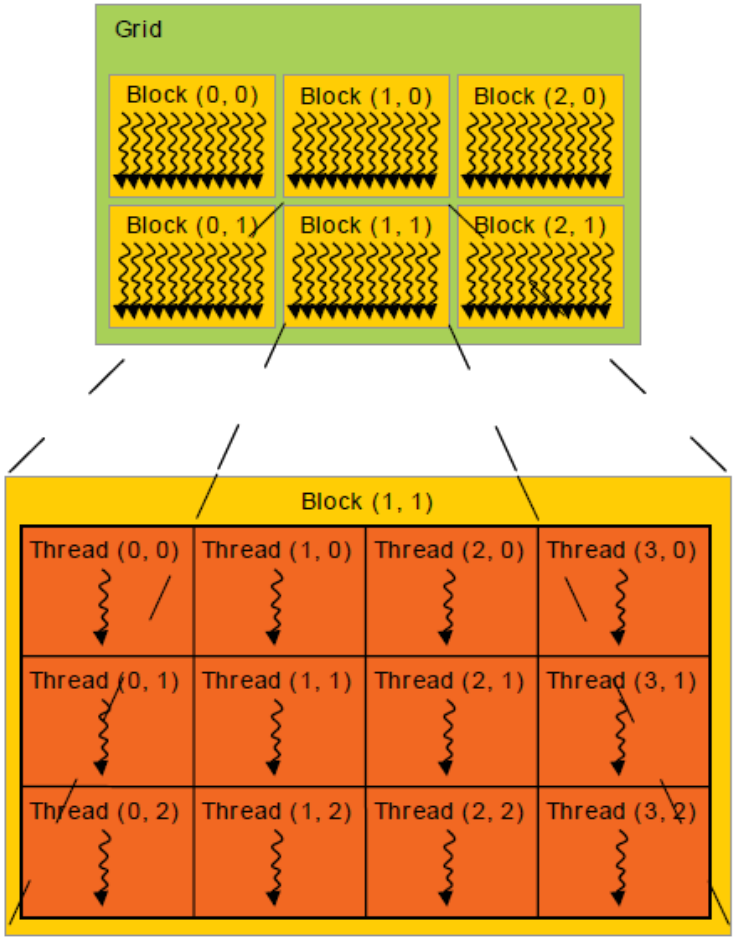
CUDA mimarisi aşağıdaki gibi baştan sona bir iş parçacığı hiyerarşisi içerisinde çalışmaktadır:

1. Grid: Bir grid bir veya iki boyutlu blokları içermektedir.
2. Bloklar: Bir blok bir, iki veya üç boyutlu iş parçalarını içermektedir. Mevcut GPU'lar bir blokta 1024 iş parçasını içerebilmektedirler. Bloklar bağımsız yürütülmekte ve ölçeklenebilir bir şekilde uygun olan işlemcilere yönlendirilebilmektedir.
3. İş Parçası (Thread): Bir iş parçası temel uygulama elemanıdır.

Yukarıda bahsedilen bu hiyerarşi ve yapı Şekil 3.5'de gösterilmiştir. Örneğin 1048576 pikselden oluşan bir görüntü, birbirinden bağımsız paralel olarak 512 blok boyutunda, 2048 grid ile işlenebilmektedir.

Her bir bloktaki iş parçacığı sayısı ve her bir griddeki blok sayısı <<<…>>> yazım şekli ve int ya da dim3 tipinde tanımlanabilmektedir. İki boyutlu bloklar veya gridlere örnek Şekil 3.4'de yer alan matris toplamı örneğinde olduğu gibi verilebilir.

Gridde yer alan her bir blok, bir, iki ya da üç boyutlu olacak şekilde tanımlanabilir ve blockIdx değişkeni ile kernelden bu indeks değerine erişilebilir. İş parçacığı bloğunun boyutuna da blockDim değişkeni ile kernelden ulaşılabilir.



Şekil 3.5 : CUDA iş parçacığı hiyerarşisi (Nvidia, 2015).

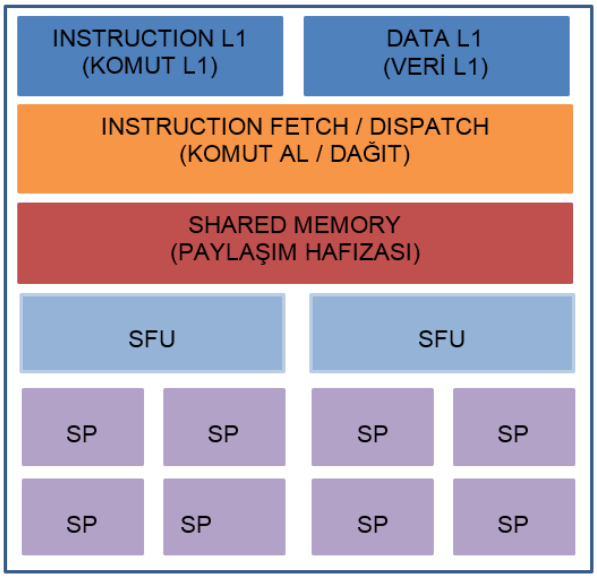
Bir önceki matris toplamı işlemini çoklu bloklara ayırdığımızda kodlama ifadesi Şekil 3.6'da gösterildiği gibi olacaktır.

```
// Kernel definition
__global__ void MatAdd(float A[N][N], float B[N][N],
float C[N][N])
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
    if (i < N && j < N)
        C[i][j] = A[i][j] + B[i][j];
}

int main()
{
    ...
    // Kernel invocation
    dim3 threadsPerBlock(16, 16);
    dim3 numBlocks(N / threadsPerBlock.x, N / threadsPerBlock.y);
    MatAdd<<<numBlocks, threadsPerBlock>>>(A, B, C);
    ...
}
```

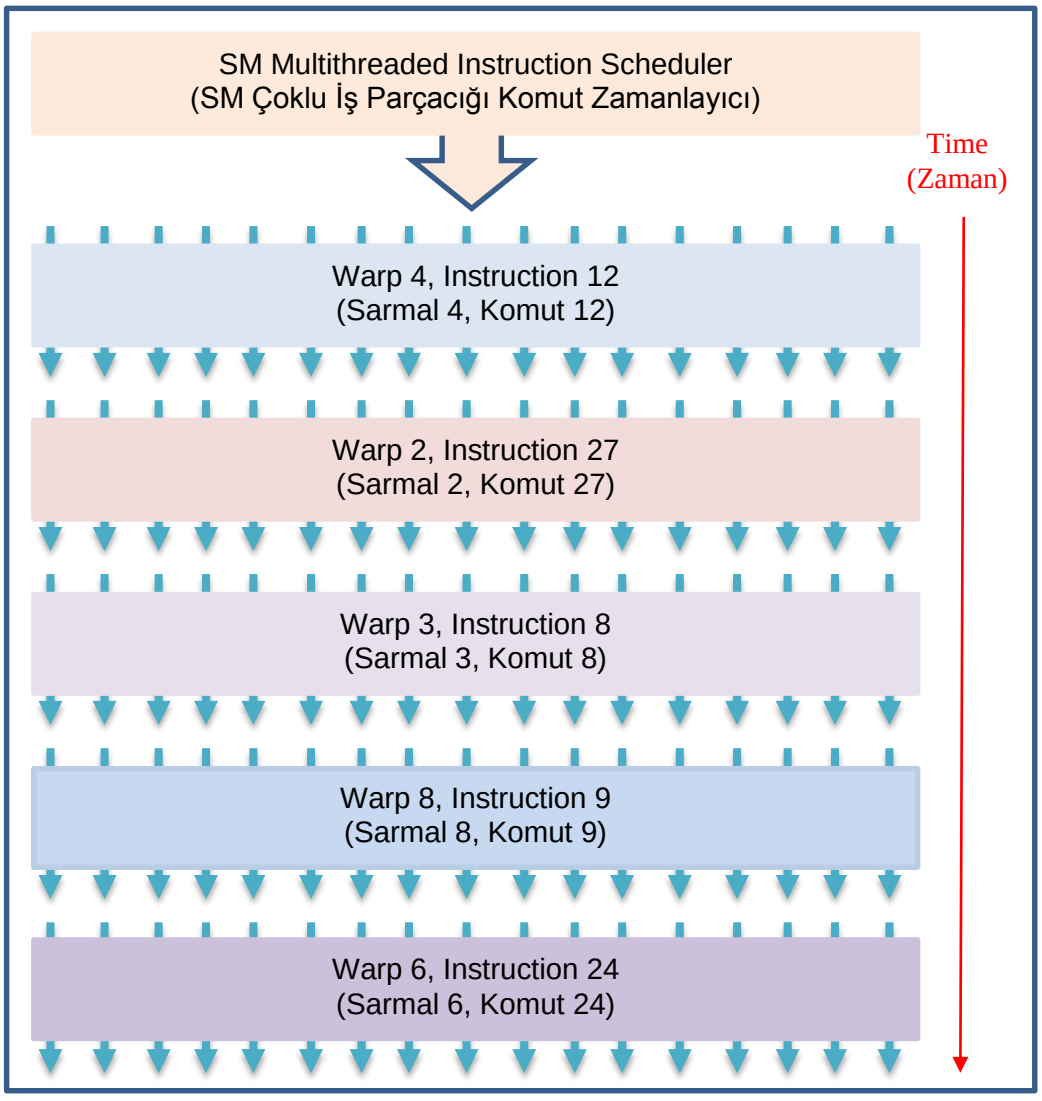
Şekil 3.6 : Bloklara ayrılan kernel tanımlaması.

CUDA işlemcisine sahip GPU'lar, birçok çoklu iş parçacığı akış çoklu işlemcisine (streaming multiprocessors-SM) sahiptir. Bir SM'nin yapısı Şekil 3.7'de verilmektedir. Örneğin GTX295 GPU'su üzerinde 60 tane SM barındırmaktadır. Her bir SM de sekiz skaler işlemciye (Scalar Processors-SP) sahiptir. Şekil 3.7 incelenecek olursa SM'in aynı zamanda iki özel fonksiyon birimine sahip olduğu görülür: bir çoklu işlem tanımlama birimi ve çip üzerinde paylaşım hafızası. Her bir SP, 32 adet iş parçacığını içeren tekil sarmal paketi eş zamanlı çalıştırabilir. Burada yer alan SFU (Special Functions Unit-Özel Fonksiyonlar Birimi) içerisinde sin, cosine, karekök ve ters alma gibi fonksiyonları içermekte ve bu fonksiyonları yerine getirmektedir.



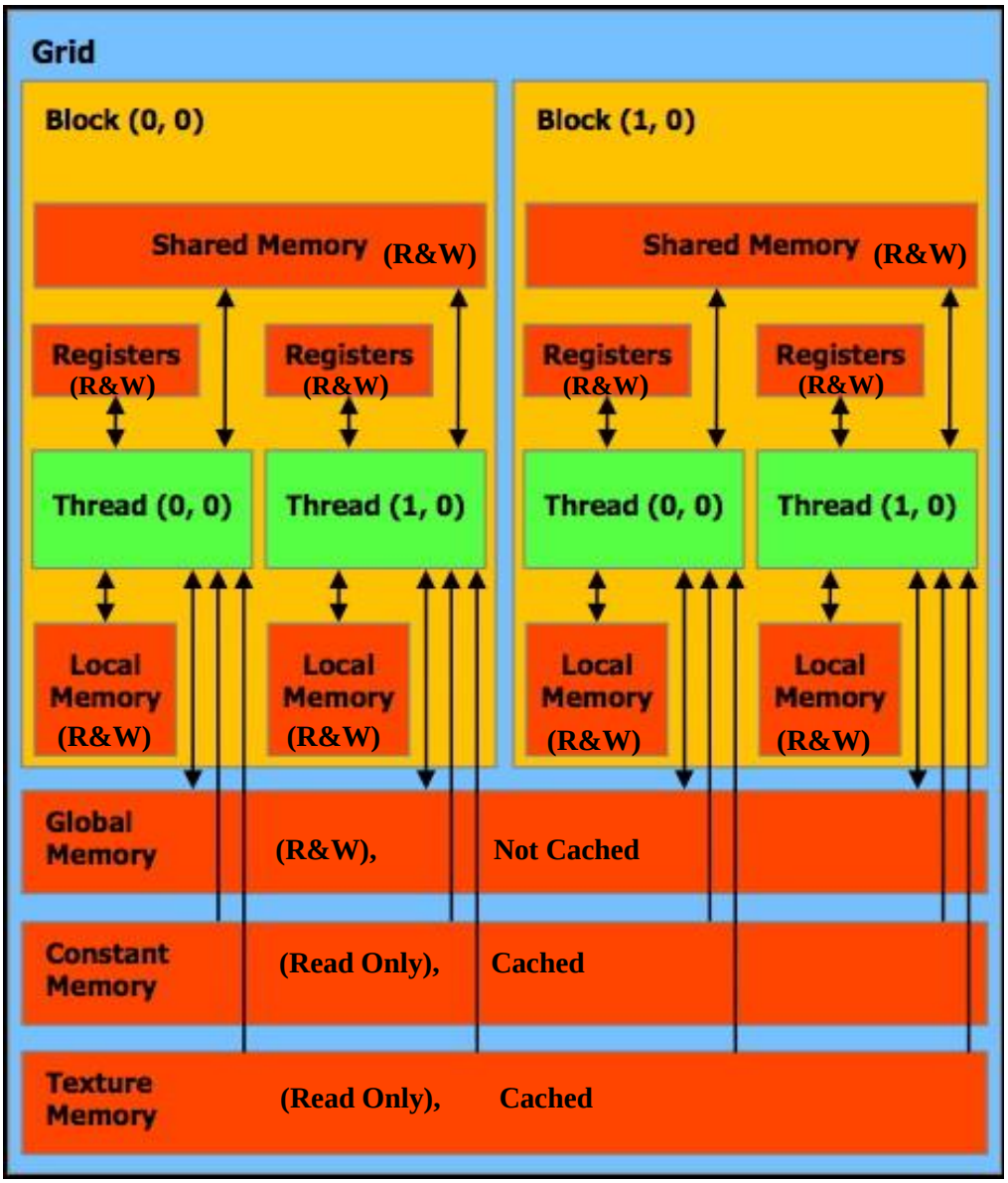
Şekil 3.7 : Akış çoklu işlemcisi (Nvidia, 2015).

SIMT (Single Instruction Multiple Thread -Tek Komutlu Çoklu İş Parçacığı) mimarisi eş zamanlı çok sayıda iş parçacığını yönetebilmektedir. Her bir tekil sarmal içindeki her türlü konuyu işleyen SIMT birimi, bütün SM'lerin herbiri için ayrı ayrı mevcuttur. SIMT'nin açıklamalı bir gösterimi Şekil3.8'de verilmiştir. Her seferinde genel komut her bir adımda aktif iş parçacığına uygulanmaktadır. Böylelikle, her iş parçacığı aynı uygulama yolunu takip ederek performansın artmasını sağlamaktadır. Fakat dallanma nedeniyle sarmal içerisindeki iş parçacıkları genel komut gelene kadar beklemektedir, bu da bir miktar gecikmeye sebep olmaktadır. Bunun önüne geçmek ve en iyi performansı yakalayabilmek için, benzer paralel iş parçacıkları sıralı olarak düzenlenmelidir. Bu da ancak algoritmanın ve programın tasarım aşamasında yapılmalıdır.



Şekil 3.8 : Tek komutlu çoklu iş parçacığı mimarisi (Nvidia, 2015).

Bu aşamadan sonra karşımıza diğer bir etken olarak GPU hafızası çıkmaktadır. GPU paralel programlamanın ne kadar etkin olduğu, GPU hafızasının ne kadar iyi kullanıldığına bağlıdır. Bir GPU hafızası, çeşitli karakteristiklerinin bilinmesini gerektiren altı farklı hafıza bölümünden oluşmaktadır. Bu hafıza bölümleri genel olarak Şekil 3.9'da gösterilmektedir.



Şekil 3.9 : GPU hafıza mimarisi (Nvidia, 2015).

- Register (Kayıt) Hafıza: Çip üzerinde bulunur ve hızlı erişimlere cevap verir. Bununla birlikte bu hafıza sadece tek bir iş parçacığı tarafından erişilebilir ve çok sınırlı bir alana sahiptir.

- Local (Yerel) Hafıza: Yerel hafıza register hafızanın ihtiyaçları karşılamadığı durumlarda ekstra alan olarak kullanılır. Bu hafıza çip üzerinde yer almaz, cachelenmez, band genişliği optimizasyonu açısından sıkıntılıdır.
- Shared (Paylaşımlı) Hafıza: Paylaşımlı hafıza performans için tasarlanmıştır. Çip üzerinde yer alan bu hafıza, akış işlemcisine yakın konumlandırılmıştır ve çok düşük gecikme süresine sahiptir. Blok içerisindeki iş parçacıkları tarafından erişilebilir ve küresel hafızadan çok daha hızlıdır. Mevcut paylaşımlı hafıza her bir blokta 16KB ile sınırlı bir alana sahiptir. Bu hafıza alanına blok içerisindeki tüm iş parçacıkları ulaşabilir. Bununla birlikte çok hızlı erişim için tasarlanmasına rağmen sınırlı kapasiteye sahiptir. Her bir blok için seçilebilecek en fazla 512 tane iş parçacığı ile sınırlıdır. Hızlı erişime ek olarak, paylaşımlı hafıza, aynı zamanda senkronizasyona da olanak tanır. Küresel hafıza yerine paylaşımlı hafızanın kullanımı, band genişliğinden büyük miktarda tasarruf edilmesini sağlar. Bununla birlikte, paylaşımlı hafıza ile algoritmaların yeniden yazılması ekstra bir çabayı gerektirir.
- Global (Küresel) Hafıza: Çip üzerinde yer almayan bu hafıza birimi, kullanılabilecek en esnek hafıza alanıdır. Tüm iş parçacıklarının ve GPU'nun okuma ve yazma işlemlerine olanak tanır. Fakat küresel hafızanın en büyük dezavantajı erişimindeki gecikmedir. Bu hafıza önbelleğe alınmadığından, daha iyi performans için kaynaştırılmış bellek erişimine ihtiyaç vardır. Ayrıca çok geniş hafıza alanına sahip değildir.
- Constant (Sabit) Hafıza: Bu hafıza da küresel hafıza gibi çip üzerinde değildir ve sadece okumaya izin vardır. Bununla birlikte cachelenebilir ve daha iyi performans verebilir. Mevcutta sabit hafıza miktarı 64KB ile sınırlıdır.
- Texture (Doku) Hafıza: Doku hafızası aslında sadece okunabilir küresel hafızadır. Küresel hafıza ile karşılaştırıldığında, bu hafızanın cache yapısı olduğu için daha iyi performansa sahiptir.

CUDA ile yazılan kodların performansının artırılması için Nvidia üç farklı stratejinin kullanılabileceğini önermektedir (Nvidia, 2015):

1. Paralel hesaplamanın performasını en üst seviyeye çıkarmak için ilk strateji önem arzetmektedir. Bu stratejide problemin paralelleştirilebilir bir yapıya

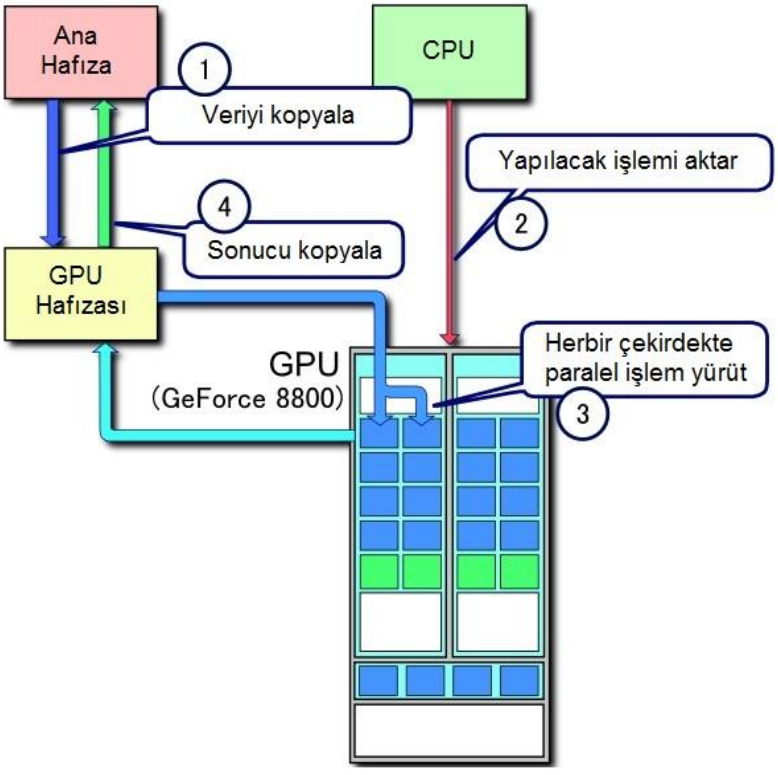
dönüştürülmesi sağlanır. Tez içerisindeki problemimiz doğrudan görüntü pikselleri üzerinde aynı işlem adımlarının yapılması olduğu için paralelleştirilebilir bir problemdir.

2. İkinci strateji band genişliğinin etkin olarak kullanılmasıdır. Bu strateji ile donanımlar arasındaki veri transferini en aza indirmek hedeflenir.
3. Üçüncü yöntem ise, problemin optimize edilmesidir. Bu strateji ile yüksek performans elde edebilmek için probleme göre alternatif yaklaşımlar değerlendirilir.

Yukarıdaki anlatımlar ışığında genel olarak CUDA işlem akışı dört adımdan oluşmaktadır ve şu şekilde özetlenebilir:

1. Veri ana hafızadan GPU hafızasına kopyalanır.
2. CPU yapılacak işlemi GPU ya aktarır.
3. GPU her bir çekirdek için paralel işlemi yürütür.
4. Sonuçlar GPU hafızasından ana hafızaya kopyalanır.

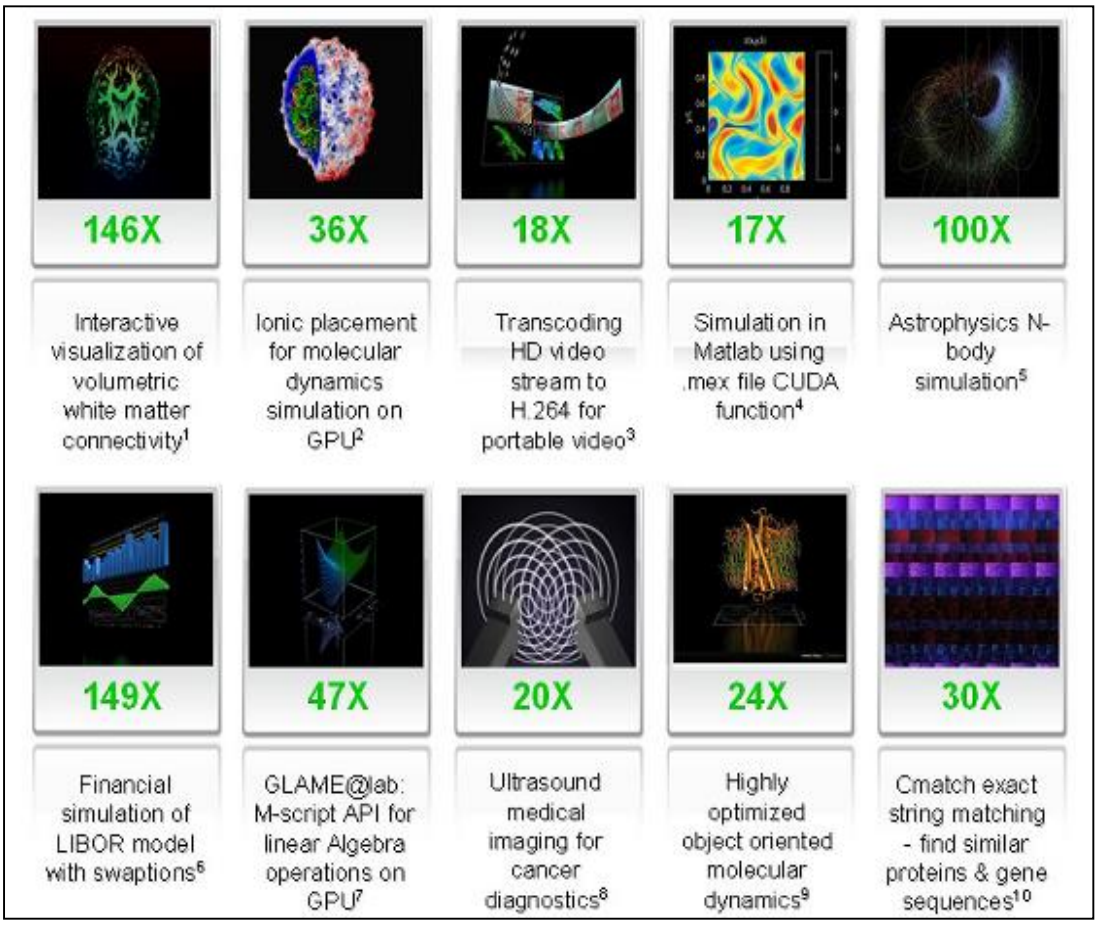
Bu adımlar bütün uygulamalar için geçerlidir. Bu adımların nasıl işlediği Şekil 3.10'da gösterilmeye çalışılmıştır.



Şekil 3.10 : CUDA işlem akışı (Nvidia, 2009).

4. UYGULAMALAR

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 gibi daha hızlı işlem yapabildiği ortaya çıkmıştır. NVIDIA firmasının yaptığı araştırmalara göre elde edilen bazı sonuçlar Şekil 4.1'de görüldüğü gibidir.



Şekil 4.1 : NVIDIA firması tarafından yapılan uygulamalarda elde edilen bazı sonuçlar (Nvidia, 2015).

Şekil 4.1 incelendiğinde, çok şaşırtıcı hız 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 ve benzeri uygulamalara ilişkin daha detaylı bilgi "http://www.nvidia.com.tr/object/gpu-computing-applications-tr.html" adresinden edinilebilir.

Tıbbi görüntüleme konu başlığı altında, son yıllarda hızla artan meme kanseri konusu incelendiğinde, kanserin erken safhada tespiti için mamografi adı verilen tıbbi görüntüleme ve ölçme yöntemi kullanılmaktadır. Fakat bu yöntemde hasta yüksek oranda X-ışınına maruz kalmakta ve bu durum hasta açısından çok riskli olmaktadır. Bu sebeple de doktorlar, X-ışınına göre daha güvenli yöntem olan "ultrasound" görüntülemeyi tercih etmektedirler. Fakat geleneksel yöntemlerin de çeşitli kısıtları bulunmaktadır. Bu noktada "TechniScan Tıp Sistemleri" adında bir firma üç boyutlu ultrasound görüntüleme yöntemini geliştirmiş ve hesaplama kısıtlarını ortadan kaldırmak için de CUDA mimarisini kullanmaya başlamıştır. Geliştirdikleri sistemin adına "TechniScan Svara" adını vermişlerdir. Bu sistem ile onbeş dakikalık tarama sonucunda elde edilen 35GB'lık görüntü verisi yirmi dakika içerisinde doktorların kullanabileceği çok detaylı üç boyutlu görüntü ve modeller haline dönüştürülmektedir.

Bir diğer CUDA ile geliştirilen örnek uygulama ise çevre bilimi üzerine verilebilir. Deterjan, şampuan gibi temizleme elemanlarının etken maddesi olarak bilinen yüzey-aktif madde (surfactants) moleküllerinin temizleme kapasiteleri çevre için en tehlikeli konulardan biridir. Bu moleküller genel anlatımla nesneler üzerindeki kirlere yapışır ve bu moleküllere yapışan kir moleküllerle birlikte nesneden su ile uzaklaştırılır. Bu noktada yeni araştırılan yüzey-aktif maddelerin temizleme kapasitelerinin ölçümü birçok materyalin kombinasyonunu içeren uzun ve geniş laboratuvar testlerine ihtiyaç duyulmaktadır. Bu işlem adımı çok ağır, uzun ve pahalı olabilmektedir. "Procter and Gamble" firması ile birlikte çalışan Temple Üniversitesi kir, su ve diğer materyallerle yüzey-aktif maddelerin etkileşimlerini ortaya çıkarmak için moleküler simülasyon üzerine çalışmaktadırlar. Geliştirilen bu simülasyon platformunda CUDA teknolojisini kullanmaktadırlar ve yaptıkları simülasyonlarda

16 kat hızlara ulaşmışlardır. Bu da aslında haftalarca sürecek olan bir simülasyonun birkaç saat içerisinde sonuçlanması anlamına gelmektedir.

Yapılan tez çalışması kapsamında çeşitli farklı uygulamalar gerçekleştirilmiştir. Bu uygulamaların test edilebilmesi için kullanılan bilgisayar; Intel Core 2 Duo 2.2 GHz işlemciye, GPU olarak 8600M GT serisi bir ekran kartına ve 3GB ram'e sahiptir.

4.1 Görüş Analizi Uygulaması

Özellikle araziye yönelik 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, kullandığı çeşitli 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. Sonuç olarak kullanılacak algoritma benzer olacaktır.

Görüş analizinde öncelikle hesaplanması gereken eğimdir. Bu amaca yönelik olarak Şekil 4.2'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 4.2 : 3x3 boyutunda arama penceresi.

Arama penceresinde yer alan "e" hücresinin "x" ve "y" yönündeki değerleri aşağıdaki (4.1) ve (4.2) denklemleri yardımı ile hesaplanır.

$$[dz/dx] = ((c + 2 * f + i) - (a + 2 * d + g))/(8 * hucrebuyuklugu) \quad (4.1)$$

$$[dz/dy] = ((g + 2 * h + i) - (a + 2 * b + c))/(8 * hucrebuyuklugu) \quad (4.2)$$

Arazinin durumuna göre "e" hücresinin eğimi radyan cinsinden (4.3) denklemi ile hesaplanabilir.

$$Egim_rad = ATAN(z_factor * \sqrt{([dz/dx]^2 + [dz/dy]^2)}) \quad (4.3)$$

Ayrıca eğimin değiştiği tarafa "eğim yönü" denilir. Bu yön radyan cinsinden 0 ile 2 değerleri aralığında değişir. Bu hesaplama için ise ilgili algoritma Şekil 4.3'deki gibi verilebilir. Bu elde edilen eğim ve eğim yönü ile aynı zamanda gölgelendirme değeri de hesaplanabilmektedir.

```
if [dz/dx] ≠0:

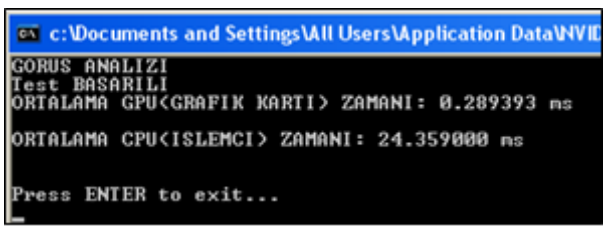
  Aspect_rad = atan2 ([dz/dy], -[dz/dx])
  if Aspect_rad < 0 then
    Aspect_rad = 2π + Aspect_rad

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 4.3 : Eğim yönü hesabı algoritması.

Görüş analizi problemi dikkate alındığında aynı hesaplama yönteminin sırayla bütün yükseklik verisini oluşturan tüm grid hücrelerine ayrı ayrı uygulanmasının gerektiği görülmektedir. Bu durumda aslında karşımıza çoklu işlem problemi çıkmaktadır. Bu durum da mevcut problemin çözümünde GPGPU'nun kullanılabilmesini olanaklı hale getirmektedir. Böylelikle problem CUDA programlama dili kullanılarak yeniden ele alınmış, problem iş parçalarına bölünüp dağıtık bir şekilde, birçok GPU işlemcisine aynı anda verilip, sonuç hesaplamalar elde edilebilir hale getirilerek, çok hızlı sonuçlar elde edilebilmiştir.

Hazırlanan program parçası hem CUDA mimarisini kullanarak grafik kartı üzerindeki işlemcileri (GPU) kullanarak işlem yapmakta, hem de mevcut CPU'yu kullanarak aynı işlemi tekrarlayabilmektedir. Uygulamanın çalıştırılması ile elde edilen işlem sürelerine ilişkin sonuçlar Şekil 4.4'deki gibi bir arayüz yardımı ile görülebilmektedir.



Şekil 4.4 : Görüş analizi uygulamasının gösterim arayüzü.

Şekil 4.4 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. Elde edilen sonuçlar Çizelge 4.1'de detaylı olarak verilmiştir. Çizelge incelendiğinde verinin boyutunun küçülmesiyle, CPU'nun performansının iyileştiği görülmektedir. Fakat bunun yanında GPU'nun performansı aynı kalmaktadır. Bu da çok yoğun ve tekrarlı hesap gerektiren problemlerde GPU'nun daha hızlı sonuçlar ürettiğini ortaya koymaktadır.

Çizelge 4.1 : Görüş analizinde, GPU ve CPU ile yapılan hesaplama hızı sonuçları.

Raster Hücre Boyutu	GPU zamanı (milisaniye)	CPU zamanı (milisaniye)	CPU / GPU (oran) kat
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

4.2 Gauss Bulanık (Gaussian Blur) Filtreleme Uygulaması

Gauss bulanık filtreleme algoritması, özellikle görüntülerin yumuşatılması için kullanılan bir filtreleme yöntemidir. Fotogrametride de çeşitli uygulamalar içerisinde kullanılmaktadır. Bu filtrenin uygulanması ile görüntülerdeki gürültü ve istenilen detaylar azaltılabilmektedir. Bu yöntem içerisinde Gauss normal dağılım fonksiyonundan faydalanılır. Gauss bulanık filtreleme, yöntem olarak aslında ortalama filtreye benzer.

Bir boyutlu Gauss fonksiyonu (4.4) denklemiyle formüle edilebilir.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{x^2}{2\sigma^2}} \tag{4.4}$$

Ayrıca iki boyutlu Gauss fonksiyonu ise (4.5) denklemiyle formüle edilebilir.

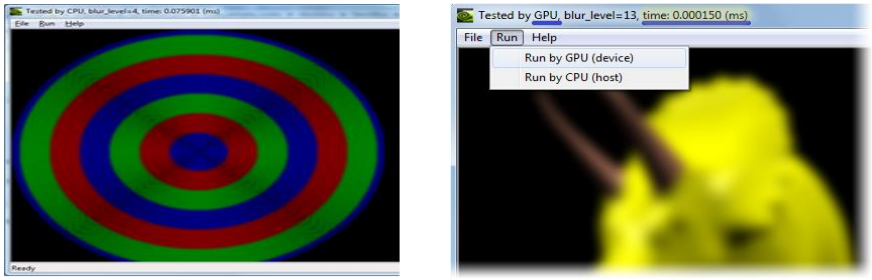
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.4}$$

Bu denklemlerde:

- x = yatay eksende merkezden mesafesi
- y = dikey eksende merkezden mesafesi
- σ = Gauss dağılımının standart sapmasıdır.

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 hesaplanır. Bir önceki uygulamada olduğu gibi bu uygulamada da aslında aynı hesaplama adımlarının birden fazla piksele uygulanması, haliyle çoklu işlem problemi söz konusudur. Bu noktada benzer olarak GPGPU yönteminin kullanılabilmesi mümkündür.

Yöntemin mevcut matematiksel denklemleri kullanılarak elde edilen algoritma, CUDA ile kodlanmış ve elde edilen uygulama arayüzü Şekil 4.5'de görülmektedir.



Şekil 4.5 : Gauss bulanık filtreleme uygulamasının arayüz örnekleri.

Uygulama içerisinde bulanıklık seviyesi değiştirilebilmektedir. Aynı zamanda yapılan işlemin süresi bakımından GPU ve CPU karşılaştırması da yapılabilmektedir. Uygulamadan elde edilen sonuçlar Çizelge 4.2'de verilmiştir. Bu çizelge incelendiğinde GPU ile yapılan bulanıklaştırma işleminin, CPU ile yapılan işleme oranla yaklaşık 450 kata kadar daha hızlı yapıldığı görülmektedir.

Çizelge 4.2 : GPU ve CPU ile yapılan Gauss bulanık filtreleme hızı sonuçları.

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

4.3 Uygulamalarda Kullanılan Görüntülerin Teknik Özellikleri

Uygulamalar içerisinde UltracamX Vexcel sayısal kamerasından elde edilen çeşitli görüntüler kullanılmıştır. Kullanılan görüntülerin ortak özellikleri Çizelge 4.3'de verilmektedir. Ayrıca bu görüntülere bir örnek Şekil 4.6'daki gibidir.

Çizelge 4.3 : UltracamX Vexcel sayısal kameranın teknik özellikleri.

Pankromatik görüntü boyutu	14430 x 9420 piksel
Pankromatik fiziksel piksel boyutu	7.2 µm
Her bir görüntüdeki girdi verisi	435 Mega Byte
Görüntü düzleminin boyutları	104 mm x 68.4 mm
Pankromatik lens odak uzaklığı	100 mm
Lens açıklığı	f=1/5.6
Nadirden olan enine görüş açısı (boyuna)	55° (37°)
Renkli 4 kanal	RGB & NIR
Renkli görüntü boyutu	4810 x 3140 piksel
Renkli piksel boyutu	7.2 µm
Renkli lens odak uzaklığı	33 mm
Renkli lens açıklığı	f=1/4.0
Renkli Nadirden olan enine görüş açısı (boyuna)	55° (37°)
Çekim hızı	1/500 - 1/32 aralığında
İleri hareket dengeleme (Forward Motion Compensation - FMC)	TDI kontollü
Maksimum FMC imkanı	50 piksel
500 m uçuş yüksekliğinde yerdeki piksel boyutu (300m'de)	3.6 cm (2.2 cm)
Saniyedeki çerçeve miktarı	1.35 saniyede 1 çerçeve
Analogdan dijitale dönüşümde, her bir kanaldaki radyometrik çözünürlük	14 bit



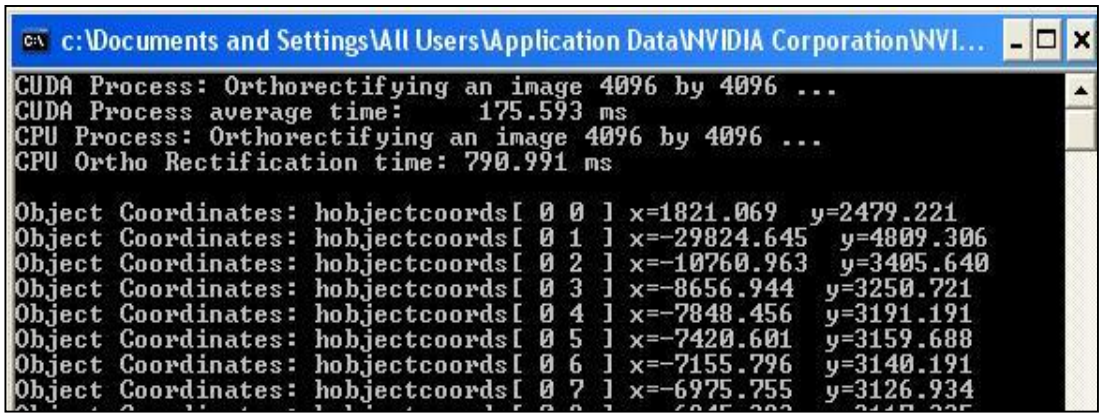
Şekil 4.6 : Uygulamalarda kullanılan görüntülere örnek.

4.4 Projektif Dönüşüm Uygulaması

Ortorektifikasyon amacıyla kullanılabilecek rektifikasyon yöntemlerinden bir olan projektif dönüşüm yöntemi, çalışma içerisinde örnek bir uygulama için kullanılmıştır. Örnek uygulamada bir düzlem alanda çekilen 4096x4096 piksellik bir örnek görüntü değerlendirilmiştir. Bu uygulama içerisinde kodlanacak program için işlem adımları aşağıdaki gibi özetlenebilir:

1. Görüntü Koordinatları ve Obje Koordinatları girdi olarak kullanılır.
2. Matris yapıda denklem sistemi yazılır ve bu denklem sistemi çözülerek katsayılar hesaplanır.
3. Katsayıları hesaplanan denklemler ile bilinmeyen obje koordinatları, görüntü üzerindeki her bir pikselin görüntü koordinatlarından hesaplanır.
4. Hesaplanan obje koordinatları bir dosyaya yazdırılır.
5. 3 ve 4 nolu işlem adımları her bir piksel için tekrarlanır.

Yukarıda verilen işlem adımları da dikkate alınarak 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 geliştirilen algoritma CUDA programlama dili kullanılarak kodlanmış ve programın hem CPU, hem de GPU'yu kullanarak aynı işlemleri yapması sağlanarak, işlem süreleri arasındaki farklar karşılaştırılmıştır. Kodlanan program çalıştırıldığında elde edilen ekran görüntüsü Şekil 4.7'deki gibidir.



Şekil 4.7 : Projektif dönüşüm ile ortorektifikasyon uygulamasının çıktı arayüzü.

Yapılan çalışmada uygulanan projektif dönüşüm ile ortorektifikasyon 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, hesaplama sonuçları 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 4.7'de görülmektedir.

Projektif dönüşüm ile ortorektifikasyon uygulamasının hesaplama sonuçları 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 kullanılan 4096x4096 boyutunda bir görüntü yerine daha küçük boyutlarda görüntülere aynı işlem uygulandığında CPU'nun performansının iyileştiği Çizelge 4.4'de görülmektedir. Bu da çok yoğun ve tekrarlı hesap gerektiren problemlerde GPU'nun daha hızlı sonuçlar ürettiğini ortaya koymaktadır. Aynı uygulama içerisinde 14430 x 9420 boyutundaki örnek bir görüntü de uygulamaya sokulmuş ve Çizelge 4.4'de gösterilen hız farkına ulaşılmıştır.

Çizelge 4.4 : GPU ve CPU ile yapılan uygulamanın hız testi sonuçları.

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
14430 x 9420	414,238	2823,654	6,82

4.5 Diferansiyel Ortorektifikasyon Uygulaması

Gerçek zamanlı ortorektifikasyona ihtiyaç duyulan çalışmalarda, görüntülerin doğrudan ortorektifikasyonu aşamasında diferansiyel ortorektifikasyonun kullanılabileceği değerlendirilerek çalışma içerisinde diferansiyel ortorektifikasyon yöntemi de kullanılmıştır. Doğrudan ortorektifikasyonda asıl amaç algılayıcının dönüklük parametreleri ile konumunu belirleyebilmektir. Bu olanak, algılayıcı konumu ve hassas dönüklük parametrelerine ihtiyaç duyulan veri toplama ve çeşitli haritalama uygulamalarında kullanılabilir bir teknoloji halini almıştır. Yöntem ile ilgili ayrıntılı açıklama ikinci bölümde yapılarak, yöntemin programlanabilmesi için gerekli algoritma ve akış diyagramı da Şekil 2.3'de detaylı bir şekilde açıklanarak verilmişti.

Uygulama, projektif rektifikasyonda olduğu gibi, görüntüyü oluşturan her bir piksel için tekrarlanmaktadır. Böylelikle de CUDA programlama dili kullanılarak problem iş parçalarına bölünüp dağıtık bir şekilde, birçok grafik işlemciye aynı anda verilip, hesaplama sonuçları elde edilebilecek yapıdadır.

Bu noktadan hareketle, diferansiyel ortorektifikasyon için ortaya konulan algoritma, CUDA programlama dili ile kodlanmış ve diferansiyel ortorektifikasyonu gerçekleştiren bir uygulama ortaya çıkarılmıştır. Uygulama ile diferansiyel ortorektifikasyon hem grafik kartı üzerindeki işlemciler kullanılarak yapılabilmekte, hem de mevcut CPU'yu kullanılarak yapılabilmektedir. Her iki farklı yöntem için de işlem süreleri Çizelge 4.5'de verilmiştir. Sonuç görüntü de Şekil 4.8'deki gibidir.

Çizelge 4.5 : GPU ve CPU ile yapılan uygulamanın hız testi sonuçları.

Görüntü Boyutu (piksel)	GPU Zamanı (milisaniye)	CPU Zamanı (milisaniye)	CPU / GPU (oran)
1024 x 1024	84675	324563	3,83
2048 x 2048	342634	934547	2,73
4096 x 4096	1328234	6827282	5,14
14430 x 9420	3824302	28129128	7,35



Şekil 4.8 : Ortorektifikasyon sonucu elde edilen görüntü örneği.

4.6 Uygulama Sonuçlarının Değerlendirilmesi

Çeşitli platformlarla elde edilen görüntülerin hızlı bir şekilde ortorektifiye edilmesi ihtiyacı, sürekli gündemde olmuştur. Bu konu üzerine birçok bilimsel çalışma yapılmıştır. Thomas ve diğ. (2008) tarafından yapılan çalışma incelendiğinde görüntülerin anlık ortorektifikasyonu için toplam beş adet çok güçlü donanıma sahip bilgisayarın kullanıldığı görülmektedir. Bu çalışma içerisinde 2048x2048 piksellik bir görüntünün ortorektifikasyonu için elde edilen süre 95 milisaniye olarak verilmiştir. Yapılan bu tez çalışması içerisinde uygulanan GPGPU yöntemi ile, aynı boyutlardaki görüntünün ortorektifikasyonu için harcanan süre yaklaşık 346 milisaniyedir. Bu elde edilen sonuç, çok basit bir ekran kartı ile alınmıştır. Haliyle 5 tane çok güçlü, paralel çalışan bilgisayarlardan elde edilen zaman sonucu dikkate alındığında, tek bir GPU ile elde edilen sonucun çok daha başarılı olduğu ortadadır. Donanım altyapısının geliştirilmesi ve daha güçlü yeni nesil bir GPU'nun kullanımıyla sonucun çok daha başarılı olacağı değerlendirilmektedir.

Yang ve diğ. (2008) tarafından görüntü işlemede CUDA teknolojisinin kullanıldığı ve sonuçlarının verildiği bilimsel bir çalışma yapılmıştır. Yapılan çalışmada çeşitli görüntüler üzerinde yer alan bulut tabakasının kaldırılması için çeşitli görüntü işleme teknikleri kullanılmıştır. Yapılan çalışmada, bu tez çalışmasındaki ile aynı ekran kartı ve donanım altyapısı kullanılmıştır. Bu noktada 4096x4096 piksellik bir görüntünün CPU ile işlenmesinin 32 saniye, GPU ile işlenmesinin ise sadece 0,4 saniye aldığı ortaya konulmuştur. Bu da GPU'nun CPU'ya göre aynı işlemi 80 kat daha hızlı yapabildiğini göstermiştir.

Çizelge 4.6 : GPU ve CPU ile yapılan görüntü üzerinden bulut kaldırılması uygulamasının hız testi sonuçları (Yang ve diğ, 2008).

Görüntü Boyutu (piksel)	GPU Zamanı (milisaniye)	CPU Zamanı (milisaniye)	CPU / GPU (oran)
4096x4096	402,01	31925,99	79,41
2048x2048	97,09	7644,09	78,73
1024x1024	22,53	1778,28	78,93
700x525	21,98	1745,02	79,39

Plaza ve diğ. (2009) tarafından hiperspektral görüntülerin sıkıştırılması için GPGPU yöntemi kullanılmış ve Çizelge 4.6'daki sonuçlara ulaşılmıştır. Sonuçlar incelendiğinde 25 kat hız farklarına ulaşılmıştır. Bu elde edilen sonuçlar da, bu tez

çalışmasında kullanılan donanım altyapısına benzer bir donanımla elde edilmiştir. Benzer şekilde Santos ve diğ. (2013) tarafından da hiperspektral görüntülerin sıkıştırılması üzerine bir çalışma gerçekleştirilmiştir. Bu çalışmada da, GPU'nun CPU'ya oranla yaklaşık 15 kat daha hızlı sıkıştırma işlemini gerçekleştirdiği sonucuna ulaşılmıştır. Bu çalışmalardan yola çıkılarak mevcut modern fotogrametrik kamera sistemlerinde, görüntülerin daha henüz platform üzerinde çekildikten hemen sonra, GPU'lar sayesinde bir ön işleme tabi tutulmaksı sağlanmıştır. Böylelikle ofis ortamına ön işlemesi tamamlanmış görüntüler aktarılabilmektedir. Bu da görüntü işleme süresini azaltmaktadır.

Çizelge 4.7 : GPU ve CPU ile yapılan hiperspektral görüntü sıkıştırma uygulamasının hız testi sonuçları (Plaza ve diğ, 2009).

Görüntü Boyutu (MB)	GPU Zamanı (milisaniye)	CPU Zamanı (milisaniye)	CPU / GPU (oran)
18	6,72	172,94	25,73
36	13,88	341,06	24,57
70	26,12	675,21	25,85
140	52,60	1345,18	25,57

Stereo görüntülerden nokta bulutu oluşturmaya yönelik olarak, Bae Systems firmasına danışmanlık yapan Zhang (2015) tarafından gerçekleştirilen çalışma incelendiğinde de, Nvidia Quadro K5000 serisi bir GPU ile yapılan çalışmada Intel Xeon 3GHz'lik bir işlemciye göre 9 kat hız farklarına ulaşılmıştır.

Zhang ve diğ. (2011) tarafından İHA'lar ile elde edilen görüntü çiftlerinden otomatik nokta bulutu oluşturmak için yapılan çalışmalar incelendiğinde, bu çalışmalarda Nvidia Quadro K5000 serisi bir GPU ile Intel Xeon 3GHz'lik bir CPU kullanıldığı ve kullanılan GPU'nun CPU'ya göre ortalama 5 kat hız artırımı sağladığı ortaya konulmuştur.

Fang ve diğ. (2014) tarafından gerçekleştirilen ve optik uydu görüntülerinin relatif radyometrik düzeltme, medyan filtreleme, geometric düzeltme gibi ön işleme faaliyetlerinin çoklu GPU'lar ile yapılabileceği üzerine gerçekleştirilen çalışmada, GPGPU yöntemi kullanılmıştır. Bu çalışmada donanım olarak Nvidia Tesla M2050 GPU, Intel Xeon E5650 CPU kulanılmıştır. Çalışma sonucunda 24576x24576 piksel boyutunda toplam 90 GB'lık 80 adet görüntünün işlenmesi, GPU'da 41 saniye,

CPU’da 74 dakika sürmüştür. Bu da GPU’nun CPU’ya göre yaklaşık 108 kat daha hızlı olduğunu göstermektedir.

Yapılan tüm bu çalışmalarla, tez çalışması sonucunda elde edilen sonuçlar incelendiğinde, birbirleri arasında bir paralellik olduğu açıkça görülmektedir. Görüntü işlemenin uygulandığı, paraleleştirilebilir tüm işlem adımlarında GPU’ların CPU’lara oranla çok önemli hız artırımları yaratacağı değerlendirilmektedir.

5. SONUÇ VE ÖNERİLER

Günümüzde modern GPU'lar sadece çok güçlü grafik motoru olmaktan çıkarak, bilgisayar işlemcilerine göre aritmetik işlem yapabilme hızı ve hafıza band genişliği hızı çok daha yüksek olan ve üst seviyede paralel programlanabilir işlemciler halini almışlardır. Grafik işlemcilerin bilgisayar işlemcilerine bir alternatif olarak gündeme gelmesinin asıl nedeni ise; çok güçlü ve bunun yanında ucuz temin edilebilir donanım olmalarıdır. Böylelikle bu grafik çipler, sabit uygulama donanımları olmaktan çıkarak günümüzde modern, güçlü ve programlanabilir genel ihtiyaçları karşılayabilecek işlemcilere dönüşmüşlerdir.

GPU'lar 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 çalışmada grafik kartlarının, yani GPU'ların, GPGPU yönteminin genel amaçlı paralel programlama ve hesaplama gücünün fotogrametrik çalışmalar amacıyla ve özellikle ortorektifikasyon uygulamalarında nasıl kullanılabildiği üzerinde durulmuş, uygulamalar yapılmış ve çeşitli sonuçlar elde edilmiştir.

5.1 Sonuçlar

Çalışma içerisinde yapılan uygulamalardan da görüleceği gibi, 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.

Uygulamalar sonucunda elde edilen sonuçlar değerlendirildiğinde, GPGPU yönteminin, 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. Özellikle ortorektifikasyon amacıyla yapılan uygulamalarda, aynı donanımla CPU'ya oranla 7 kat hız farklarına ulaşılmıştır. Sonuçlar biraz daha detaylı incelendiğinde görüntü boyutu büyüdükçe CPU'nun performansının düştüğü, GPU'nun ve GPGPU yönteminin performansının arttığı 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 yöntemlerin uygulanabilir olduğu görülmektedir.

Günümüzde harita üretiminde, özellikle görüntüler üzerinden değerlendirme sonuçlarının hızla elde edilmesi ve karar vericilere aktarılmasının gerektiği felaket senaryolarında, orman yangınları ve depremler gibi yaşanan doğal afetlerde ve kriz durumlarında, özellikle askeri açıdan hedef istihbaratı ve hedef konumunun hızlıca tespitinde çok çeşitli platformlardan elde edilen görüntüler sıklıkla kullanılmaktadırlar. Bu görüntülerin anlamlı hale gelmesi ve belirtilen uygulamalarda kullanılabilmesi için öncelikli olarak yapılması gereken işlem adımı; bu görüntülerin ortorektifikasyonudur.

Özellikle son zamanlarda yaygınlaşarak çeşitli kurum ve kuruluşlar tarafından birçok uygulamada kullanılmaya başlanan İHA'lar yardımıyla elde edilen görüntüler üzerinden çok hızlı karar vermek, çıkarımlar yapmak ve çeşitli hedef tespiti yaparak bu hedefleri de doğru koordinatlarla tarifleyebilmek için, elde edilen görüntülerin yöneltilmesi yani ortorektifiye edilmesi ihtiyacı bulunmaktadır. Bu amaca yönelik olarak çekilen görüntüler bir depolama ünitesinde depolanmakta ve ardından elde edilen görüntüler İHA yere indirildikten sonra yerdeki istasyonlarda yer alan güçlü işlemcilere ve donanıma sahip bilgisayarlar yardımıyla ortorektifikasyon işlemine tabi tutularak koordinatlandırılmakta ve bu aşamadan sonra da koordinatlı olarak kullanılabilir hale gelmektedir.

Bu kapsamda uygulama içerisinde ortaya konulan yöntemle; GPU'ların genel amaçlı paralel programlama ve hesaplama gücünü kullanarak GPGPU yöntemiyle sayısal hava kameraları ile elde edilen görüntülerin, insansız hava araçları gibi çok çeşitli platformlardan elde edilebilecek görüntülerin hızlı bir şekilde gerçek zamanlı

ortorektifikasyonunun yapılabileceği değerlendirilmektedir. Bu noktada elde edilen sonuçlar incelendiğinde ortorektifikasyon işleminde 1024x1024 piksel büyüklüğünde bir görüntü için 3,83 kat, 14430 x 9420 piksel boyutundaki bir görüntü için ise 7,35 kat GPU'ların CPU'lara göre hızlı olduğu belirlenmiştir.

Yapılan bu çalışma ile daha önce çeşitli platformlardan elde edilen görüntülerin ortorektifikasyonu için kullanılmamış olan bir yöntem ve donanım altyapısı kullanılarak, programlama algoritması uygulanmıştır. Çalışma içerisinde gerçekleştirilen uygulamalar ile GPGPU yöntemi kullanılarak görüntülerin çok hızlı ve donanım olarak çok daha ekonomik bir şekilde ortorektifikasyonunun yapılabileceği değerlendirilmektedir. Böylece, bu çalışmayla halen anlık karar verme ve hedef tespiti için çok önemli bir unsur olan zaman faktörünün en aza indirilmesi için bir yöntem ortaya konulmuş ve bu yöntemin geliştirilmesi için de bir başlangıç yapılmıştır.

5.2 Öneriler

İHA'ları olumsuz yönde etkileyen en büyük etken faydalı yük ağırlığıdır. Örneğin İHA üzerine takılan kamera faydalı yük olarak değerlendirilmektedir. İHA'ların taşıyabileceği faydalı yük miktarı İHA'nın kabiliyetine göre değişmekle birlikte, belirli ve sınırlıdır. O nedenle İHA üzerinde yer alan kamera yanında işlem kapasitesi yüksek bir bilgisayar konulması ekstra bir yük olacağı için mümkün olamamaktadır. Bu noktada GPGPU yöntemi kullanılarak, İHA üzerine takılabilecek hafif ve daha küçük kapasitede, üzerinde GPU taşıyan bilgisayarlar ile ortorektifikasyon işleminin görüntü alımıyla birlikte platform üzerinde yapılabileceği ve gerçek zamanlı ya da gerçek zamana yakın ortorektifiye edilmiş görüntüler elde edilebileceği değerlendirilmektedir. Bu konu üzerine çalışma ilerletilirse gerçekleştirilebileceği ve olumlu sonuçlar elde edilebileceği düşünülmektedir.

Çalışma kapsamında yapılan ve ortaya konulan uygulamalarda yazılan CUDA kodu içerisinde herhangi bir optimizasyon çalışması yapılmamıştır. Literatür incelendiğinde GPGPU'nun uygulandığı başka uygulamalarda, CUDA kodunun optimize edilmesiyle performansın arttığı görülmüştür. Çalışma kapsamındaki uygulama içerisinde de şayet bu optimizasyon gerçekleştirilmiş olsaydı, uygulamaların performansının artması kaçınılmaz olacaktır. İlerleyen çalışmalarda

bu optimizasyonun da yapılarak uygulamaların performanslarının artırılacağı değerlendirilmektedir.

Mevcut yazılan uygulama için kullanıcı dostu bir arayüz geliştirilememiştir. Uygulama çalışma içerisinde tümüyle test amacıyla geliştirilmiştir. Bu nedenle ilerleyen çalışmalarda bu uygulama arayüzünün de geliştirilmesi, kullanıcılar açısından faydalı olacaktır.

Güncel literatür incelendiğinde, GPU ile yapılan hesaplama işlemlerinin CPU ile yapılan hesaplama işlemlerine göre daha doğru sonuçlar verdiği görülmektedir. Uygulamalar üzerine yapılacak ileriki çalışmalarda, yapılan hesaplamaların ve sonuçların CPU ile yapılan hesaplamalara göre ne kadar doğru olduğunun test edilebileceği ve buna yönelik çeşitli uygulamalar geliştirilebileceği mümkün olabilecektir.

KAYNAKLAR

Bäumker, M. & Heimes, F.J. (2001). New Calibration and Computing Method for Direct Georeferencing of Image and Scanner Data Using the Position and Angular Data of an Hybrid Inertial Navigation System, *Proceedings OEEPE Workshop Integrated Sensor Orientation*, Hannover, Germany; September 17-18.

Bettemir, O.H. (2006). *Sensitivity And Error Analysis of A Differential Rectification Method For CCD Frame Cameras and Pushbroom Scanners.* (Yüksek Lisans tezi). Ortadoğu Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Ankara.

Biesemans, J. & Everaerts, J. (2006). Image Processing Workflow for the Pegasus HALE UAV Payload, *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XXXVI-1/W44 Antwerp, Belgium.

Biesemans, J., Sterckx, S., Knaeps, E., Vreys, K., Adriaensen, S., Hooyberghs, J., Meuleman, K., Kempeneers, P., Deronde, B., Everaerts, J., Schläpfer, D., Nieke, J. (2007). Image processing workflows for airborne remote sensing, *5th EARSeL SIG Imaging Spectroscopy Workshop*, Bruges, Belgium.

Colomina, I. (1999). GPS, INS and Aerial Triangulation: What is the Best Way for the Operational Determination of Photogrammetric Image Orientation, *The International Archives of the Photogrammetry and Remote Sensing*, Vol. XXXII, Part 3-2W5, München, Germany.

Cramer, M. & Stallmann, D. (2001). On the use of GPS/inertial exterior orientation parameters in airborne photogrammetry, *OEEPE Workshop, Integrated Sensor Orientation*, Hannover, Germany.

Fang, L., Wang, M., Ying, H., Hu, F. (2014). Multi-GPU Based Near Real-Time Preprocessing and Releasing System of Optical Satellite Images, *International Geoscience and Remote Sensing Symposium*, Québec, Canada; July 13-18.

Farrell, J. & Barth, M. (1999). *The Global Positioning System and Inertial Navigation*, McGraw-Hill, NewYork, USA.

Gruen, A. & Beyer, H. (2001). *Calibration and Orientation of Cameras in Computer Vision*, Springer Series in Information Sciences Springer, Vol. 34,Verlag Berlin Heidelberg, Germany.

Heipke, C., Jacobsen, K., Wegmann, H., (2001). Analysis of the Results of the OEEPE Test "Integrated Sensor Orientation, *Proceedings OEEPE Workshop Integrated Sensor Orientation*, Hannover, Germany; September 17-18.

Hemmleb, M. & Wiedemann, A., (1997). Digital Rectification and Generation of Orthoimages in Architectural Photogrammetry, *CIPA International Symposium, IAPRS*, XXXII, Part 5C1B: 261-267, Göteborg, Sweden.

Jacobsen, K. (2002). Calibration aspects in direct georeferencing of frame imagery, *International Archives PhRS (34)*, 1 I, pp. 82-89, Denver, USA.

Jacobsen, K. & Wegmann, H. (2001). Dependicies and Problems of Direct Sensor Orientation, *OEEPE Workshop, Integrated Sensor Orientation*, Hannover, Germany.

Karslioglu, M., O. & Friedrich J. (2005). A New Differential Geometric Method to Rectify Digital Images of the Earth's Surface Using Isothermal Coordinates, *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 43, No. 3, March.

Kiracı A., C. (2008). *Direct Georeferencing and Orthorectification of Airborne Digital Images*. (Yüksek Lisans tezi). Ortadoğu Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Ankara.

Kraus, K. (2007). *Fotogrametri Cilt 1 Fotoğraflardan ve Lazer Tarama Verilerinden Geometrik Bilgiler* (O.Altan, S.Külür, G.Toz, H.Demirel, Z.Duran ve M.Çelikoyan, Çev.). İstanbul Teknik Üniversitesi, İstanbul: Nobel Yayın Dağıtım.

Kurz, F., Müller, R., Stephani, M., Reinartz, P., Schroeder, M. (2007). Calibration of a wide-angle digital camera system for near real time scenarios, *Proc. ISPRS Hannover Workshop High Resolution Earth Imaging for Geospatial Information*, Hannover, Germany; May 29-June 1.

Lozano, O., M. & Otsuka, K. (2008). Real-Time Visual Tracker by Stream Processing, *Journal of Signal Processing Systems*, New York, USA.

Marqu´es, M., Quintana-Ort´ı, G., Enrique S., Quintana, O., Geijn, R., (2009). Using graphics processors to accelerate the solution of out-of-core linear systems, *8th IEEE International Symposium on Parallel and Distributed Computing*, Lisbon, Portekiz.

Mostafa, M. (2001). Direct Georeferencing Column: An Introduction, *Photogrammetric Engineering & Remote Sensing*.

Müller, Ru., Lehner, M., Reinartz, P., Schroeder; M. (1999). Procedures for geometric rectification of airborne scanner data, *Proceedings of Workshop on In-orbit characterization of optical image systems*, Bordeaux, France.

Müller, R., Lehner, M., Reinartz, P., and Schröder, M. (2002). A program for direct georeferencing of airborne and spaceborne line scanner images, *In Proceedings of the ISPRS Commission I Symposium: Integrating Remote Sensing at the Global, Regional and Local Scale*, Denver, USA; November 10–15.

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

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

Nvidia, (2011). *OpenCL Programming Guide for the CUDA Architecture*, Nvidia Corp., California, USA.

Nvidia, (2015). *CUDA C Programming Guide*, Nvidia Corp., California, USA.

Plaza, A., Plaza, J., Sanchez, S., Paz, A. (2009). Lossy Hyperspectral image compression on NVidia Graphics Processing Units, *Proc. SPIE Int. Soc. Opt. Eng.*, Volume 7455, (pp. 74550F).

Santos, L., Magli, E., Vitulli, R., Núñez, A., López, J., Sarmiento, R. (2013). Lossy hyperspectral image compression on a graphics processing unit: parallelization strategy and performance evaluation, *Journal of Applied Remote Sensing*, Vol. 7, (pp. 74599/1-15).

Reinartz, P., Lachaise, M., Schmeer, E., Krauss, T., Runge, H. (2006). Traffic monitoring with serial images from airborne cameras, *ISPRS Journal of Photogrammetry & Remote Sensing*.

Scherzinger, B., M. (2001). History of Inertial Navigation System in Suvey Applications, *Photogrammetric Engineering & Remote Sensing*, 67(11) (pp.1225-1227).

Schwarz, K., P. (1993). IntegratedAirborne Navigation System for Photogrammetry, *Photogrammetric Week'95*, Wichmann, Germany.

Schwarz, K., P., Chapman, M., E., Cannon, E., Gong, P. (1993). An Integrated INS/GPS Approach to the Georeferencing of Remotely Sensed Data, *Photogrammetric Engineering & Remote Sensing*, 59(11), (pp.1667-1674).

Skaloud, J. (2002). Direct Georeferencing in Aerial Photogrammetric Mapping, *Photogrammetric Engineering & Remote Sensing*, 68(3), (pp.207-210).

Şahin, H. & Külür, M., S. (2011). GPGPU Yöntemi İle Fotogrametrik Uygulamalar, *Türkiye Ulusal Fotogrametri ve Uzaktan Algılama Birliği VI. Teknik Sembozyumu TUFUAB,* Antalya, Türkiye; Şubat 21-25.

Şahin, H. & Külür, M., S. (2012). Orthorectification By Using GPGPU Method, *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume XXXIX-B4, Melbourne, Australia; August 25- September 01.

Thomas, U., Kurz, F., Rosenbaum, D., Mueller, R., Reinartz, P. (2008). GPU-based Orthorectification of Digital Airborne Camera Images in Real Time, *The Internatıonal Archives Of The Photogrammetry, Remote Sensing And Spatial Informatıon Sciences, ISPRS Congress*, Volume XXXVII Part B1 Commission I Beijing, China. 2008.

White, S. & Aslaksen, M. (2006). NOAA's Use of Direct Georeferencing to Support Emergency Response, *PERS Direct Georeferencing Column*.

Yastıklı, N., (2003). *GPS/IMU Verilerini Kullanarak Hava Fotoğraflarının Doğrudan Yöneltilmesi ve Birleştirilmiş Blok Dengeleme Olanakları*. (Doktora tezi). Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

Yılmaz, E., (2010). *Massive Crowd Simulation with Parallel Processing*. (Doktora tezi). Ortadoğu Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Ankara.

Zhang, B., (2015). GPU-accelerated 3-D point cloud generation from stereo images, *GPU Technology Conference*, California, USA: March 17-20.

ÖZGEÇMİŞ

Ad-Soyad **:** Hakan ŞAHİN

Doğum Tarihi ve Yeri **:** 04.11.1978, Samsun

E-posta **:** hsahin55@yahoo.com

ÖĞRENİM DURUMU:

- **Lisans** **:** 2000, Zonguldak Karaelmas Üniversitesi., Mühendislik Fakültesi, Jeodezi ve Fotogrametri Mühedisliği Bölümü
- **Yükseklisans** **:** 2003, Zonguldak Karaelmas Üniversitesi, Jeodezi ve Fotogrametri Mühendisliği Anabilim Dalı, Jeodezi ve Fotogrametri Mühendisliği Programı

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2001-2007 yılları arasında Zonguldak Karaelmas Üniversitesi Mühendislik Fakültesi Jeodezi ve Fotogrametri Mühendisliği Bölümünde Öğretim Görevlisi olarak çalıştı.
- 2007 yılından bugüne Harita Genel Komutanlığında Mühendis Subay olarak çalışmaya devam ediyor.

DOKTORA TEZİNDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

- Şahin, H., Külür, M., S., 2011. GPGPU Yöntemi İle Fotogrametrik Uygulamalar, *Türkiye Ulusal Fotogrametri ve Uzaktan Algılama Birliği VI. Teknik Sembozyumu TUFUAB,* 21-25 Şubat, Antalya, Türkiye.

- Şahin, H., Külür, M., S., 2012. Orthorectification By Using GPGPU Method, *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* Volume XXXIX-B4, August 25- September 01, Melbourne, Australia.

- Şahin, H., Külür, M., S., 2016. GPGPU Yöntemi İle Görüntülerin Gerçek Zamanlı Ortorektifikasyonu, *Harita Dergisi*, Sayı 155, 12-22.