T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

GRAFİK İŞLEMCİLER ÜZERİNDE ÇOKLU-ODAKLI GÖRÜNTÜLERİN PARALEL PROGRAMLAMA İLE BİRLEŞTİRİLMESİ

(Yüksek Lisans Tezi)

Hazırlayan
Zülkif KORKMAZ

Danışman
Yrd. Doç. Dr. Rifat KURBAN

Bu çalışma; Erciyes Üniversitesi Bilimsel Araştırma Projeleri Birimi tarafından FYL-2013-4798 kodlu proje ile desteklenmiştir.

Haziran 2015
KAYSERİ

BİLİMSEL ETİĞE UYGUNLUK

Bu çalışmadaki tüm bilgilerin, akademik ve etik kurallara uygun bir şekilde elde edildiğini beyan ederim. Aynı zamanda bu kural ve davranışların gerektirdiği gibi, bu çalışmanın özünde olmayan tüm materyal ve sonuçları tam olarak aktardığımı ve referans gösterdiğimi belirtirim.

Zülkif KORKMAZ

YÖNERGEYE UYGUNLUK

" Grafik İşlemciler Üzerinde Çoklu-Odaklı Görüntülerin Paralel Programlama ile Birleştirilmesi" adlı Yüksek Lisans tezi, Erciyes Üniversitesi Lisansüstü Tez Önerisi ve Tez Yazma Yönergesi'ne uygun olarak hazırlanmıştır.

Tezi Hazırlayan

Zülkif KORKMAZ

Danışman

Yrd. Doç. Dr. Rifat KURBAN

Bilgisayar Mühendisliği ABD Başkanı

Prof. Dr. Derviş KARABOĞA

Yrd. Doç. Dr. Rifat KURBAN danışmanlığında **Zülkif KORKMAZ** tarafından hazırlanan "**Grafik İşlemciler Üzerinde Çoklu-Odaklı Görüntülerin Paralel Programlama ile Birleştirilmesi**" adlı bu çalışma, jürimiz tarafından Erciyes Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında **Yüksek Lisans** tezi olarak kabul edilmiştir.

17/06/2015

JÜRİ:

Danışman : Yrd. Doç. Dr. Rifat KURBAN .....

Üye : Yrd. Doç. Dr. Celal ÖZTÜRK .....

Üye : Yrd. Doç. Dr. Bekir Hakan AKSEBZECİ B.H. Aksebzeci

ONAY:

Bu tezin kabulü Enstitü Yönetim Kurulunun 23/06/2015 tarih ve 2015/25-09 sayılı kararı ile onaylanmıştır.

23/06/2015

Prof. Dr. Kâzım KEŞLİOĞLU

Enstitü Müdürü

ÖNSÖZ / TEŞEKKÜR

Çalışmanın her adımında bilgi, tecrübe ve pozitif enerjisini esirgemeyen danışman hocam Sayın Yrd. Doç. Dr. Rifat KURBAN'a;

Maddi desteklerinden dolayı FYL-2013-4798 nolu proje ile çalışmayı destekleyen Erciyes Üniversitesi Bilimsel Araştırma Projeleri (BAP) Koordinasyon Birimi'ne;

Çalışmalarım süresince sabır gösteren, manevi desteklerini hep yanımda hissettiğim eşime, oğluma ve beni bu günlere getiren değerli aileme en içten teşekkürlerimi sunarım.

Zülkif KORKMAZ

Kayseri, 2015

GRAFİK İŞLEMCİLER ÜZERİNDE ÇOKLU-ODAKLI GÖRÜNTÜLERİN PARALEL PROGRAMLAMA İLE BİRLEŞTİRİLMESİ

Zülkif KORKMAZ

Erciyes Üniversitesi, Fen Bilimleri Enstitüsü

Yüksek Lisans Tezi, Haziran 2015

Danışman: Yrd.Doç. Dr. Rifat KURBAN

ÖZET

Görüntüler, farklı odaklamaya sahip kameralar tarafından elde edilirken belirli uzaklıktaki nesnelere odaklanılır. Odaklanılan kısımlar görüntünün net kısımlarıdır. Odak noktasının ilerisinde ve gerisinde olan bölgeler ise net olarak elde edilemez. Farklı odaklı kameralarla elde edilen birden fazla görüntülerden tek ve anlamlı bir görüntü elde edilebilir. Bu amaçla görüntü birleştirme teknikleri kullanılır.

Görüntü işleme teknikleri giderek karmaşıklaşırken, üzerinde işlem yapılan görüntülerin çözünürlüğü de artmaktadır. Bu aynı zamanda performans sorununu da meydana getirmektedir. Son yıllarda bu alandaki çalışmalar artmaktadır. Bunlardan birisi de grafik işlemciler üzerinde genel amaçlı programlama yapmaktır (GPGPU). GPU'lar, ekrana grafik çizilmesi işleminin doğası gereği, çok çekirdeklidir ve yüksek bir paralelliğe sahiptirler. Grafik yongaları, sabit fonksiyonlu grafik işlemcilerdir. Ancak giderek programlanabilir ve hesaplanabilirlik açısından güçlü hale gelmişlerdir. GPGPU programlamada, CUDA ve OpenCL çok kullanılan kütüphanelerdir.

Bu çalışmada, görüntü birleştirme tekniklerinden, dönüşüm uzayında çalışan, Laplacian Piramidi metodu kullanılarak GPU'lar üzerinde genel amaçlı programlama yapılmıştır. Karşılaştırmalarda kullanılmak üzere, CPU üzerinde seri çalışan bir program da geliştirilmiştir. Bu programlarla, GPU ve CPU üzerinde, Laplacian Piramidinin her adımında geçen süreler karşılaştırılmıştır. GPU'ların paralel işlem performansının yüksekliği açık bir şekilde gözlemlenmiştir. Özellikle büyük çözünürlüklü görüntülerde performans farkı daha fazla belirgin olmuştur.

Anahtar Kelimeler: Çoklu odaklı görüntü birleştirme, GPU'da genel amaçlı hesaplama (GPGPU), Laplacian piramit dönüşümü

MULTI-FOCUS IMAGE FUSION ON GRAPHICS PROCESSING UNIT WITH PARALLEL PROGRAMMING

Zülkif KORKMAZ

Erciyes University, Graduate School of Natural and Applied Sciences

M. Sc. Thesis, June 2015

Supervisor: Assist. Prof. Dr. Rifat KURBAN

ABSTRACT

While images can be attained by cameras featured to focus differently they focus on objects at specific distance. The focused part is the sharpest part of the image. The part around the focus becomes blurry. From multiple images obtained by differently focused cameras, single and meaningful image can be constructed. Fort this particular reason, image-fusion techniques are used.

Image processing techniques are getting more and more complex and resolution of images processed are being higher. Along with this, the problem of performance becomes important. The number of the studies in this issue has increased in the recent years. One of these studies is realizing a general purpose programming on graphics processor unit(GPGPU). GPUs, naturally, are multi-cored due to drawing graphics on the screen and have a high parallelism. GPU's chips have stable functions. However, they are more programmable and more powerful in terms of calculability. In GPGPU programs, CUDA and OpenCL libraries are used extensively.

In this study, by using the Laplacian Pyramid method, one of the image fusion methods, general purpose programming on GPU has been made. For comparisons, a serial program on single CPU has been developed. Through these programs, duration of every step of Laplacian Pyramid method on GPU and CPU is compared. It has been clearly observed that GPUs have a high parallel processing performance. Especially, the performance difference is observed to be a lot more in high resolution images.

Keywords: Multi-focus image fusion, general purpose computing on GPU (GPGPU), Laplacian pyramid transform

İÇİNDEKİLER

GRAFİK İŞLEMCİLER ÜZERİNDE ÇOKLU-ODAKLI GÖRÜNTÜLERİN PARALEL PROGRAMLAMA İLE BİRLEŞTİRİLMESİ

KISALTMA VE SİMGELER

LPD	Laplacian piramit dönüşümü yöntemi
LP	Laplacian piramit
GP	Gaussian piramit
GF	Gaussian filtre
ADD	Ayrık dalgacık dönüşümü yöntemi
GPU	Grafik işlemci birimi (graphics processing unit)
GPGPU	Grafik işlemciler üzerinde genel amaçlı hesaplama yapma (general purpose computing on GPU)
GFLOP	İşlemcinin saniyedeki kayan nokta işlem sayısı milyar cinsinden değeri
CPU	Merkezi işlem birimi (central process unit)
SSIM	Yapısal benzerlik (structural similarity)
SS	Standart sapma
SN	Saniye

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

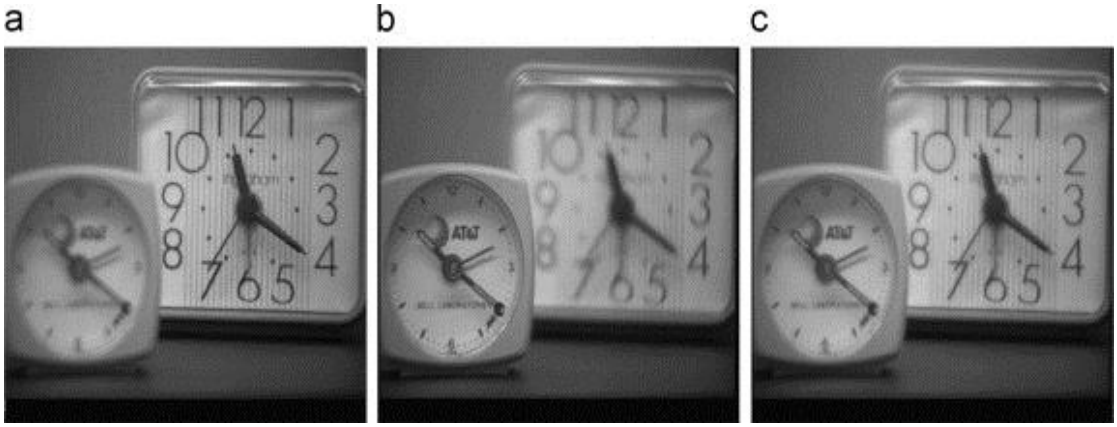
GİRİŞ

Görüntü birleştirme genel olarak, farklı birden fazla resmin birleştirilerek yeni bir resim elde edilmesidir. Sık kullanılan durumlarda ise, aynı çevrenin, lens ayarlamalarıyla farklı odaklanmış makinelerle, farklı odaklanmasından veya farklı sensörlerle, birden fazla görüntüsünün elde edilmesi sonucunda ortaya çıkan çoklu odaklı görüntülerin birleştirilmesidir [1]. Birleştirilecek resimlerden anlamlı kısımlar alınarak çeşitli birleştirme yöntemleriyle daha anlamlı ve gelişmiş yeni görüntüler elde edilir. Görüntü birleştirme, medikal görüntüleme [2], güvenlik, üretim işlemlerinin izlenmesi, uydu görüntüleme [3] vb. alanlarda kullanılmaktadır.

Tezin Motivasyonu

Çoklu odaklı görüntü birleştirme, görüntü birleştirmede en önemli alanlardan biridir. Çünkü limitli alan derinliğiyle odaklanabilen optik lenslerle çeşitli mesafelerdeki nesneleri net bir şekilde almak zordur. Çoklu odaklı görüntülerde, görüntünün bir kısmı net iken odaklanmamış olan kısmı ise bulanıktır. Fakat farklı görüntüleri farklı odaklarla almak kolaydır. Çoklu odaklı görüntülerde görüntü birleştirmenin amacı bu farklı odaklanmış, farklı nesneleri net olarak gösteren resimleri birleştirerek yeni ve resmin tamamının odaklanıldığı, net görüntüler elde edebilmektir [4].

Şekil 1.1'de farklı odaklanmadan kaynaklı olarak aynı çevreden farklı görüntüler elde edilmiştir. (a) resminde arkadaki saate odaklanılmıştır. Netlik aralığı içinde kalan arkadaki saat net çıkmış, netlik aralığı dışında kalan öndeki saat bulanık çıkmıştır. (b) resminde ise yine aynı çevrede alınan görüntüde bu sefer ön tarafa odaklanılmış ve ön taraf net çıkmıştır, odaklanılmayan arka taraf ise bulanık çıkmıştır, (c) resmi ise bu iki resmin birleştirilmiş hali olup, resmin tamamında net görüntü elde edilmiştir.



Şekil 1.1 Farklı odaklanmış görüntüler [4].

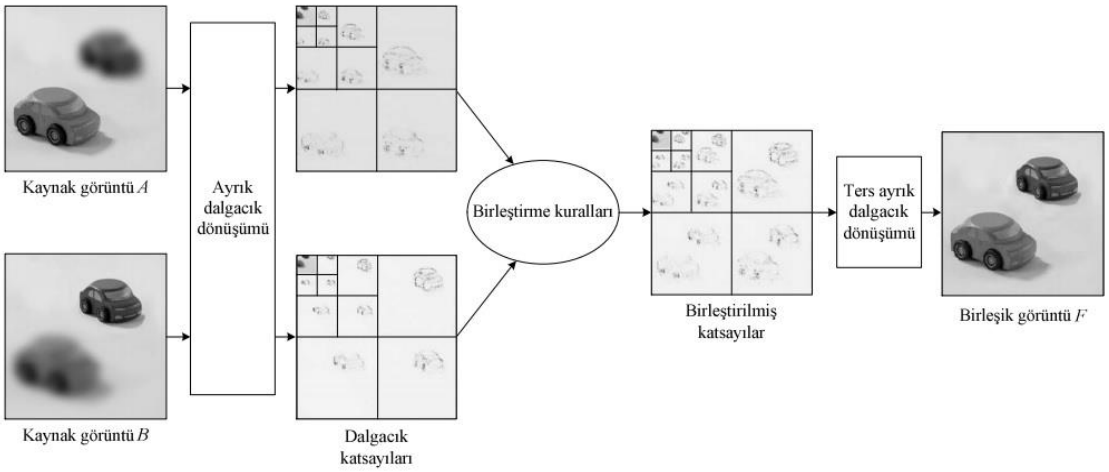
Net resimlerin elde edilmesinde farklı yöntemler vardır. Bu yöntemler her geçen gün geliştirilmekte ve başarı oranı artmaktadır. Fakat görüntü işlemede kullanılan görüntülerin kalitesi artarken, görüntü işleme teknikleri de her geçen gün daha karmaşıklaşmaktadır. Bu ise, görüntü işlemede performans artışları için yeni yöntemlerin araştırılmasını zorunlu kılmaktadır [5,6].

Gerçek zamanlı görüntü işleme genellikle büyük hacimli ve karmaşık algoritmalardan oluşmaktadır. Basit olarak işlem hacmine göre görüntü işleme üç kısma ayrılabilir. Birincisi, alt seviye, sadece piksel operasyonlarına dayanan işlemlerdir. Orta seviyede; hareket çıkarma, resim ayırma, özellik çıkarma veya eşleştirme gibi işlemlerdir. En üst performans seviyesinde ise genellikle yapay zekâ algoritmaları vb. dayalı yorumlama gerektiren, önceki çevre bilgilerine ihtiyaç duyulan alanlardır [7].

Literatür incelemesi

Görüntü birleştirme temel olarak iki ana yöntemle yapılmaktadır. Birincisi resim uzayında birleştirilir. En temel yöntem iki resmin piksel piksel birleştirilmesidir. Bu yöntemde genel kontrast azalır. Diğer bir resim uzayında görüntü birleştirme yöntemi ise resim uzayında blok seçme yapılarak birleştirilir. Bu yöntemde bulanık kısımlar ile net kısımlar bloklar halinde karşılaştırılır ve net kısımlar sonuç resmi oluşturur [8]. İkincisi; resim, dönüşüm uzayında birleştirilir. Bu yöntemlerden ikisi Laplacian Piramit dönüşümü(LPD) ve ayrık dalgacık dönüşümüdür(ADD) [9]. Bu yöntemlerde görüntüler çoklu çözünürlüklü olarak ayrıştırılır, sonrasında bu çözünürlük seviyelerinde belirli

kurallar çerçevesinde birleştirilme işlemi yapılır. Son olarak bu çoklu çözünürlükler tekrar ters dönüşümle birleştirilerek sonuç resim elde edilir [8]. Şekil 1.2'de ADD şeması gösterilmiştir.



Şekil 1.2 Ayrık Dalgacık Dönüşümü [8].

Laplacian Piramidi(LP) yönteminde resim öncelikle filtreden geçirilir ve çözünürlüğü düşürülür. Bu şekilde birkaç adımda yapılan çözünürlüğün düşürülmesi işleminin sonucunda Gaussian piramidi oluşturulur. Gaussian piramitlerindeki her bir görüntüden Laplacian piramit seviyeleri oluşturulur. Bu farklı çözünürlüklerde her iki resim için seçim kuralları ile seçim yapılır. Oluşan bu yeni LP'ne uygulanan ters dönüşüm ile yeni birleştirilmiş görüntü elde edilir [10].

Görüntü birleştirme yöntemleri geliştirilmeye devam etmektedir. Bu yöntemlerde yeni karmaşık algoritmalar sunulmaktadır. Gelişen yeni teknolojiler ve ihtiyaçlarla birlikte resimlerin çözünürlükleri de artmaktadır. Bu ise performans problemini de beraberinde getirmektedir. Genel olarak resim işlemedeki performans problemi olduğu gibi, görüntü birleştirmede de bu durum üzerinde durulması gereken önemli bir alan teşkil etmektedir. Bu konuda da farklı çalışmalar yapılmaktadır [7,11-13].Bu çalışmalara göre yüksek başarımlı işleme gücüne sahip gelişmiş makineler oluşturulmaktadır. Birden fazla işlem birimleri veya bilgisayarlar paralel olarak birleştirilerek güçlü sistemler oluşturulmuştur. Bu sistemlerden biri olan Warp Bilgisayarı çok fazla sayıda işlemci birimini bir araya getirmiştir 1980-1990 yılları arasında yapılmıştır [11].

Paralel sistem mimarisine yönelik çalışmalar son yıllarda gitgide artmaktadır. Paylaşımlı elemanlar ile modüler ve ölçeklenebilir, fakat görüntü işlemede karmaşık ve tahmin edilebilirliği az yapılardır. Dağıtık sistemler bazıları DSP (Texas C4X, ADI SHARC) tarafından üretilmektedir. Doğrudan noktadan noktaya iletişim ile gerçek zamanlı görüntü işlemede kullanım alanı sunmaktadır. Yeni nesil yüksek başarımlı görüntü işleme işlemcileri, TMS320C6X (Texas Instruments) ve TigerSHARC (Analog Devices, Inc.) ailesi gibi, C4X ya da SHARC cihazlarına göre daha yüksek kapasiteli performans göstermektedirler. Bazı görece basit paralel sistemler daha az DSP ile yüksek performans almak üzere inşa edilebilir. Ayrıca paralel algoritmaların karmaşıklığı azaltılabilir. Böylece makineler arasındaki iletişimden kaynaklanan performans kaybı gözle görülür oranda azaltılabilir [7,12].

Diğer yöntem, var olan görüntü işleme algoritmalarının karmaşıklığını azaltmak veya yeni karmaşıklığı az algoritmalar geliştirmektir. Bu ise zaten iyice karmaşıklaşan hesaplamalardan dolayı nispeten faydasını gösterememe durumuyla karşı karşıyadır.

Bir diğer yöntemde grafik işlemcilerin paralel hesaplama gücünden faydalanmaktır. Bu yöntemde grafik işlemciler, gün geçtikçe programlanabilir hale geldiklerinden hem daha masrafsız hem de kolay erişilebilir hale gelmiştir [13].

Tezin amacı ve katkıları

Çoklu odaklı görüntülerin birleştirilmesinde performans iyileştirmesi için çalışmalar yapılmıştır. Üzerinde çok fazla çalışma yapılmamış bir alan olan Laplacian piramidi ile görüntü birleştirmesi üzerine çalışılmıştır.

Performans iyileştirmesi için grafik işlemcileri üzerinde paralel programlama teknikleri (GPGPU) kullanılmıştır. OpenCL ve CUDA programlama dilleriyle ayrı ayrı yapılan Laplacian piramit yöntemiyle görüntü birleştirme yazılımı yapılmıştır. GPU üzerinde paralel çalışması sağlanmıştır. Ayrıca CPU'ya özel geliştirilen Laplacian piramit ile paralel olmayan program da yine yapılmıştır.

CPU ve GPU üzerinde ayrı ayrı geliştirilen Laplacian Pramit ile görüntü birleştirme yöntemlerinin performans karşılaştırması yapılmıştır. OpenCL için ve Nvidia ve ATI

grafik kartları için testler yapılmıştır. CPU'lar üzerinde de yine seri programlar test edilmiştir.

Tezin Organizasyonu

Birinci bölümde, genel olarak görüntü birleştirme teknikleri, özelde ise Laplacian Piramit ile görüntü birleştirme anlatılmıştır.

İkinci bölümde, grafik işlemcilerde paralel programlama anlatılmıştır. Genel olarak GPGPU programla üzerinde durulmuştur. CUDA ve OpenCL paralel programlama kütüphaneleri ile Arrayfire kütüphanesi tanıtılmıştır.

Üçüncü bölümde, deney düzeneği ve deneydeki veriler tanıtılmış olup, çoklu odaklı görüntülerin birleştirilmesi için LP ile GPGPU programlama ile geliştirilen programımızın çıktıları incelenmiştir. Nvidia ve AMD ATI grafik işlemciler ile AMD ve Intel işlemciler üzerinde paralel programlanan kodlarımız çalıştırılmıştır. Ayrıca CPU için seri olarak geliştirilen C++ programı da testlerdeki performans karşılaştırmalarında kullanılmıştır.

Dördüncü bölümde, tez kapsamındaki test sonuçları yorumlanmış ve öneriler sunulmuştur.

1 BÖLÜM

GÖRÜNTÜ BİRLEŞTİRME TEKNİKLERİ

1.1 Giriş

Bu bölümde, çoklu odaklı görüntülerin birleştirilmesinde kullanılan yöntemlerden Laplacian Piramit Dönüşümü anlatılmıştır.

1.2 Laplacian dönüşümü ile görüntü birleştirmesi

1.2.1 Laplacian Piramit

Görüntü piramitleri, bir görüntüye alçak veya bant geçiren bir filtre uygulayarak kademe kademe çözünürlüğün düşürülmesiyle elde edilir. Laplacian piramidinde de yine aynı şekilde çok ölçekli çözünürlükler oluşturulur. Her çözünürlük katmanı kendinden önceki katmana filtre uygulayıp, çözünürlük düşürülerek, arasındaki fark alınarak elde edilir [14].

Öncelikle filtreden geçirilen görüntülerin çözünürlükleri düşülerek 1. piramit seviyesi oluşturulur. 0. piramit seviyesi ise orijinal görüntüdür. Sonrasında bu işlem laplacian piramit seviyesi kadar(genelde ideal seviye dörttür.) işleme devam edilir. Böylece ortaya gaussian piramit çıkar. Sonrasında tekrar çözünürlükler yükseltilir ve filtreden geçirilir. Bu haliyle önceki seviyeden çıkarılır ve Laplacian piramidinin adımı oluşmuş olur. Bu işlem piramidin her seviyesi için tekrarlanır [16-18].

1.2.2 Gaussian Piramit

Laplacian piramidinin ilk adımı Gaussian piramididir. Orijinal görüntü G0 alçak geçiren filtreden geçirilerek G1 gaussian görüntüsü elde edilir. G1 orijinal görüntü G0'ın "İndirgenmiş" hali olarak adlandırılır. G1, G0'in hem çözünürlüğünün hem de yoğunluğunun düşürülmüş halidir. Aynı yolla G2, G1'in filtrelenmiş ve çözünürlüğünün

düşülmüş halidir. Bu şekilde son seviyeye kadar gidilir. Filtreleme simetrik ağırlıkları olan bir gaussian filtresi ile resimleri konvolüsyona tabi tutmaktır. Bu sıralanmış G0, G1, G2…GN resim koleksiyonuna Gaussian Piramit denir [10,15].

1.2.3 İndirgeme İşlemi

Gaussian pramidi oluştururken ilk başta C piksel kolon R piksel satıra sahip bir resmimiz olduğu varsayalım. G0 başlangıçtaki orijinal resmimizdir. Piramidin birinci seviyesi G0'in alçak geçiren filtreyle indirgenmiş hali olan G1'dir. Birinci seviyedeki her değer 0. seviyedeki değerin ağırlıklı ortalamasının filtre ile (genellikle 5×5'lik bir pencere) hesaplanmasıyla oluşur. Seviye iki de aynı şekilde oluşturulur. Pencere boyutunun 5×5 olması kesin gerekmemekle beraber, performans ve kalite açısından örüntü olmuştur. Seviye-seviye ortalama alarak indirgeme işlemi "İndirge" fonksiyonu olarak gösterilmiştir.

$$g_k = \text{İndirge}(g_{k-1}) \tag{1.1}$$

Her seviye için $0 < l < N$ ve i, j; $0 < i < C_l$, $0<j<R_l$,

$$g_l(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) g_{l-1}(2i+m, 2j+n) \tag{1.2}$$

Burada N piramidin seviyesini gösterir. C ve R ise birinci seviyedeki görüntünün boyutlarıdır. C_l, ve R_l ise l. seviyedeki piramit görüntüsün boyutlarıdır. W(m, n) filtrelemede kullanılacak pencerenin boyutlarıdır. (i, j) noktası için bu işlem uygulanmaktadır. Şekil 1.1'de de gösterildiği üzere dört tane iki boyutlu çerçeveden bir hücre oluşturulur [10].

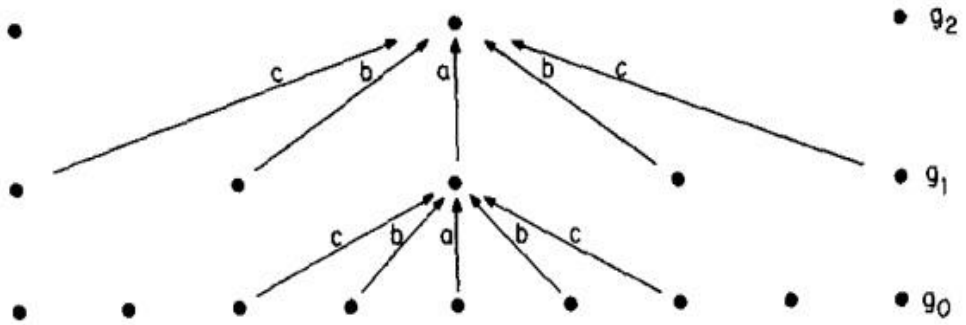
Filtre oluşturma

5×5 lik filtre w(5×5) her piramit seviyesinde kullanılmaktadır. Bu değerler örüntü olarak kabul edilmiştir.

$$w(m,n) = w\hat{}(m).w\hat{}(n) \tag{1.3}$$

Burada bir boyutlu ve uzunluğu 5 olan wˆ fonksiyonu normalize edilmiş değerleri toplamı 1'dir. Ayrıca bu kernel aşağıda ifade edildiği gibi simetrik olmalıdır [10].

$$\hat{w}(i) = \hat{w}(-i),\ i = 0,1,2 \text{ için} \tag{1.4}$$



Şekil 1.1 Gaussian Piramit İndirgeme İşlemi

Şekil 1.2'de örnek bir Gaussian piramit görüntüsü verilmiştir.

Şekil 1.2 Gaussian Piramidi

1.2.4 Genişletme İşlemi

Genişletme (Expand), indirgeme işleminin tersidir. (N,M) değerli bir resim genişletmek işlemiyle (2×M,2×N) boyutlarına getirilmektedir. "l" gaussian piramidi seviyesi olmak üzere, G(*l*) seviyesindeki gaussian görüntüsü g(*l-1*) seviyesinde görüntüyle aynı çözünürlüğe getirilir.

$$G_{l,n} = \text{Genişlet}(G_{l,n-1}) \tag{1.5}$$

Genişletmenin anlamı, 0< l <N ve 0<n seviyeleri;

i ve j ise piksel elemanlarını, (0≤ $i \leq$ C (l-n) , 0≤ $j \leq$ R(l-n))olmak üzere;

$$G_{l,n}(i,j) = 4\sum_{m=-2}^{2}\sum_{n=-2}^{2} w(m,n)G_{l,n-1}\left(\frac{i-m}{2}, \frac{j-n}{2}\right) \tag{1.6}$$

Eğer genişletme işlemini seviye sayısı kadar yaparsak en baştaki görüntünün çözünürlüklerine ulaşmış oluruz.

Laplacian piramidinin seviyeleri, Gaussian piramidinin seviyeleri arasındaki genişletme işlemleri ile aynı seviyedeki resim arasındaki hatalardan oluşur. L0,L1,L2…LN seviyesindeki resimler, Gaussian piramidindeki G0,G1..GN seviyeleri arasındaki sıralı olarak oluşan hatalar zinciridir.

0 ≤ l ≤ N olmak üzere;

$$L_l = G_l - Genişlet(G_{l+1}) \tag{1.7}$$

G(N+1) görüntüsü olmadığı için L(N) = G(N) olarak alınır.

1.2.5 Birleştirme

Oluşturulan Laplacian Piramit'in her bir seviyesindeki görüntüler, genişletme işleminden sonra Gaussian filtreden geçirilirler. Bir önceki seviyedeki piramit seviyesi ile aynı çözünürlükte olan yeni bir görüntü elimizde artık mevcuttur. Sonraki adımda bu iki görüntü birleştirilir.

L(N) =G(N) ve l= N-1,l=N-2…l=0 için;

$$L_l = G_l - \text{Genişlet}(G_{l+1}) \tag{1.9}$$

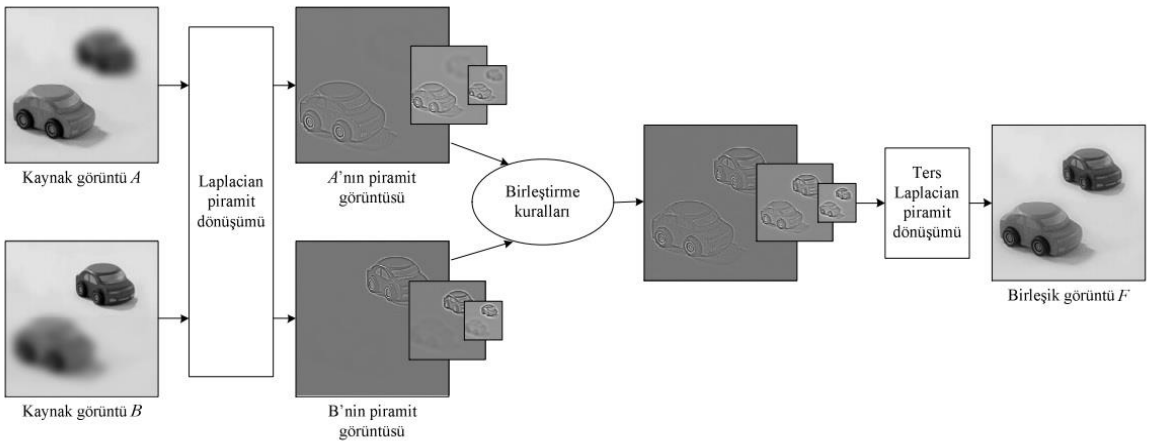
Birleştirme işlemleri için çeşitli yöntemler vardır. En basiti ve kullanışlı yöntemlerden biriside piksellerin değerlerinin mutlak değeri alınarak hangisi daha büyükse o alınır [18-19].

$$F_i(x,y) = \begin{cases} A_i(x,y), |A_i(x,y)| > |B_i(x,y)| \; ise \\ B_i(x,y), diğer \end{cases} \quad (1.10)$$

Aşağıdaki denklemden de görüleceği üzere ters dönüşümle orijinal görüntüye kadar ulaşılabilir [10].

$$g_0 = \sum_{i=0}^{N} L_i \quad (1.8)$$

Şekil 1.3'te Laplacian piramidinin blok diyagramı gösterilmiştir [8].



Şekil 1.3 Laplacian Piramit ile Görüntü Birleştirme.

2 BÖLÜM

GRAFİK İŞLEMCİLERDE HESAPLAMA

2.1 Giriş

Bu bölümde, önce genel olarak seri ve paralel programlamadan bahsedilmiştir. Grafik işlemcilerde paralel programlama üzerine durulmuş olup, genel amaçlı grafik işlemcilerde programlama(GPGPU) anlatılmıştır. Sonrasında paralel programlama platformları olan CUDA, OpenCL, DirectCompute ve ArrayFire anlatılmıştır.

2.2 Seri ve Paralel Programlama

Genel tanımıyla paralel programlama bir işlemi gerçekleştirmek için birden fazla kaynağı eş zamanlı olarak kullanmaktır. Problemler ufak işlere bölünerek koordineli bir şekilde bu görevler işlenir [40].

Seri (1 işlemci)

1 + 2 + 3 + 4 + 5 + 6

3 + 3 + 4 + 5 + 6

6 + 4 + 5 + 6

10 + 5 + 6

15 + 6

21

Paralel (2 işlemci)

1 + 2 + 3 + 4 + 5 + 6

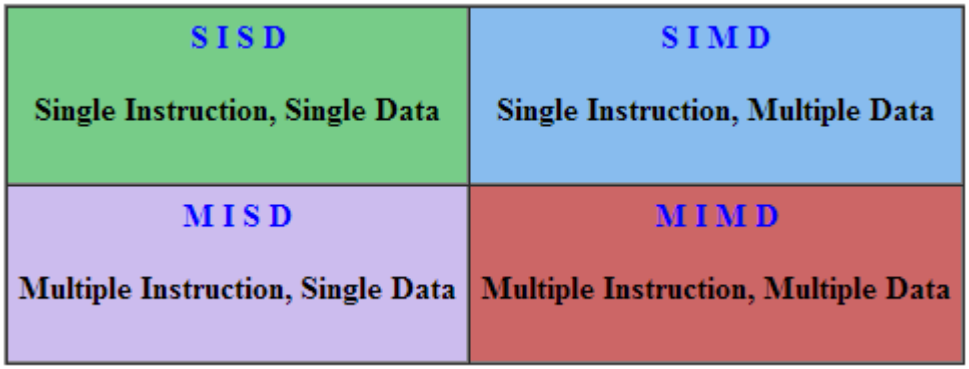
3 + 3 + 4 + 11

6 + 15

21

5 işlemden 3 işleme düşmüştür, yaklaşık 1,66 kat hızlanma sağlanır. Daha fazla işlem olursa hızlanma 2 kata kadar çıkar.

Flynn's Classical Taxonomy'ye gore Paralel programlamada dört önemli yapı bulunmaktadır [40].



Şekil 2.1 Paralel programlama yapıları

Single Instruction, Single Data (SISD): Tek kod, tek data, tek çekirdekli makine şeklinde de düşünebiliriz.

Single Instruction, Multiple Data (SIMD): Tek Direktif, Çok Data. Bir işlemi büyük veri kümelerine uygulamada kullanılabilir.

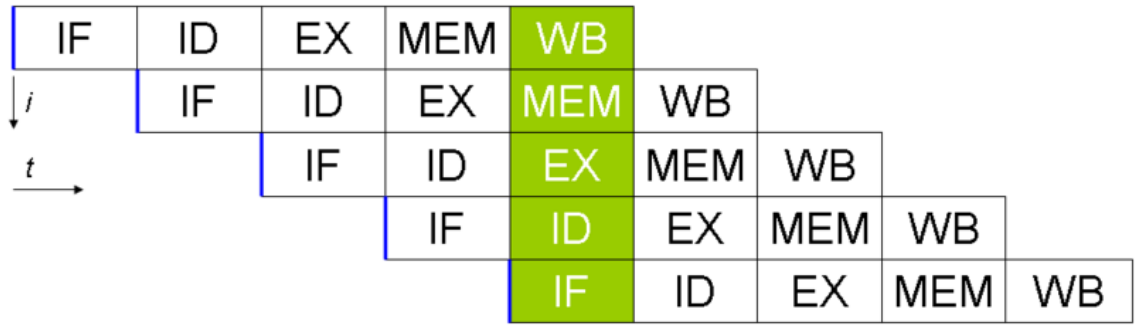
Multiple Instruction, Single Data (MISD): Çok kullanılan bir yapı değildir.

Multiple Instruction, Multiple Data (MIMD): Bu durumda hem datalar hem kodlar bölünmüş bir yapıdadır [21].

Günümüz hesaplama sistemlerinde paralelliği 4 ana model ile ele alabiliriz. Bunlar bit seviyesi(bit-level), komut seviyesi(instruction-level), veri paralelliği(data parallelism) ve görev paralelliğidir (task parallelism) [21].

Bit seviyesinde paralellik: işlemcinin tek bir komutunda işleyebileceği veri boyutunu ilgilendirir. En temel örnek ile açıklayacak olursak 8-bit bir işlemci 16-bit tam sayı toplamını 2 aşamada 2 ayrı 8-bitlik toplama şeklinde yaparken 16-bit bir işlemci tek aşamada yapacaktır.

Komut seviyesinde paralellik: Boru hattı bir işlemi birbirine bağlı farklı evrelere ayırıp farklı yönergelerin farklı evrelerini paralel olarak işleyebilmesine denebilir.



Şekil 2.2 Komut seviyesinde paralellik [40]

RISC mimarisindeki beş seviye (IF : Instruction Fetch, ID : Instruction Decode, EX : Execute, MEM : Memory access, WB : Register write back)

i : yönerge(instruction) sayısı, t : çevrim(cycle) sayısı

IF, ID, EX, MEM, WB bir yönergenin farklı ve sırasıyla birbirine bağımlı evreleri

Veri seviyesinde paralellik: Aynı görevlerin farklı veriyle paralel olarak işletilmesinden ileri gelir. Bunlara örnek olarak genelde büyük veriler üzerinde fark gözetmeksizin aynı işlemin yapıldığı ya da aynı kod satırının çalıştırıldığı grafik işlem birimlerini verebiliriz.

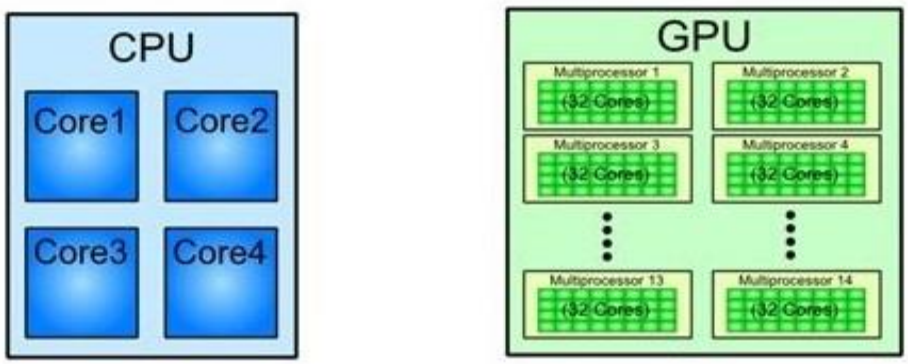
Görev paralelliği: Birden fazla işlem birimi içeren sistemlerde farklı iş parçacıklarının farklı ya da aynı veriyle eş zamanlı işletilmesine denir.

Ayrıca Paylaşımlı hafıza ve ayrık hafıza olarak da ayrımlar yapılabilir.

2.3 GPU'da Paralel Programlama(GPGPU)

GPU'lar ekrana grafik çizilmesi işleminin doğası gereği (ekran üzerindeki piksellerin birbirlerini koşullandırmaması), çok çekirdeklidir ve yüksek bir paralelliğe sahiptirler. Grafik yongaları, sabit fonksiyonlu grafik işlemcilerdir. Şekil 2.3'te GPU ve CPU çekirdek sayıları karşılaştırılmıştır.

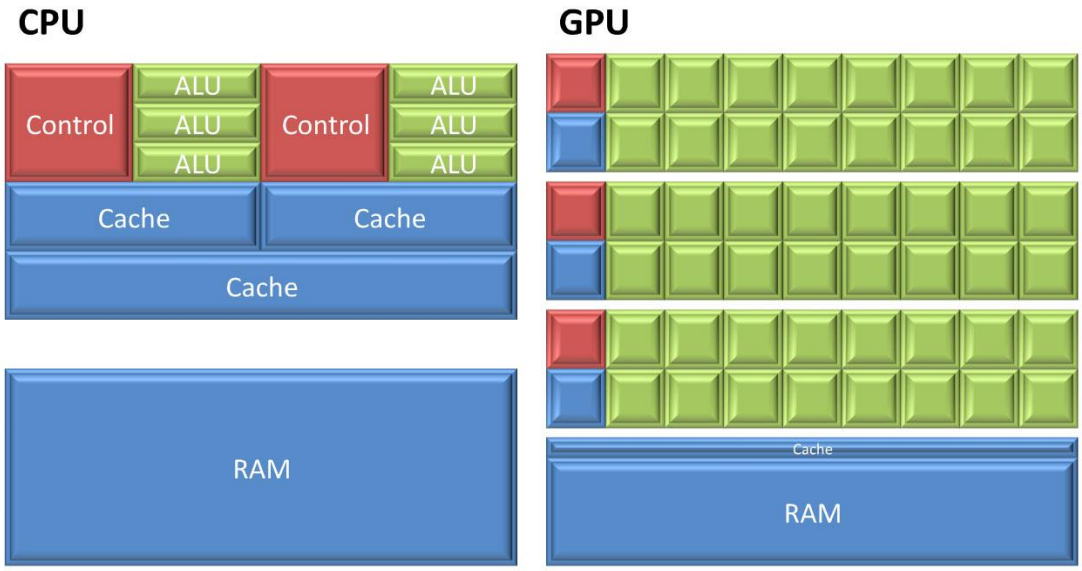
Ancak giderek programlanabilir ve hesaplanabilirlik açısından güçlü hale gelmişlerdir. Her geçen gün performansları artmaktadır. Çeşitli alanlarda GPU'lar, birtakım uygulamaları hızlandırmak için kullanılmaya başlanmıştır [22-24].



Şekil 2.3 CPU/GPU Mimarisi çekirdek sayılarının karşılaştırması [31]

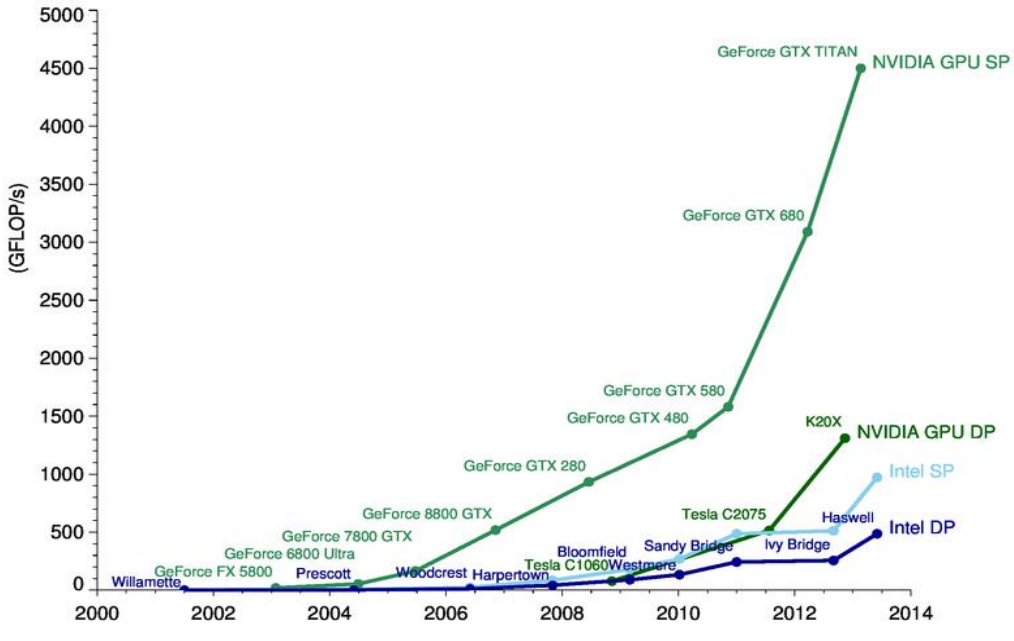
GPGPU (General Purpose Computing on Graphical Processing Unit) normalde CPU üzerinde yapılan işlemlerin, normalde grafik işleme yapan GPU'lar üzerinde yapılmasıdır. Çok çekirdekli sistemlerle benzer düşünülse de ayrı bir yapı olarak genel amaçlı GPU programlamayı ele alabiliriz [25].

Grafik işlemcisi üzerinde hesaplama yapılması; paralel doğadaki algoritmalarda çok ciddi performans artışları sağlayabilmektedir. Artan performans haricinde watt başına düşen işlem gücü de GPU'ların işlem birimi olarak kullanılmasında etkendir. Bunun temel nedenlerinden biri GPU mimarisinin benzer zar alanı ve benzer üretim standardında çip üzerinde daha fazla hesaplama birimi ve çekirdek içermesidir. Şekil 2.4'te GPU ve CPU mimari yapısı gösterilmiştir.



Şekil 2.4 CPU/GPU mimari karşılaştırması [31]

Buna nazaran merkezi işlem birimlerinde uygulamadaki performans artışını sağlamak için çip üzerindeki alanın büyük bir kısmı önbellek hafıza birimleri için kullanılır.



Şekil 2.5 CPU/GPU saniye bazında işlem performans karşılaştırması [41]

Heterojen Programlama

CPU ve GPU güçlü bir kombinasyondur; çünkü CPU'lar seri işleme için optimize edilmiş birkaç çekirdekten oluşur. Oysa GPU'lar, paralel performans için tasarlanmış daha küçük ve daha etkili binlerce çekirdekten meydana gelir. Kodun seri kısımları CPU üzerinde çalışırken, paralel kısımlar GPU üzerinde çalışabilir [26].

GPU'lar hem donanım performansı olarak hem de diğer CPU vb. Bileşenlerle uyum ve geliştirme ara yüzleri olarak çok hızlı bir şekilde gelişti ve gelişmeye devam etmektedir. Gelecekte CPU + GPU hibrid sistemlerin hesaplama mimarilerinde çok önemli yer tutacağı tahmin edilmektedir.

2.3.1 GPU Programlama Kullanım Alanları

- Bilim ve mühendislik alanlarında: Biyoteknoloji, genetik, kimya, moleküler bilim, mekanik mühendislik, bilgisayar bilimi ve matematik.

- Endüstriyel ve ticari alanlarda: Veritabanı, veri madenciliği, web arama motoru, web tabanlı iş servisleri, finansal ve ekonomik model.

- HPC: Akışkanlar dinamiği hesaplamalarında, sismik görüntülemede, DNA, protein modellemede, iklim simülasyonlarında.

- Multimedia: 3D işlemlerinde, video, resim işlemede [27-28].

2.3.2 Olası problemler

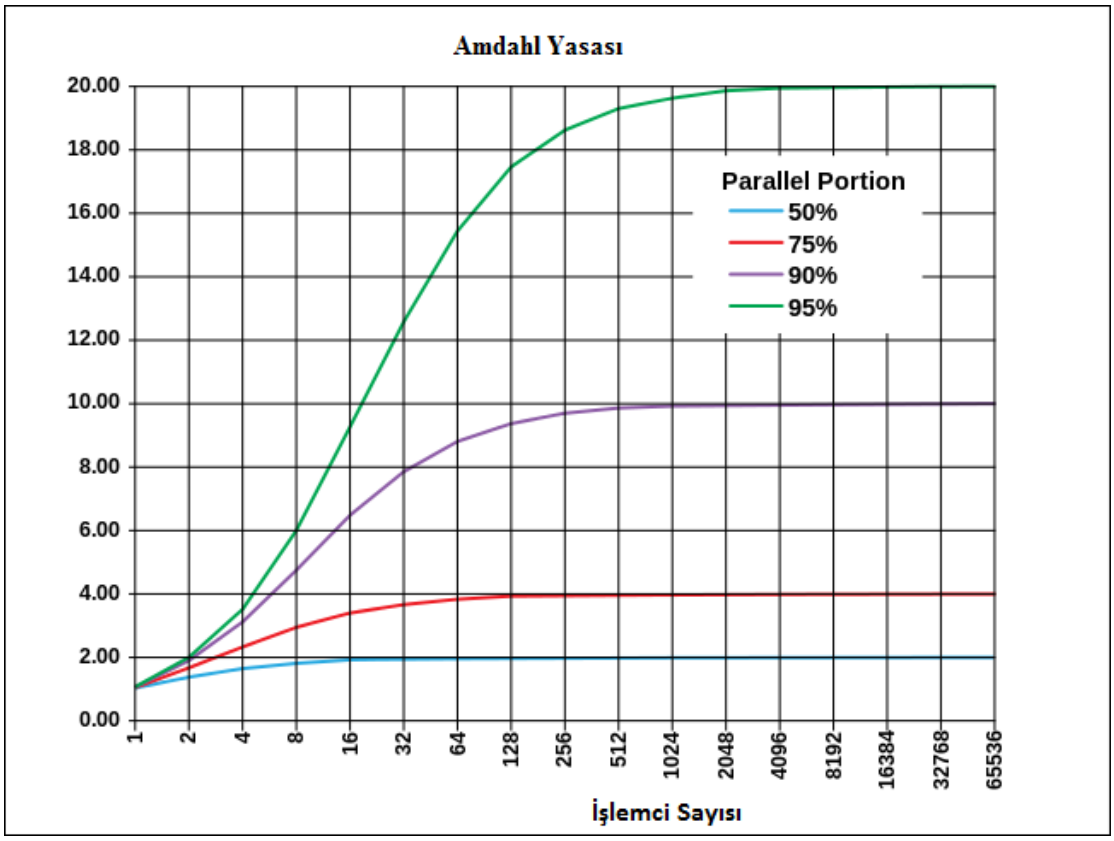
Pratikte işlemci sayısıyla orantılı hızlanmayı başarmak çok zordur. Bunun nedeni, Amdahl yasasında da belirtildiği üzere doğada birçok algoritma aslında sıralıdır.

Ekstra işlemciler eklendikçe, bazı iş yükleri, boru hattı (pipeline) paralellik kullanarak belli bir noktaya kadar fayda sağlar. Eğer bir insan bir çukuru bir dakikada kazıyorsa, 60 insanın bir çukuru bir saniyede kazması gerekir.

Pek çok algoritma, paralel donanımın kullanımını daha verimli yapmak için tekrardan tasarlanmalıdır.

Tek işlemcili sistemlerde iyi çalışan programlar, paralel sistemlerde aynı performansı vermeyebilir. Aynı programın çoklu kopyaları, birbirlerini etkileyebilirler (aynı anda aynı hafıza adresine yazma/okuma yapma). Bu yüzden paralel sistemlerde dikkatli programlama yapılması gerekmektedir.

Ayrıca program geliştirme ve hata ayıklama da daha zordur. Hangi işleme hangi algoritma çözüm olur çok iyi hesaplanması gerekmektedir [29,40].



Şekil 2.6 Amdahl Yasası [40].

Programın paralelleştirmeden kaynaklı performansı programın ne kadar paralelleştirildiğiyle ters orantılı olarak düşer. Program yüksek oranında paralelleştirildiğinde kaç tane işlemcinin kullanıldığına bakılmaksızın performans artışında bir yavaşlama başlar. İlk başlardaki üst düzey seviyeleri yakalayamayabiliriz [30].

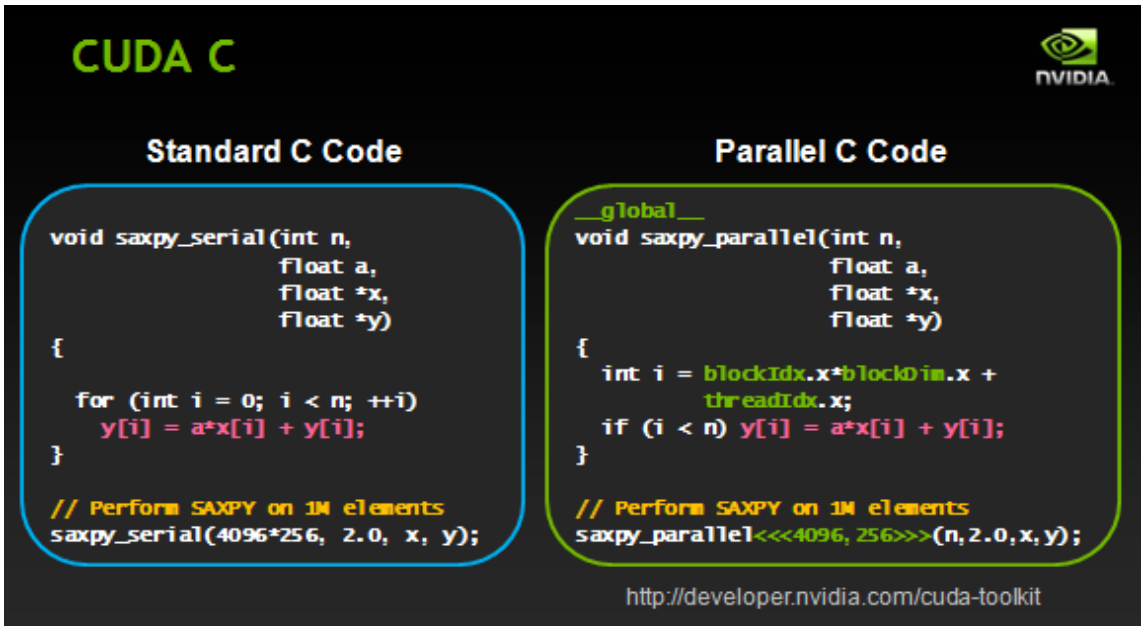
2.3.3 Nvidia CUDA

CUDA (Compute Unified Device Architecture – Birleşik Hesap Cihazı Mimarisi) Nvidia firması tarafından 2006 yılında Nvidia GPU'larda genel amaçlı hesaplama yapılmasına olanak sağlamak üzere tasarlanmış bir mimaridir.

Linux, Windows ve Mac OSX üzerinde çalışabilen hem düşük seviyeli hem de yüksek seviyeli birer yazılım geliştirme ara yüzü (API) Nvidia tarafından sunulmaktadır.

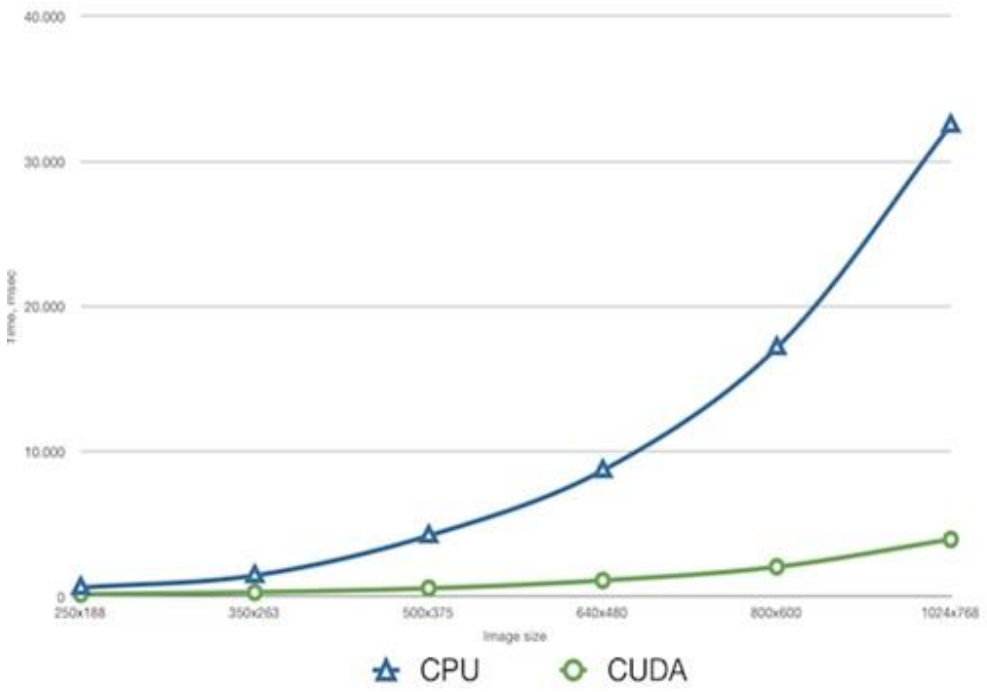
CUDA sadece Nvidia GPU'larda çalışması itibariyle rakiplerinden farklı olsa da, dünya üzerinde 300 milyondan fazla CUDA destekli GPU olduğu bilinmektedir.

CUDA G8X üzeri, GeForce, Quadro ve Tesla'yı içeren her GPU da çalışır. Nvidia, ekran kartı mimarilerinin ileriye doğru kod uyumluluğu sayesinde, Geforce 8 için geliştirilen programların herhangi bir düzeltme yapılmadan gelecek nesil ekran kartlarında hızlanmalardan otomatik olarak faydalanacak şekilde kullanılabileceğini ifade etmektedir. CUDA kütüphanesi, geliştiricilerin CUDA özellikli GPU'lar üzerindeki hafızalara ve işlemcilere hükmedebilmesini sağlar. İlk CUDA Geliştirici seti (SDK) 15 Şubat 2007 de yayınlandı [31]. Şekil 2.7'de CUDA kod örneği verilmiştir.



Şekil 2.7 CUDA kod örneği [31]

Şekil 2.8'de ise iris tanımada CPU ve CUDA arasındaki performanslar karşılaştırılmıştır.



Şekil 2.8 Iris tanıma CPU/GPU performans karşılaştırması [31]

2.3.4 OpenCL

OpenCL(Open Computing Language), Apple tarafından 2008 yılında kar amacı gütmeyen teknoloji şirketleri birliği Khronos Group'a önerilen, kabul gördükten sonra spesifikasyonu pek çok şirketin katkılarıyla hazırlanan heterojen hesaplama platformudur. OpenCL; destekli grafik işlemcileri, genel amaçlı işlemciler gibi farklı platformlarda hesaplama yapılmasına olanak sağlar. OpenCL AMD, Intel, Nvidia ve ARM tarafından desteklenmektedir [32].

2.3.5 Direct Compute

DirectCompute Microsoft tarafından oluşturulan, DirectX 10 ve 11 de desteklenen bir GPGPU geliştirme arayüzüdür. 2015 Mayıs ayı itibariyle yalnızca Windows Vista, Windows 7 ve Windows 8 işletim sistemlerinde desteklenmektedir [42].

2.3.6 ArrayFire

ArrayFire C, C++ ve Fortran üzerinde yapılan kodlamaların grafik işlemcilerde çalıştırılmasına yardımcı olan bir yazılım kütüphanesidir. İki versiyonu vardır; birincisi CUDA GPU'lar için, diğeri de OpenCL aygıtlar içindir. Kullanımı kolay çok miktarda fonksiyon barındırmaktadır. Bu fonksiyonlar görüntü işleme, lineer cebir, işaret işleme, istatistik vb. alanlarda kullanılabilirler. Görsel olarak da aynı anda OpenGL ile birlikte çalışabilecek grafik kütüphanelerine sahiptir.

ArrayFire Cuda destekli Nvidia GPU'ları, OpenCL destekli aygıtları destekler. OpenCL destekli aygıtlara örnek olarak AMD GPU ve CPU ile Intel CPU'lar gösterilebilir. Aynı zamanda ARM, Qualcomm vb. destekli mobil cihazları da desteklemektedir.

ArrayFire yüksek başarım için yapılmış bir yazılım kütüphanesidir. Maksimum üretkenliği ve hızı hedef almaktadır [33]. Ek olarak, ArrayFire açık kaynak kodlu bir yazılım olmuştur [34].

3 BÖLÜM

DENEYSEL ÇALIŞMALAR

3.1 Giriş

Bu bölümde öncelikle, deneylerde kullanılan yazılımlar, donanımlar ve veri kümesi tanıtılmıştır. Sonrasında yapılan deneylerde elde edilen sonuçlar hem sayısal olarak hem de görsel olarak verilmiştir. Sonuçlardan beklenti paralel programlamayla işlemlerin hızını artırmak olduğundan daha çok deney sonuçlarında performans üzerinde durulmuştur.

Kalite metrikleri, performans ölçümlerinde programın çalışma zamanı dikkate alınmıştır. Aynı zamanda görüntü birleştirmede performansa odaklanıldığı gibi resmin kalitesinin de seri veya paralel uygulamalarda değişmemesi üzerinde durulmuştur.

3.2 Deneysel Düzenek

Her bir test verisi üzerine 30 adet deney yapılmış olup, elde edilen sayısal performans değerleri bu deneylerin ortalamasıdır.

3.2.1 Kullanılan Yazılımlar

Kodlamalar C++ programlama dili üzerinde, seri ve paralel programlama olarak iki farklı şekilde yapılmıştır. Seri programlama C++ kodlarıyla yapılmıştır [35]. Böylece paralel programlama ile seri programlama arasındaki performans karşılaştırması için deneyler yapılması sağlanmıştır.

Paralel programlamada ise CUDA ve OPENCL olarak, iki farklı dile de destek veren ArrayFire paralel programlama kütüphanesi kullanılmıştır. ArrayFire kütüphanesi paralel programlamada kullanılmak üzere oldukça kullanışlı bir dizi fonksiyona sahip bir kütüphanedir [34].

CUDA, Nvidia firmasının geliştirdiği grafik işlemciler üzerinde genel amaçlı programlama yapmaya imkân tanıyan bir programlama dilidir. Desteği sadece Nvidia grafik kartları içindir. Fakat piyasadaki en büyük grafik kart üreticilerinden biri olan Nvidia firması CUDA programlama dilinin arkasında durmakta ve daha fazla gelişmesi için çaba harcamaktadır. Nvidia firmasına ait grafik kartlarının günümüzde oldukça yaygın olarak kullanılması da bu dilin yaygınlaşmasına olanak sağlamaktadır [31].

OpenCL programlama dili ise KRONOS adlı teknoloji devlerinin oluşturduğu birlik tarafından kurulan ve desteklenen paralel programlama dilidir. Nvidia CUDA programlama dilinden farklı olarak GPU, CPU, ARM vb. birçok cihazda çalışmaktadır. Ayrıca Nvidia firması dahil piyasada birçok büyük çaplı firma tarafından destek görmektedir [32].

3.2.2 Kullanılan Donanımlar

Grafik kartlarında performansımızı artırmasını beklediğimiz en önemli donanım özelliği GPU çekirdek sayısıdır. Çünkü paralel programlamada GPU'lardaki bu çekirdekler kullanılmakta ve her bir çekirdek bir işlemci birimi gibi davranarak performans artışı sağlanmaktadır.

Bunların yanında veri yolu ve ram miktarları da önemli etkenlerdendir. Çünkü grafik işlemcinin özelliklerini kullanmak için CPU ve GPU arasında oldukça fazla veri akışı olmaktadır.

3.2.2.1 AMD CPU

- **AMD Phenom II X6 1055T 2.8Ghz işlemci [39]**

Tablo 3.1 AMD Phenom II X6 1055T 2.8Ghz işlemci

Frekans sayısı	2.8 GHz
Toplam L2 Cache	3MB
L3 Cache	6MB
Power	125W
CMOS Tech.	45nm SOI

3.2.2.2 Nvidia CUDA Destekli Grafik Kartları

Nvidia CUDA grafik kartlarında CUDA programla dilinin kullanımına özel CUDA çekirdekleri de mevcuttur.

- **Nvidia GeForce GTX 560 Ti** [36]

Tablo 3.2 Nvidia GeForce GTX 560 Ti

GPU Motoru Özellikleri		**Teknik Özellikler**	
CUDA çekirdeği	**384**	Maksimum Dijital Çözünürlük	2560x1600
İşlemci Saat Hızı	1645	Ortam Bağlantısı	2048x1536
Grafik Saat Hızı (Mhz)	822	En Yüksek VGA Çözünürlük	Mini HDMI, DVI
Doku Doldurma Hızı (milyar/san)	52.5	Çoklu Monitör	X
Bellek Özellikleri		HDCP	X
Bellek Hızı (Mhz)	4008	HDMI	X
Bellek Hızı	1024	HDMI için ses girişi	Internal
Bellek Arayüzü	256-bit GDDR5	**Standart Grafik Kartı Boyutları**	
Bellek Bant Genişliği	128	Yükseklik	4.376 inç Inç
Özellik Desteği		Uzunluk	9 inç Inç
Programlama Ortamı	CUDA	Yol Desteği	PCI-E2.0x 16
DirectX	11		
OpenGL	4.1		

Tablo3.2'deki Nvidia grafik kartı orta seviye bir kart olup, görece eski bir karttır. Fakat Nvidia firmasının paralelleştirmeye verdiği destek dolayısıyla daha iyi performans sergileyebilmektedir. CUDA desteği bu kartta mevcuttur. Toplam 384 CUDA çekirdeği mevcuttur.

3.2.2.3 AMD/ATI Destekli Grafik Kartları

- **AMD/ATI RADEON™ HD 7950 GPU[37]**

Tablo 3.2 AMD/ATI RADEON™ HD 7950 GPU

GPU Saat Hızı	850 MHz
Hafıza Miktarı	**3GB GDDR5**
Hafıza Saat Hızı	1250MHz (5.0 Gbps GDDR5)
Hafıza Bant Genişliği	240GB/s
Tekli İşlem Gücü	2.87 TFLOPS
GCN Mimarisi	28 işleme birimi (**1792** akış işlemcisi)
	128 z/stencil ROP birimi
	32 renk ROP birimi
	Çift geometri birimi
	Çift Eşzamansız İşleme Motorları (ACE)
Veri Yolu Arayüzü	PCI Express 3.0 x16

- **AMD/ATI R9-280X GPU [38]**

Tablo 3.3 AMD/ATI R9-280X GPU

GPU Mimarisi	28nm
API Desteği	DirectX® 12, Mantle, OpenGL 4.3, OpenCL
PCI Express Versiyon	3
GPU Saat Hızı	1000 MHz
Hafıza Bant Genişliği	288 GB/s
Hafıza miktarı	3GB GDDR5
İşlemi birimi sayısı	**2048**

3.2.2.4 Grafik Kartları Karşılaştırması

Tablo 3.4 Grafik kartları çekirdek sayısı ve işlem gücü karşılaştırması

	Çekirdek Sayısı	Ç.Saat Hızı	Teorik GFLOP [36-38,43]
Nvidia GeForce GTX 560 Ti	384	823	1263
AMD 7950	1792	925	2867
AMD R9 280X	2048	1150	3481

Tablo 3.5'de de görüleceği üzere teorik GFLOP en yüksek işlem gücüne sahip kart AMD R9 280X, dolayısıyla en yüksek performans bu karttan beklenebilir. AMD 7950 ise ikinci derecede güçlü olan kart, Nvidia GeForce GTX 560 Ti ise üçüncü sırada işlem gücüne sahiptir. Fakat Nvidia kart mimarisi farklı olduğundan ve AMD ile yazılım desteği olarak da farklı durumda olduklarından, işlem gücü değerleri bazen işlemler üzerinde beklenilen sonucu vermeyebilir.

3.3 Test Görüntüleri

Bu bölümde deneylerde kullanılan veriler tanıtılacaktır. Yapay olarak elde edilen görüntüler ve deneysel olarak elde edilen görüntüler deneylerde kullanılacaktır. Ayrıca filtreleme için kullanılan Gaussian filtre verilecektir.

Yapay olarak elde edilen görüntüler;

Gerçek odaklı resimlerle çalışıldığında referans resim olmadığından referans tabanlı kalite metrikleriyle görüntü işlemenin kalitesi ölçülememektedir. Bu nedenle yapay olarak elde edilen çoklu odaklanmış görüntülere ihtiyaç vardır.

Bu işlem için öncelikle elimizde net bir resmin olması gerekmektedir. Bu net resimden öncelikle resmin bir bölgesi seçilerek alçak geçiren filtre ile bulanıklaştırılır, yeni resim kaydedilir. Sonrasında ise, net resimden bu seçilen kısım hariç diğer kısmı seçilir ve bu kısım bulanıklaştırılır ve kaydedilir. Böylelikle elde referans resmin iki adet bulanıklaşmış hali mevcut olur.

Büyük çözünürlüklü ve küçük çözünürlüklü görüntülerin karşılaştırılabilmesi için iki adet resimden toplam 4 adet yapay olarak odaklanmış görüntü elde edilmiştir.

Arabalar adlı görüntü 256×256 çözünürlüğünde görece küçük boyutlu bir resimdir. Arabalar adlı net görüntüden yapay olarak Arabalar1 ve Arabalar2 adlı iki yeni görüntü oluşturulmuştur.

Şekil 3.1 Arabalar (Net Görüntüsü)

Şekil 3.1'de Arabalar net görüntüsü verilmiştir. Bu görüntüde resmin içindeki iki arabada net olarak gözükmektedir. Bu görüntüde çeşitli yazılımlarla arkadaki bölgeyi bulanıklaştırsak yeni bir bulanık resim elde ederiz. Şekil 3.2'deki Arabalar1 görüntüsünde öndeki araba net olarak bırakılmıştır. Arkadaki araba ise net değildir.

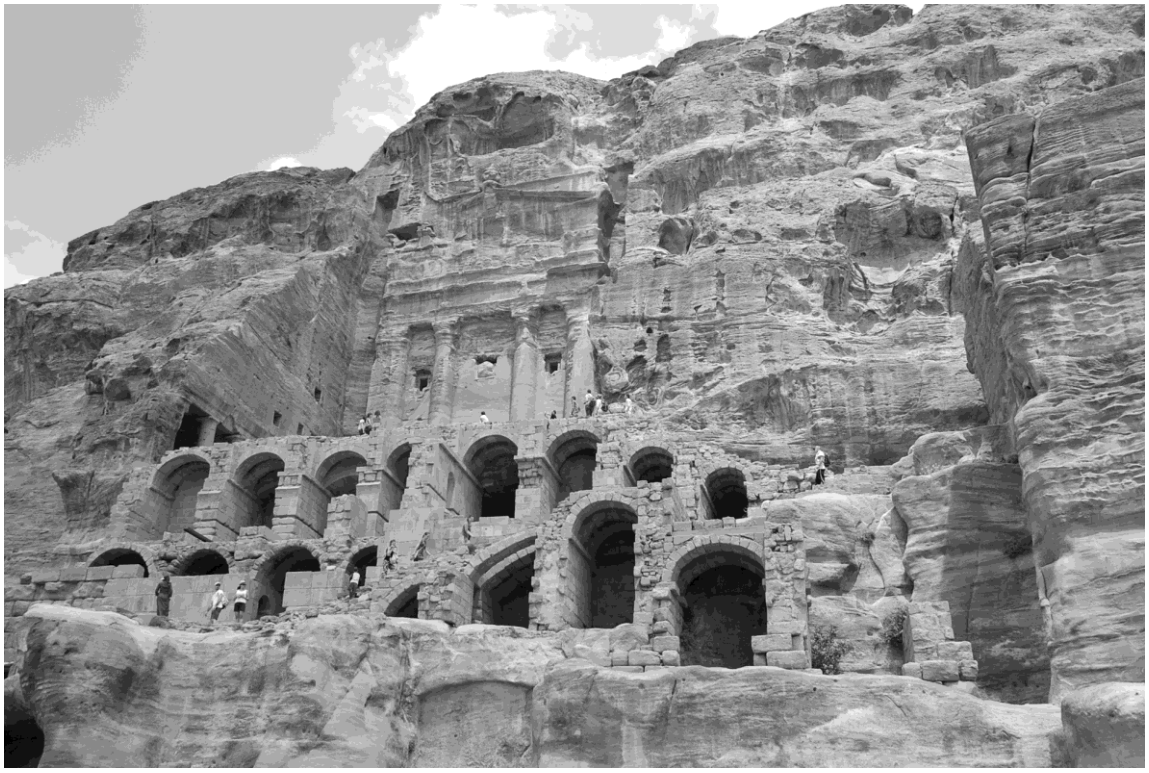
Aynı işlemi bu sefer öndeki araba bulanık ve arkadaki araba net olarak elde edebiliriz. İşlemler sırasında ojinal net görüntü saklanır. Daha sonra birleştirilmiş görüntülerle karşılaşıtırılabilir.

Şekil 3.2 Arabalar1 (Bulanık Görüntüsü)

Şekil 3.3'de Arabalar2 görüntüsü verilmiştir. Arabalar net görüntüsünün ön tarafını bulanıklaştırarak Arabalar2 bulanık görüntüsünü elde edilmektedir.

Şekil 3.3 Arabalar2 (Bulanık Görüntüsü)

Petra adlı görüntü net bir görüntüdür. Petra adlı görüntüden Petra1 ve Petra2 adlı iki yeni görüntü elde edilmiştir. Çözünürlüğü 1936×1296 olup, görece orta çaplı boyutta bir resimdir.



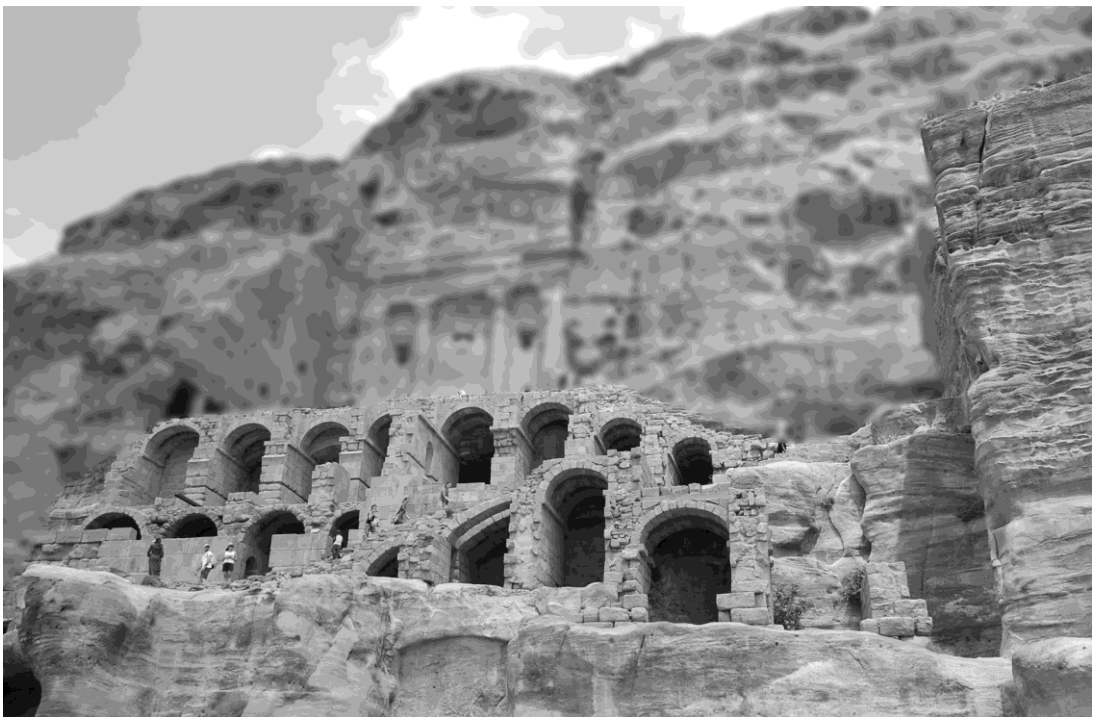
Şekil 3.4 Petra (Net Görüntüsü)

Şekilde 3.4'de Petra görüntüsünün net hali verilmiştir. Görüntünün her tarafı nettir. Bu görüntünün önce alt taraf bulanıklaştırılır, üst tarafı ise net olarak bırakılır. Böylece eldeki resmin artık bir kısmı bulanık bir kısmı nettir.

Şekil 3.5'teki yeni Petra1 adlı bulanık görüntü Petra net görüntünün alt tarafı bulanıklaştırılarak elde edilmiş halidir. Petra net resmin orjinal hali Araba net resmi saklandığı gibi saklanır. Bu durumda daha sonra oluşturulacak olan birleşmiş net görüntüyle karşılaştırma imkanına sahip oluruz.

Şekil 3.5 Petra1 (Bulanık Görüntüsü)

Şekil 3.6'de Petra net görüntüsünden üst taraf bulanıklaştırılarak yeni Petra2 bulanık görüntüsü elde edilmiştir.



Şekil 3.6 Petra2 (Bulanık Görüntüsü)

- **Deneysel olarak elde edilen görüntüler;**

Bu bölümde fotoğraf makinesiyle çekilen görüntüler üzerinde yapılan çalışmalar açıklanmıştır. Nikon firmasına ait D7000 model kamera ile elde edilmiştir. Beraberinde gelen uygulama geliştirme arabirimi (API) programlama kütüphaneleri ile C# ve MATLAB gibi ortamlara kolaylıkla entegre edilebilmektedir. Bu cihazın en önemli özelliği yakınlaştırma, odaklama, diyafram açıklığı, pozlama süresi ve kazanç vb. tüm profesyonel kamera parametrelerinin her hangi bir programlama diliyle geliştirilen bir yazılımla kontrol edilebiliyor olmasıdır [8].

CdKitap1 adlı görüntü arka tarafa odaklanıldığından ön taraf bulanık çıkmış doğal bir görüntüdür. Arka taraftaki kitaplar netlik aralığı içinde olduğundan net çıkmışlardır. Ön tarafaki cd ise netlik aralığı dışında olduğundan bulanık çıkmıştır. Görüntü 4928×3264 çözünürlüklerinde olup günümüz şartlarında büyük sayılabilecek bir boyuttadır. İlgili görüntü Şekil 3.7'de verilmiştir.

Şekil 3.7 CdKitap1 (Bulanık Görüntüsü)

Şekil 3.8'de CdKitap2 ise yine aynı çevrenin bu sefer ön tarafa odaklanılmasıyla elde edilmiş görüntüsüdür. Bu defa da arka taraf bulanık çıkmıştır.

Şekil 3.8 CdKitap2 (Bulanık Görüntüsü)

Gaussian Filtre

5×5'lik bir Gaussian filtre kullanılmıştır. Filtrede elemanların toplamı 1 olmalıdır. Ayrıca bir birlerine simetrik bir yapıda olmalıdırlar. Kullanılan filtre aşağıda verilmiştir.

Tablo 3.5 Gaussian filtre

0.0030	0.0133	0.0219	0.0133	0.0030
0.0133	0.0596	0.0983	0.0596	0.0133
0.0219	0.0983	0.1621	0.0983	0.0219
0.0133	0.0596	0.0983	0.0596	0.0133
0.0030	0.0133	0.0219	0.0133	0.0030

3.4 Deneysel Sonuçlar

Bölümde deney düzeneğinde, test verilerinde yapılan deneylerin sonuçları verilmiştir. Performans odaklı bir çalışma olduğundan daha çok yapılan işlemlerin süreleri üzerine durulmuştur. Laplacain Piramit adımları ile ilgili süreler hesaplanmıştır. Her cihaz için ayrı ayrı zaman hesaplamaları yapılarak birbirleriyle karşılaştırmalar yapılmıştır.

Kalite metrikleri ölçümünde SSIM algoritması kullanılarak, çıkan resmin orijinal resme benzerliği ölçülmüştür.

Süreler saniye cinsinden olup virgülden sonra 4 hane alınmıştır. Karşılaştırmalarda ise CPU'da yapılan işlemlerdeki geçen süre GPU cihazlarındaki işlemlerde geçen süreye bölünerek hız artışı katsayıları elde edilmiştir.

Şekil 3.9'da CdKitap1 ve CdKitap2 çoklu odaklı resimlerinin Laplacian Piramit ile birleştirilerek net görüntülerin elde edilmesi adım adım gösterilmiştir.

Birinci adımda, her iki resim içinde, Gaussian Piramidi oluşturulmuştur. İlk önce GF'den geçen orijinal görüntü dört komşu pikselden bir piksel oluşturularak dörtte bir oranına getirilmiştir. Yanı enine ve boyuna yarı yarıya küçültülmüştür. Bu durumda G0(orijinal görüntü)'dan G1 GP adımı elde edilmiştir.

Aynı işlemler uygulanarak G1 seviyesinden G2 seviyesindeki GP katmanı elde edilir. Bu şekilde seviye sayısı kadar ilerlenir en sonunda G4 adımına kadar varılır. Bu testlerde seviye 4 alınmıştır. Bu seviye örüntü olarak kabul edilmiş olup, yüksek kalite performans oranına sahiptir. Zaten resim boyutlarına göre belirli bir adımdan sonraki resimler bir anlam ifade etmemektedirler.

Böylece artık elde her iki farklı odaklı görüntü içinde GP mevcuttur. Sonraki adımda GP kullanılarak LP elde edilmiştir. Burada yine bir dizi işlem gerçekleştirilmiştir. Birinci adımda G0 ve G1 seviyesindeki görüntüler kullanılarak LP sıfırıncı seviye L0 elde edilir.

Şekil 3.9 CdKitap1 ve CdKitap2 için Gaussian Piramidi Adımları.

L0 seviyesindeki görüntüyü elde edebilmek için G1 GP görüntüsü her piksel 4 piksel edecek şekilde bir genişletme yapılır. Sonrasında Gaussian Filtre ile filtrelemeye tabi tutulur. Bu işleme genişletme denir. Eldeki geniş resim G0 resminden çıkarılır. Bu hata farkı sıfırıncı seviye LP L0 değerini verir.

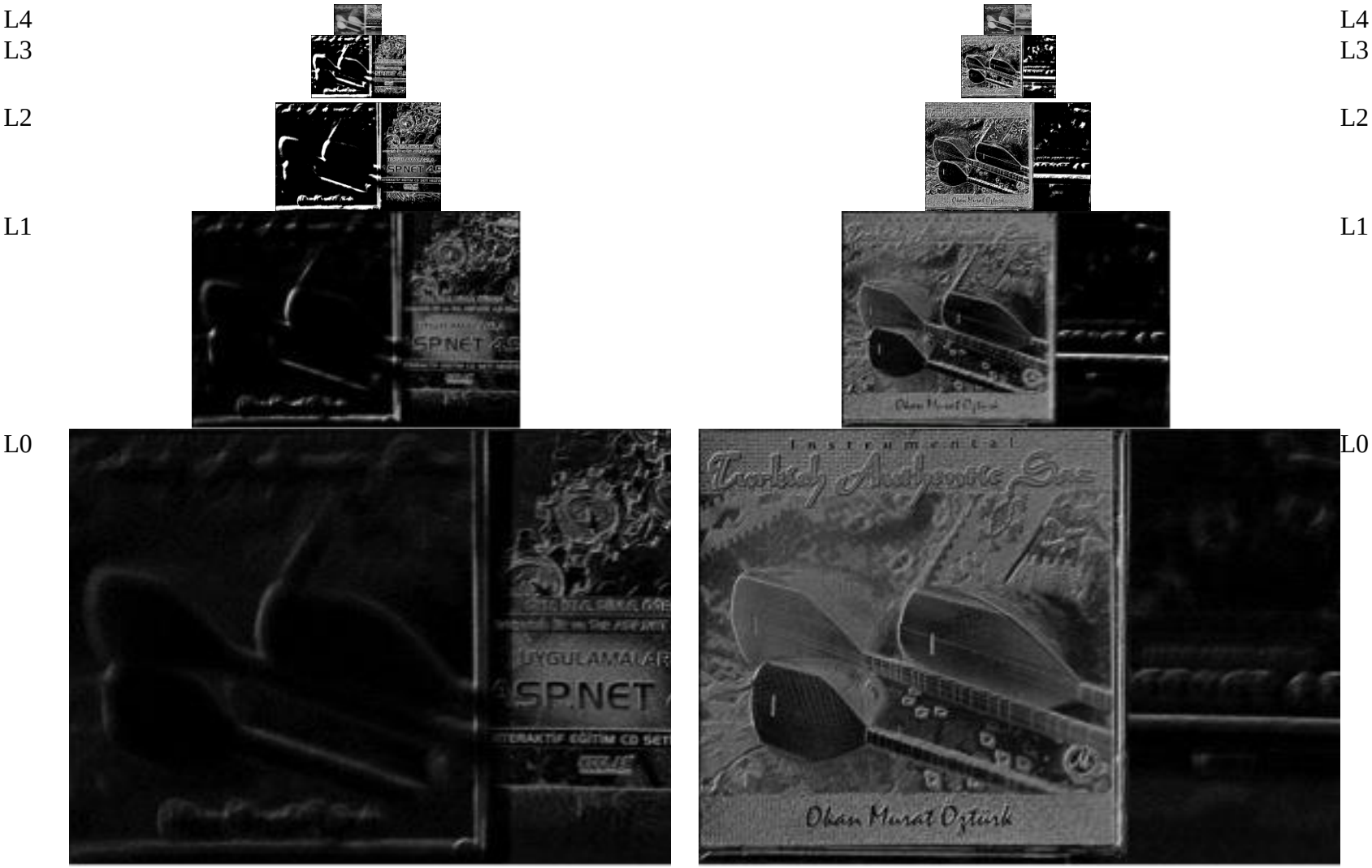
Aynı adımları G1 ve G2 içinde yaparız, GP'nin bu seviyedeki görüntülerinden L1 elde edilir. Bu şekilde adım sayısına kadar gidilir. Fakat son adımda G(N+1) seviyesinde GP seviyesi olmadığından son LN değeri GN değeriyle eşitlenir.

Aynı işlemler her iki resme ait GP için uygulanarak iki tane LP elde edilir. Artık elde her iki resim için de LP vardır. Her iki resim için oluşturulan LP seviyelerindeki görüntüler seçilme işlemiyle daha az bulanık olan LP adımı elde edilir. Her LP seviyesinde her iki resmin LP görüntüleri karşılaştırılır. Daha net olan kısımlar birleştirilir. Örneğin birinci görüntünün L0 görüntüsü ve ikinci resmin L0 görüntüsü karşılaştırılarak hangisinin hangi parçası daha netse o kısmı alınır ve yeni birleşik bir L0 elde edilir. Buna seçim işlemi denir. Bu şekilde her LP adımları için toplam seviye sayısı kadar yeni birleşik LP seviyesi oluşturulmuş olur. Şekil 3.10'da oluşturulan LP seviyelerindeki görüntüler verilmiştir.

Artık elde birleştirilmiş LP görüntüleri mevcuttur. Yani yeni birleştirilmiş LP seviyeleri vardır. Bu LP seviyesindeki görüntülere ters işlemler uygulanarak yeni net görüntü elde edilebilir. Burada da en sondaki LP seviyesinden başlayarak önce büyütme sonra önceki adımla toplama işlemi yapılır. Yani L(N) seviyesindeki görüntü 4 katına çıkarılır. L(N+1) seviyesindeki LP adımıyla toplanır. Yeni elde ettiğimiz LP görüntüsüne de yine aynı işlemler uygulanarak L0 seviyesine kadar gelinir.

Son durumda eldeki L0 seviyesi net görüntüdür. Her iki görüntünün de net kısımlarına sahip bir görüntü elde edilmiştir.

Şekil 3.11 'de seçilen LP adımları gösterilmiştir. Yine aynı şekilde yan tarafta, bu piramidin birleştirme algoritmasıyla birleştirilip elde edilen, birleşmiş resim verilmiştir.



CdKitap1 CdKitap2

Şekil 3.10 CdKitap1 ve CdKitap2 için Laplacian Piramidi Adımları.

Şekil 3.11 CdKitap1 ve CdKitap2 için Seçilen Laplacian Piramidi Adımları.

Aşağıdaki tabloda deneylerdeki cihazlara kısa isimler verilmiştir.

Tablo 3.6 CPU ve GPU kısa isimlendirmeleri

CPU	AMD Phenom II X6 1055T 2.8Ghz İşlemci
G560OCL	Nvidia GeForce GTX 560 Ti (OpenCL Kodu ile)
G560CUDA	Nvidia GeForce GTX 560 Ti(CUDA Kodu ile)
280X	AMD R9 280X
7950	AMD 7950

Her deney 30 kez tekrarlanmıştır. Deney sonucunda ortaya çıkan birleştirme süreleri saniye(sn) cinsinden verilmiştir. Yine bu deneylerde hesaplanan sürelerin standart sapmaları verilmiştir. GPU işlem birimlerinin CPU işlem süresine göre karşılaştırmaları da tablo halinde verilmiştir.

Tablo 3.7 Araba1 ve Araba2 resimleri birleştirme işlemi için geçen süreler(sn).

	CPU	**G560OCL**	**G560CUDA**	**280X**	**7950**
Konvolusyon0	0.003699	0.000234	0.000171	0.000292	0.000358
İndirgeme0	0.000592	0.000124	0.000105	0.000377	0.000433
Genişletme0	0.000941	0.000624	0.000116	0.000337	0.000348
GenişletmeConv0	0.003809	0.000261	0.000552	0.000370	0.000476
Çıkarma0	0.000377	0.000035	0.000057	0.000049	0.000055
Seçme0	0.000757	0.000063	0.000073	0.000070	0.000084
Birleştirme	0.063239	0.000564	0.000017	0.000452	0.000427
LapToplam	0.016264	0.003506	0.003499	0.004422	0.004388

Tablo 3,8'de Araba resimlerinin birleştirilmesine ait işlem süreleri verilmiştir. Araba resimleri küçük boyutlu olduğundan süreler göreceli olarak az çıkmıştır. Ayrıca genel olarak cihazlar güçlü ve yapılan işlemde sadece iki resmin birleştirilmesi olduğundan süreler düşük çıkmaktadır. Fakat bu işlemlerin ardı ardına çok fazla miktarda yapıldığı uygulamalarda sürenin önemi daha artacaktır.

Tablo 3.8 Araba1 ve Araba2 resimleri birleştirme işlemi için geçen sürelerin Standart Sapması.

	CPU	**G560OCL**	**G560CUDA**	**280X**	**7950**
Konvolusyon0	0.000225	0.000034	0.000013	0.000057	0.000041
İndirgeme0	0.000047	0.000037	0.000028	0.000090	0.000109
Genişletme0	0.000045	0.000075	0.000021	0.000063	0.000060
GenişletmeConv0	0.000217	0.000044	0.000069	0.000048	0.000101
Çıkarma0	0.000158	0.000010	0.000007	0.000017	0.000023
Seçme0	0.000104	0.000016	0.000008	0.000014	0.000037
Birleştirme	0.000157	0.000117	0.000249	0.000123	0.000077
LapToplam	**0.000672**	**0.000242**	**0.000004**	**0.000460**	**0.000286**

Tablo 3.9'da araba resimlerinin birleştirilmesinde standart sapmalar(SS) küçük çıkmıştır. Yani yapılan testleri sonuçları genel olarak bir birine yakındır.

Tablo 3.9 Araba1 ve Araba2 resimleri birleştirme işlemi için geçen sürelerin CPU'da hesaplanan süreye oranı.

	G560OCL	**G560CUDA**	**280X**	**7950**
Konvolusyon0	15.81	21.59	12.65	10.35
İndirgeme0	4.77	5.64	1.57	1.37
Genişletme0	1.51	8.08	2.79	2.71
GenişletmeConv0	14.57	6.90	10.30	8.00
Çıkarma0	10.85	6.62	7.65	6.90
Seçme0	12.11	10.37	10.75	9.06
Birleştirme	112.12	3646.37	139.81	148.18
LapToplam	**4.64**	**4.65**	**3.68**	**3.71**

G560OCL, 280X ve 7950 cihazlarından elde edilen süreleri CPU'dan elde edilen sürelere bölerek elde edilen Araba resimlerine ait sonuçlar tablo 3.10'da listelenmiştir. Sonuçlar bir birlerine yakın çıkmıştır. Bunun temel nedeni araba1 ve araba2 görüntüsünün boyutlarının küçük olmasıdır. Bu durum grafik kartları arasındaki

performans farkının ortaya çıkmasını engellemiştir. Buna rağmen hepsi de seri programlamaya göre yaklaşık 3-4 kat daha hızlı işlem yapmıştır.

Tablo 3.10 Petra1 ve Petra2 resimleri birleştirme işlemi için geçen süreler(sn).

	CPU	**G560OCL**	**G560CUDA**	**280X**	**7950**
Konvolusyon0	0.181936	0.002874	0.002720	0.000966	0.003695
İndirgeme0	0.033703	0.000775	0.000796	0.000638	0.000989
Genişletme0	0.034427	0.001716	0.001951	0.001264	0.001955
GenişletmeConv0	0.181198	0.003680	0.003617	0.001545	0.004212
Çıkarma0	0.064619	0.000041	0.000067	0.000086	0.000089
Seçme0	0.077979	0.000060	0.000076	0.000082	0.000076
Birleştirme	0.151362	0.000527	0.000016	0.000345	0.000343
LapToplam	**0.953434**	**0.017260**	**0.019956**	**0.007240**	**0.015077**

Tablo 3.11'de Petra1 ve Petra2 görüntüleri için geçen süreler verilmiştir. Görüntünün boyutu arabalar görüntülerine göre biraz daha büyük olduğundan işlem süresi de biraz daha artmıştır.

Tablo 3.11 Petra1 ve Petra2 resimleri birleştirme işlemi için geçen sürelerin Standart Sapması.

	CPU	**G560OCL**	**G560CUDA**	**280X**	**7950**
Konvolusyon0	0.001589	0.000030	0.000035	0.000066	0.000055
İndirgeme0	0.000622	0.000106	0.000070	0.000110	0.000065
Genişletme0	0.000334	0.000108	0.000128	0.000071	0.000113
GenişletmeConv0	0.001101	0.000137	0.000146	0.000118	0.000069
Çıkarma0	0.000952	0.000010	0.000011	0.000049	0.000094
Seçme0	0.000679	0.000011	0.000021	0.000034	0.000031
Birleştirme	0.001025	0.000095	0.000006	0.000078	0.000061
LapToplam	**0.003564**	**0.000451**	**0.000557**	**0.000507**	**0.000238**

Tablo 3.12'de Petra1 ve Petra2 resimlerine ait deneylerden elde edilen sonuçlarda standart sapma(SS) değerleri verilmiştir. bu verilere göre SS sıfıra yaklaşarak deney sonuçlarının stabil olduğunu göstermektedir.

Tablo 3.12 Petra1 ve Petra2 resimleri birleştirme işlemi için geçen sürelerin CPU'da hesaplanan süreye oranı.

	G560OCL	**G560CUDA**	**280X**	**7950**
Konvolusyon0	63.30	66.90	188.35	49.24
İndirgeme0	43.51	42.33	52.79	34.07
Genişletme0	20.06	17.64	27.24	17.61
GenişletmeConv0	49.24	50.09	117.32	43.02
Çıkarma0	1593.75	962.29	748.75	729.93
Seçme0	1289.23	1026.86	956.62	1028.30
Birleştirme	287.00	9496.07	439.27	441.29
LapToplam	**55.24**	**47.78**	**131.68**	**63.24**

Tablo 3.13'de Petra1 ve Petra2 görüntülerinin GPU'larda paralel programlama değerleriyle CPU'da seri programlama sonuçları karşılaştırılmıştır. Bu görüntü işlemenin sonuçlarından da bariz olarak görüldüğüne göre görüntü boyutu büyüdüğünden GPU'ların işlem güçleri ortaya çıkmaya başlamıştır. Ayrıca GPU'larda kendi içlerinde farklılık arz etmeye başlamışlardır. En güçlü GPU olan 280X diğerlerine oranla çok daha hızlı işlem yapmıştır.

Tablo 3.13 CdKitap1 ve CdKitap2 resimleri birleştirme işlemi için geçen süreler(sn).

	CPU	**G560OCL**	**G560CUDA**	**280X**	**7950**
Konvolusyon0	1.149614	0.017447	0.016636	0.005253	0.022494
İndirgeme0	0.209247	0.002157	0.002302	0.001346	0.002708
Genişletme0	0.249090	0.007211	0.007862	0.006552	0.011657
GenişletmeConv0	1.152514	0.018878	0.018177	0.008328	0.024851
Çıkarma0	0.373820	0.000047	0.000072	0.000074	0.000101
Seçme0	0.456986	0.000058	0.000073	0.000078	0.000095
Birleştirme	0.974700	0.000537	0.000015	0.000366	0.000388
LapToplam	**6.133968**	**0.069276**	**0.070270**	**0.028733**	**0.081261**

Tablo 3.14'de CdKitap1 ve CdKitap2 görüntülerine ait birleştirme süreleri verilmiştir. CdKitap görüntüleri, çözünürlük olarak eldeki en büyük görüntülerdir. Bu nedenle de süre olarak en fazla vakit alan birleştirme işlemleri bu resimlerin olmuştur. Ayrıca CPU bu birleştirme işlemin zorlanmaya başlamıştır.

Tablo 3.14 CdKitap1 ve CdKitap2 resimleri birleştirme işlemi için geçen sürelerin Standart Sapması.

	CPU	**G560OCL**	**G560CUDA**	**280X**	**7950**
Konvolusyon0	0.009826	0.000030	0.000044	0.000166	0.000098
İndirgeme0	0.003360	0.000095	0.000114	0.000073	0.000096
Genişletme0	0.007259	0.000169	0.000143	0.000174	0.000168
GenişletmeConv0	0.006106	0.000166	0.000560	0.000233	0.000215
Çıkarma0	0.004685	0.000008	0.000011	0.000039	0.000040
Seçme0	0.003189	0.000010	0.000012	0.000041	0.000033
Birleştirme	0.007747	0.000123	0.000002	0.000090	0.000081
LapToplam	**0.007203**	**0.000451**	**0.000991**	**0.000589**	**0.000366**

Tablo 3.15'de CdKitap1 ve CdKitap1 resimlerine ait deneylerden elde edilen sonuçlarda standart sapma(SS) değerleri verilmiştir. Değerler sıfıra yakındır ve sonuçların istikrarlı olduğunu göstermektedir.

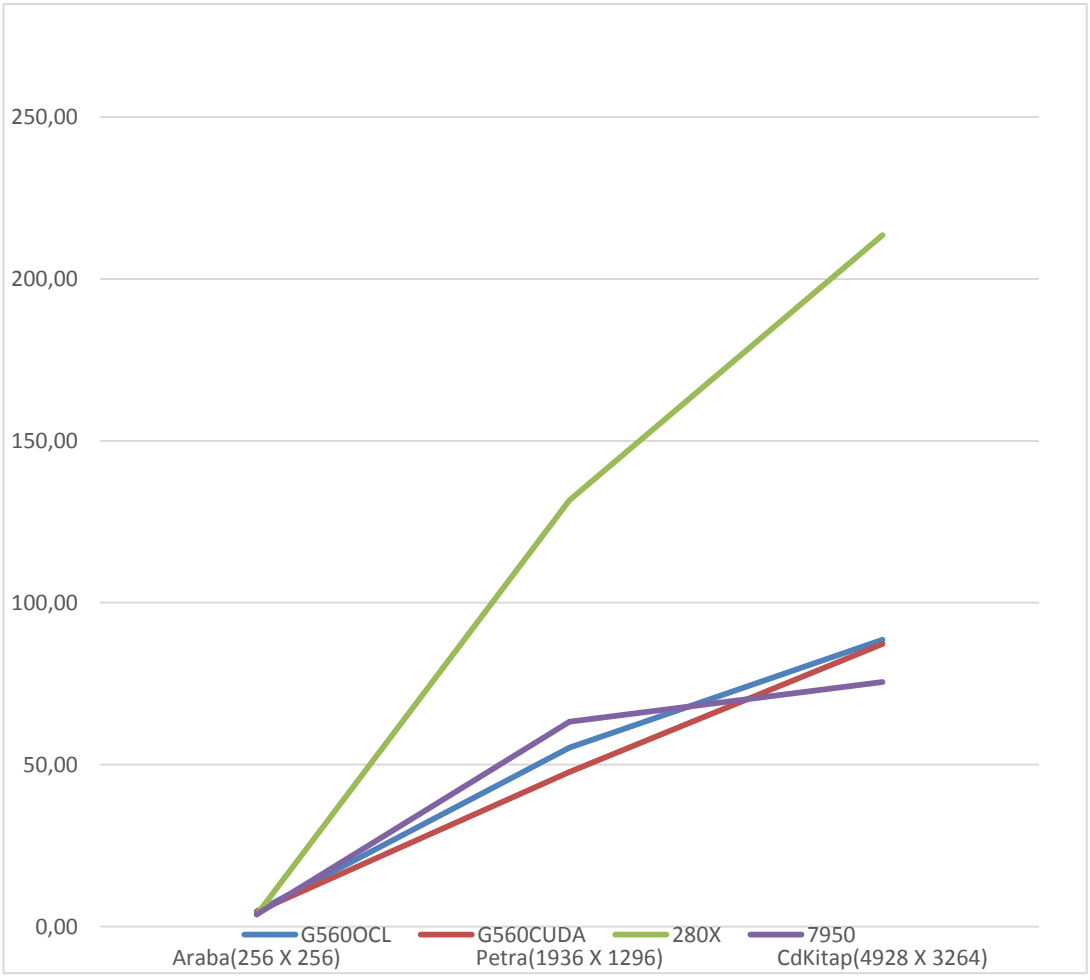
Tablo 3.15 CdKitap1 ve CdKitap2 resimleri birleştirme işlemi için geçen sürelerin CPU'da hesaplanan süreye oranı.

	G560OCL	**G560CUDA**	**280X**	**7950**
Konvolusyon0	65.89	69.10	218.86	51.11
İndirgeme0	97.03	90.91	155.47	77.27
Genişletme0	34.54	31.68	38.02	21.37
GenişletmeConv0	61.05	63.41	138.40	46.38
Çıkarma0	7911.54	5213.24	5035.13	3698.97
Seçme0	7947.58	6232.46	5881.65	4790.52
Birleştirme	1814.58	64852.83	2664.66	2513.29
LapToplam	**88.54**	**87.29**	**213.48**	**75.49**

Tablo 3.16'da gösterilen CdKitap1 ve CdKitap2 görüntülerinin birleştirilmesine ait işlemlerin CPU'ya göre hızları verilmiştir. Bu tabloya göre resimlerin boyutları büyük olduğundan sonuçlarda da bu büyüklüğün etkisi görülmüştür. GPU'ların işlem güçleri daha iyi kullanılmış ve hız farkı artmıştır. Yine 280X, G560OCL ve 7950'e göre işlem gücü farkını daha iyi ortaya koymuştur.

NVidia GPU hem OpenCL hem de NVidia CUDA programlama dilleriyle yapılan programlarla test edilmiş ve yaklaşık aynı sonuçlar elde etmiştir.

SSIM kalite metriğine göre; CPU üzerinde seri, GPU'lar üzerinde paralel ve Matlab ile elde edilen birleşmiş görüntülerin sonuçları yaklaşık olarak aynı çıkmışlardır. Bu durum, performans artışı sağlanırken kalitede bir düşüş yaşanmadığını göstermektedir.



Şekil 3.12 GPU'ların Hız Artış Karşılaştırması

Şekil 3.12’deki grafikte görüldüğü üzere, görüntülerin çözünürlükleri büyüdükçe CPU’daki seri programa göre, GPU’larda paralel çalışan programların çalışma hızındaki artış belirginleşmiştir. 280X cihazı diğerlerinden performans artışındaki yüksekliğiyle ayrılmıştır. Diğer üç şık da ise bir birine yakın sonuçlar alınmıştır.

4 BÖLÜM

TARTIŞMA SONUÇ VE ÖNERİLER

4.1 Tartışma, Sonuç

Çalışma kapsamında, çoklu odaklı görüntülerin grafik işlemciler üzerinde paralel programlama ile birleştirilmesi üzerine çalışılmıştır. Görüntülerin birleştirilmesinde performansın arttırılması esas amaç olmuştur. Geçmişe oranla giderek gelişen dijital teknoloji ve devamlı büyüyen görüntü işleme hacmi, bu alandaki performans araştırmalarını zorunlu kılmaktadır. Bu amaçla görüntüleri birleştirmek için grafik işlemcilerde genel amaçlı programlama(GPGPU) ile görüntü birleştirme işlemleri yapılmıştır.

GPGPU, grafik işlemcileri normalde sadece belirli fonksiyonları yapan cihazlar olmaktan çıkarıp, genel amaçlı programlamada da kullanılmasını sağlamaktadır. Bu alan henüz yeni olmasına rağmen CPU'larda performans artışındaki son zamanlarda ortaya çıkan darboğazı aşmada önemli görevler üstlenmektedir. GPU'lardaki çekirdekler sabit fonksiyonlu işlem birimleriyken artık programlanabilirlikleri artmaya başlamış ve araştırmacıların üzerinde program yapabilmesine olanak sağlamıştır.

Çalışmada bu alandaki yeni teknolojilerden, OpenCL ve CUDA tanıtılmıştır. Yine bu programlama dilleri için geliştirilen ArrayFire kütüphanesi anlatılmış ve çalışmada kullanılmıştır.

Laplacian Piramit metodu kullanılarak görüntü birleştirme işlemleri grafik işlemcilerde gerçekleştirilmiştir. Laplacian Piramit dönüşüm uzayında görüntü birleştirmeyi sağlayan bir görüntü birleştirme tekniğidir. İlk olarak 1983 yılında tanıtılmış olup [8] sık kullanılan yöntemlerden biridir.

Sonuçlar performans odaklı incelenmiştir. CPU ve GPU'lardan elde edilen sonuçlar süre olarak ve oran olarak çalışmada sunulmuştur. CPU'ya göre GPU'daki hız artışı belirgin bir şekilde ortaya konulmuştur. Görüntülerin çözünürlükleri arttıkça bu oran git gide artmıştır. Ayrıca GPU'ların kendi içinde de daha performanslı olanları daha iyi sonuçlar vermiştir.

4.2 Öneriler

Bu çalışma kapsamında görülmüştür ki henüz yeni gelişmeye başlayan GPGPU teknolojisi gelecek vaat etmektedir. Bu alandaki yenilikler çok iyi takip edilmeli ve sonuçları gözlemlenmelidir. Çoklu odaklı görüntü birleştirme metotlarından Laplacian Piramit üzerinde yapılan çalışma diğer görüntü birleştirme alanlarında da uygulanabilir. Görüntü birleştirme işlemleri haricinde, GPGPU programlama, diğer görüntü işleme alanlardaki performans iyileştirmeleri için de kullanılabilir.

Ek olarak, görüntü işleme dışında da GPGPU programlama üzerine araştırmalar yapılabilir. Veri yapılarının ve işlem hacimlerinin hızla arttığı ve artmaya devam edeceği gerçeğinden yola çıkarak her alanda bu çalışmalar yapılabilir.

KAYNAKLAR

1. Aslantaş, V., Kurban, R., 2009. A comparison of criterion functions for fusion of multi-focus noisy images. **Optics Communications, 282** (16): 3231-3242.
2. Wei, L., Xue-feng, Z., 2005. Medical image fusion and its application. **Chinese Journal of Medical imaging Technology, 21** (7): 1126-1129.
3. Blasch, E.P., Zheng, L., 2011. LANDSAT satellite image fusion metric assessment, pp. 237-244. *Aerospace and Electronics Conference (NAECON), Proceedings of the 2011 IEEE National*, 20-22 July 2011, Dayton, OH.
4. Cao, J., Zhou, Z., Wang, H., Liu, W., 2010. Multifocus noisy image fusion algorithm using the contourlet transform, pp. 1-4. *Multimedia Technology (ICMT), 2010 International Conference*, 29-31 Oct. 2010, Ningbo.
5. Zhang, H., Wang, P., Chen, X., Zhang, X., 2005. A new method for multi-source remote sensing image fusion, pp. 3948 - 3951. *Geoscience and Remote Sensing Symposium, 2005. IGARSS '05. Proceedings*, 25-29 July 2005.
6. Xu, L., Zhang, X., Lam, K., 2010. An image restoration method based on PDEs and a new gradient model, pp. 4165–4168. *Image Processing (ICIP), 2010 17th IEEE International Conference*, 26-29 Sept. 2010, Hong Kong.
7. Yan, L., Zhang, T., Zhong, S., 2006. A DSP/FPGA -Based paralel architecture for real-time ımage processing, pp. 10022-10025. *Intelligent Control and Automation, The Sixth World Congress*, June 21 - 23, 2006, Dalian.
8. Kurban, R., 2012. Resim Uzayında blok seçmeye dayalı yeni görüntü birleştirme yöntemleri, Erciyes Üniversitesi Fen Bilimleri Enstitüsü, Doktora Tezi, Kayseri.
9. Li, S., Kwok, J.T.-Y., Tsang, I.W., Wang, Y., 2004. Fusing images with different focuses using support vector machines. **Neural Networks, IEEE Transactions, 15** (6): 1555-1561.
10. Burt, P. J., Adelson, E. H., 1983. The laplacian pyramid as a compact ımage code. **IEEE Transactions on Communications, 31** (4): 532-540.
11. Annaratone, M., Arnould, E., Gross, T., Kung, H.T., Lam, M., Menzilcioglu, O., Webb, Jon A., 1987. The warp computer: architecture, ımplementation and performance. **Computers, IEEE Transactions, 36** (12): 1523-1538.

12. David M. H., Shirish P. K., C. Allan H., 2001. Low cost scaleable parallel image processing system. **Microprocessors and Microsystems, 25**: 143-157.
13. Garcia, V. ; Debreuve, E. ; Barlaud, M., 2008. Fast k nearest neighbor search using GPU, pp. 1 - 6. *Computer Vision and Pattern Recognition Workshops*, 23-28 June 2008, Anchorage, AK.
14. Pei, L. ; Xie, Z. ; Dai, J., 2008. Joint edge detector based on Laplacian pyramid, pp. 978 – 982. *Image and Signal Processing (CISP), 2010 3rd International Congress*, 16-18 Oct. 2010, Yantai.
15. Lakshmi, A., Rakshit, S., 2010. Gaussian Restoration pyramid : Application of image restoration to Laplacian pyramid compression, pp. 66 - 71. *Advance Computing Conference (IACC)*, 19-20 Feb. 2010, Patiala.
16. Watanabe, A., Taguchi, A., 2004. Improvement of the image enlargement method based on the Laplacian pyramid representation, pp. 585 - 588. *Circuits and Systems, 2004. MWSCAS '04. The 2004 47th Midwest Symposium*, 25-28 July 2004.
17. Pradeep,M., 2013. Implementation of image fusion algorithm using MATLAB (Laplacian Pyramid), pp. 165- 168. *Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), 2013 International Multi-Conference*, 22-23 March 2013, Kottayam.
18. Bulatov, A., 2006. Çok odaklı görüntü füzyonu, Erciyes Universitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Kayseri.
19. Şimşek, M., 2006. Ayrıştırma teknikleri kullanarak görüntü birleştirme, Gebze Yüksek Teknoloji Enstitüsü, Seminer raporu, Gebze.
20. Aslantaş, V., Kurban, R., 2010. Fusion of multi-focus images using differential evolution algorithm. **Expert Systems with Applications, 37** (12): 8861–8870.
21. **http://www.futurechips.org/chip-design-for-all/cpu-vs-gpgpu.html**(Erişim Tarihi: Ekim 2013)
22. Ikoma, N., 2014. On GPGPU parallel implementation of hands and arms motion estimation of a car driver with depth image sensor by particle filte, pp. 246-251. *World Automation Congress (WAC)*, 3-7 Aug. 2014, Waikoloa, HI.
23. Takahashi, R., Inoue, U., 2012. Parallel text matching using GPGPU, pp. 242 - 246. *Software Engineering, Artificial Intelligence, Networking and Parallel &*

Distributed Computing (SNPD), 2012 13th ACIS International Conference, 8-10 Aug. 2012, Kyoto.

24. Rogmans, S., Dumont, M., Lafruit, G., Bekaert, P., 2009. Migrating real-time depth image-based rendering from traditional to next-gen GPGPU, pp. 1-4. *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, 4-6 May 2009, Potsdam.
25. **http://www.gpgpu.org/w/index.php/Main_Page** (Erişim Tarihi: Ocak 2014)
26. Tang, Z., Won, Y., 2011. Multithread content based file chunking system in cpu-gpgpu heterogeneous architecture, pp. 58 - 64. *Data Compression, Communications and Processing (CCP), 2011 First International Conference*, 21-24 June 2011, Palinuro.
27. Iwase, Y., Abe, D., Yakoh, T., 2012. GPGPU aided method for real-time systems, pp. 841 - 845. *10th IEEE International Conference onDate of conference*, 25-27 July 2012, Beijing.
28. Archirapatkave, V., Sumilo, H., See, S.C.W., Achalakul, T., 2011. GPGPU Acceleration algorithm for medical image reconstruction, pp. 41 - 46. *Parallel and Distributed Processing with Applications (ISPA), 2011 IEEE 9th International Symposium*26-28 May 2011, Busan.
29. Hill, M.D.,Marty, M.R., 2008. Amdahl's law in the multicore era. **Computer, 41** (7): 33 - 38.
30. Woo, D., Lee, H.-H.S., 208. Extending Amdahl's law for energy-efficient computing in the many-core era. **Computer, 41** (12): 24 - 31.
31. **https://developer.nvidia.com** (Erişim Tarihi: Şubat 2014)
32. **http://www.khronos.org/opencl** (Erişim Tarihi: Şubat 2014)
33. **https://github.com/arrayfire** (Erişim Tarihi: Mart 2015)
34. **http://arrayfire.com** (Erişim Tarihi: Haziran 2014)
35. **https://isocpp.org** (Erişim Tarihi: Haziran 2014)
36. **http://www.nvidia.com.tr/object/product-geforce-gtx-560-ti-tr.html**(Erişim Tarihi: Nisan 2015)
37. **http://www.amd.com/en-us/products/graphics/desktop**(Erişim Tarihi: Nisan 2015)
38. **https://en.wikipedia.org/wiki/List_of_AMD_graphics_processing_units**(Erişim Tarihi: Nisan 2015)

39. **http://www.amd.com/en-us/products/processors/desktop/phenom-ii**(Erişim Tarihi: Nisan 2015)
40. **http://en.wikipedia.org/wiki/Parallel_computing** (Erişim Tarihi: Mart 2014)
41. **http://michaelgalloy.com/2013/06/11/cpu-vs-gpu-performance.html**(Erişim Tarihi: Mart 2015)
42. **http://en.wikipedia.org/wiki/DirectCompute** (Erişim Tarihi: Mart 2014)
43. **https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units**(Erişim Tarihi: Nisan 2015)

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı, Soyadı: Zülkif KORKMAZ

Uyruğu: Türkiye (T.C.)

Doğum Tarihi ve Yeri:1981, Sivas

Medeni Durumu: Evli

Tel: 0352-2248800

Email: zulkifkorkmaz@gmail.com

Adresi: Kayseri Abdullah Gül Üniversitesi, Sümer Kampusu, Kocasinan-Kayseri

EĞİTİM

Derece	Kurum	Mez.Yılı
Lisans	Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği, İstanbul	2006

İŞ DENEYİMLERİ

Yıl	Kurum	Görev
2012-Halen	Kayseri Abdullah Gül Üniversitesi	Uzman
2005-2012	Özel Sektör, İstanbul	Yazılımcı

TEZ İLE İLGİLİ PROJELERİ

1. Kurban, R., Korkmaz, Z., ERÜ BAP, FYL-2013-4798, “Grafik İşlemciler Üzerinde Çoklu-Odaklı Görüntülerin Paralel Programlama ile Birleştirilmesi”, 2013-2014, 6.000TL.