Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı 28, S. 1022-1027, Kasım 2021

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue 28, pp. 1022-1027, November 2021

Research Article

Çok Ölçekli Eğrilik Sınıflandırmasının CUDA ile Hızlandırılması

Mustafa Oğuz[1], Sercan Demirci[2*]

[1] Ondokuz Mayıs Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Samsun, Türkiye, (ORCID: 0000-0001-5168-3647), mustafa.oguz@bil.omu.edu.tr

[2*] Ondokuz Mayıs Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Samsun, Türkiye, (ORCID: 0000-0001-6739-7653), sercan.dermici@bil.omu.edu.tr

(1st International Conference on Applied Engineering and Natural Sciences ICAENS 2021, November 1-3, 2021)

(**DOI:** 10.31590/ejosat.1012445)

ATIF/REFERENCE: Oğuz, M. & Demirci, S. (2021). Çok Ölçekli Eğrilik Sınıflandırmasının CUDA ile Hızlandırılması. *Avrupa Bilim ve Teknoloji Dergisi*, (28), 1022-1027.

Öz

Günümüzde mekânsal ölçüm teknolojileri gelişerek büyümektedir. Yeni teknolojiler ile birlikte çok daha hızlı ve doğru ölçümler yapmak mümkün hale gelmiştir. Hata oranlarının azalması, yanında verinin yoğunluğunun artması sorununu getirmiştir. Daha yoğun bir veri her ne kadar daha doğru sonuç verse de işlem süreçlerinde dezavantaja neden olmuştur. Verilerin işlenme sürelerinde büyük artışlar meydana gelmiştir. Bunlarla birlikte günümüzde paralel programlama üzerine çalışmalar yapılmaktadır. Paralel programlama işlemci üzerinde yapılabileceği gibi ekran kartı üzerinde de yapmak mümkündür. Ekran kartları üzerinde paralel programlama yapmak için kütüphaneler mevcuttur. Bunlardan en popüleri çok iyi bilinen Nvidia CUDA kütüphanesidir. CUDA kütüphanesi ile CUDA çekirdekleri üzerinde paralel programlama yapmak mümkün hale gelmiştir. Yapılan çalışmada yer sınıflandırma algoritması üzerinde hızlanma elde edilmesi hedeflenmiştir. Yer sınıflandırma algoritması olan MCC algoritması CUDA çekirdekleri üzerine dağıtılmış ve paralel hesaplanması sağlanmıştır. Çalışma sonunda 21 kat hızlanma elde edilmiştir.

Anahtar Kelimeler: Filtreleme algoritmaları, Nokta bulutu, CUDA, Paralel programlama, İnce plaka spline, Çok ölçekli eğrilik sınıflandırması.

Accelerating Multiscale Curvature Classification with CUDA

Abstract

Today, spatial measurement technologies are growing and developing. With new technologies, it has become possible to make much faster and more accurate measurements. Along with the decrease in error rates, the increase in the density of the data has brought the problem. Although a denser data gives more accurate results, it has caused a disadvantage in the processing steps. There has been a great increase in the processing times of the data. Along with these, studies on parallel programming are carried out today. Parallel programming can be done on the processor as well as on the graphics card. Many libraries are available on the internet for parallel programming on graphics cards. The most popular of these is the well-known Nvidia CUDA library. Parallel programming on CUDA cores has become possible with the CUDA library. In the study, it is aimed to achieve acceleration on the location classification algorithm. The MCC algorithm, which is a location classification algorithm, is distributed on CUDA cores and its parallel calculation is provided. At the end of the study, 21 times acceleration was obtained.

Keywords: Filtering Algorithms, Point cloud, CUDA, Parallel programming, Thin-plate spline, Multiscale Curvature Algorithm.

[*] Sorumlu Yazar: sercan.demirci@bil.omu.edu.tr

1. Giriş

Teknolojinin gelişmesi ile birlikte mekansal ölçümlerde kullanılmak için farklı teknolojiler gelişmiştir. Gelişen teknolojiler arasında hiç şüphesiz ki en popüleri Lidar teknolojisidir. Bu teknoloji günümüzde en çok tercih edilen mekansal ölçüm teknolojisidir. 3 boyutlu olan neredeyse bütün nesneler Lidar ile taranabilmektedir. Lidar cihazına göre bu nesneleri büyüklükleri ve ölçülecek olan nesneye göre Lidar cihazının özellikleri değişiklik göstermektedir.

Lidar verilerinden Sayısal Yükseklik Modeli (SYM) çıkarımında filtreleme algoritmaları büyük önem arz etmektedir (Vosselman ve Maas, 2010). SYM filtreleme işlemlerini 4 farklı şekilde kategorize etmiştir. Bahsedilen algoritmalar; yüzey tabanlı filtreleme, morfolojik filtreleme, segmentasyon tabanlı filtreleme ve aşamalı düzensiz üçgen ağı (TIN) filtrelemeleridir.

Sınıflandırma algoritmaları arasında avantaj ve dezavantajlar bulunmaktadır. Sınıflandırma algoritmalarında genellikle yüzey tabanlı filtreleme algoritmaları kullanılmaktadır. Eğimi fark etmeksizin bir çok arazi koşulunda geçerli sonuçlar vermektedir. Buna karşılık arazinin çoğu detayı korunduğu için küçük nesnelerin sınıflandırılmasında hatalara neden olabilmektedir (Mongus ve Zalik, 2014).

Yüzey tabanlı sınıflandırma algoritmasının bu sorunlarına karşılık morfoloji tabanlı sınıflandırma algoritmaları geliştirilmiştir (Keqi vd., 2003). Morfolojik sınıflandırma algoritmaları, yüzey tabanlı sınıflandırma algoritmalarının aksine, arazinin morfolojik ayrıntılarının korunmasını sağlamaktadır. Bununla birlikte segmentasyon tabanlı algoritmalar, birbirinden farklı boyutlardaki ve şekillerdeki nesneleri algılamak için farklı özellikleri kullanmaktadır. Bu özelliklerinden dolayı bahsedilen algoritmalar şehir bölgelerinin sınıflandırılması için uygundur.

Tüm bunlara karşılık yukarıda ki algoritmalar dağınık lazer darbeleri yüzünden sık ormanlık alanlarda hatalara neden olabilmektedir (Chen vd., 2017). TIN algoritması, genel olarak doğruluk açısından diğer birçok sınıflandırma algoritmasından daha başarılı sonuçlar vermektedir (Sithole ve Vosselman, 2004). Ancak TIN algoritması dik yamaçlar gibi süreksiz arazilerin tespit edilmesinde zorlanmaktadır (Chen vd., 2016).

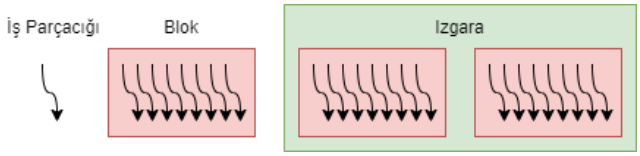
Lidar verileri ilk oluşturulduklarında ham veridirler. Verinin kullanılabilmesi için anlamlandırılması gerekmektedir. Raw veri çeşitli yöntemler kullanılarak sınıflandırılır. Bu sınıflandırma, yapılan çalışmaya göre değişiklik göstermektedir. Temelde yer ve yer olmayan olarak sınıflandırılabileceği gibi, yüksekliklerine göre ağaçlar, otlar, binalar ve özel bölgeler de sınıflandırılabilmektedir.

Lidar ölçümleri yerden ve havadan olmak üzere 2 ayrı kategoride ele alınabilmektedir. Yer Lidar taramaları genellikle bir binayı dışarıdan taramak için kullanılmaktadır. Ancak aynı şekilde binanın içine konumlandırılması halinde binanın içi de yüksek çözünürlükle taranabilmektedir. Lidar tekniği ve cihazı küçüldükçe binaya oranla daha küçük nesnelerde bu teknik ile taranabilmektedir. Yer Lidar ile aynı zamanda sokakları, Arkeolojik alanlar, yol-köprü gibi tarihi veya inşaat yapıları, maden sahaları, enerji nakil hatları da taranabilmektedir.

Diğer bir Lidar ölçüm şekli ise havadan ölçümdür. Bu ölçümde Lidar cihazı bir sabit kanat veya döner kanata monte edilmektedir. Ölçüm yapılacak alan üzerinde rota belirlenmektedir. Bu rotada hareket eden hava aracı ile birlikte Lidar cihazı alanın havadan 3 boyutlu verisini üretmektedir.

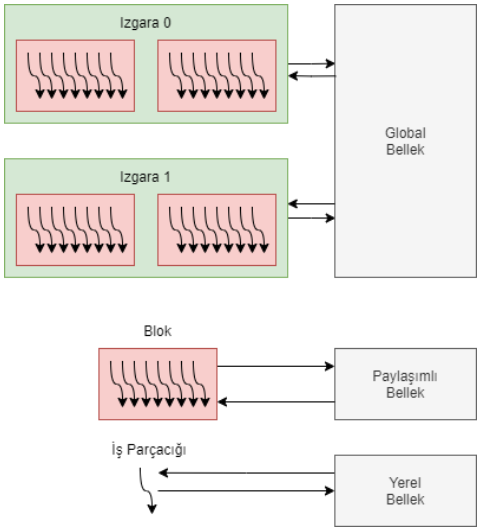
CUDA Nvidia tarafından geliştirilen paralel programlama kütüphanesidir. CUDA ile paralelleştirilmiş problem Nvidia ekran kartlarında hesaplanabilecek şekilde kodlanabilmektedir. Ama öncelikli olarak problemin paralelleştirmeye uygun olması gerekmektedir. Paralelleştirilen problem threadler arasında eş bir şekilde dağıtılmakta ve hesaplanması sağlanmaktadır. Bu sayede tek thread yapılan bir işleme göre kat be kat hızlanma elde edilmektedir.

CUDA ile programlama yapabilmek için hiyerarşik düzenlerini bilmek gerekmektedir. Thread ve Bellek hiyerarşisi olmak üzere 2 hiyerarşi bulunmaktadır. Öncelikle Thread hiyerarşisini ele almak gerekirse Şekil 1'de gösterildiği gibi genelden özele, öncelikle Izgara gelmektedir. Izgara'ları ise Blok'lar oluşturmaktadır. Blok'lar ise İş Parçacıkları ile oluşturulmaktadır.



Şekil 1. Thread Hiyerarşisi

Aynı şekilde bellek hiyerarşisinde Şekil 2'de gösterildiği gibi genelden özele, öncelikle Izgara'ya düşen Global Bellek gelmektedir. Izgara'ları oluşturan Blok'ların her birine ise Paylaşımlı Bellek'ler düşmektedir. Ve son olarak Blok'ları oluşturan her bir İş Parçacığına ise Yerel Bellek'ler düşmektedir.

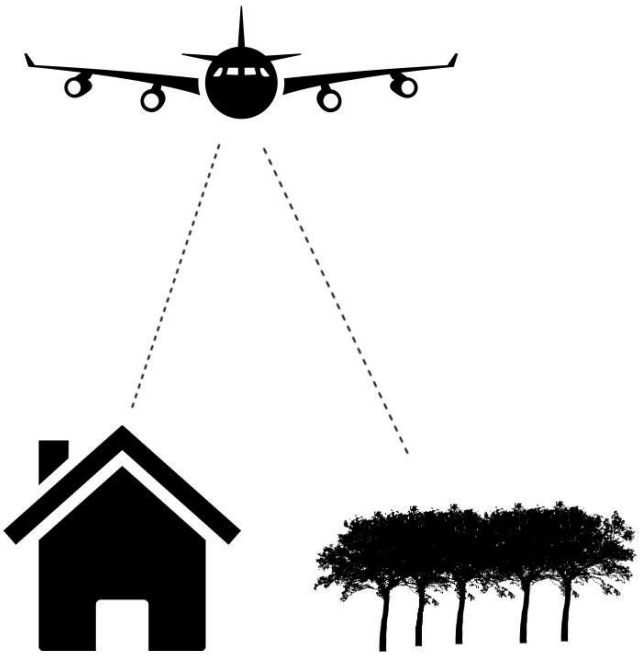


Şekil 2. Bellek Hiyerarşisi

2. Materyal ve Yöntem

Lidar cihazından çıkan lazerin bir nesneye çarpması sonucu o çarpan yerin koordinatları elde edilmektedir. Birden fazla lazer demeti bir yüzeye çarpması sonucunda ise birden fazla yerin koordinatı elde edilmektedir. Yüzeye çarpma sonucu oluşan her

bir koordinata ise nokta denilmektedir. Birden fazla nokta bir araya gelerek Nokta Bulutunu oluşturmaktadır (Otepka vd., 2013). Şekil 3'te hava lidar örneği verilmiştir.



Şekil 3. Hava Lidar

2.1. Materyal

2.1.1. Sınıflandırma

Lidar cihazından çıkan lazer demetlerinin oluşturduğu nokta bulutu haritaları ilk aşamada ham veridirler. 3 boyutlu olarak bir mekanı gösterebilmektedirler ancak daha fazla özellik taşımamaktadırlar. Bu verilerin anlamlandırılması için sınıflandırılması gerekmektedir. Sınıflandırma en temelde yer ve yer olmayan olarak yapılabileceği gibi daha spesifik şekilde bina, ağaç, otsu bitki vb. şekillerde de sınıflandırma yapılabilmektedir (Meng vd., 2019). Bu çalışmada nokta bulutu verileri yer ve yer olmayan şeklinde sınıflandırılacaktırlar.

2.1.2. İnce Plaka Spline

İnce plaka spline, verileri düzgünleştirmek için spline tabanlı bir interpolasyon tekniğidir. Bu yöntem sayesinde verilen tüm noktalardan geçen pürüzsüz bir yüzey elde edilmektedir. İnce plaka spline'ı fiziksel olarak metal bir levhanın bükülmesine benzetilebilmektedir. Metal levhanın bükülmeye karşı direnci olduğu gibi ince plaka spline'larının da bükülmeye karşı dirençleri bulunmaktadır. İnce plaka spline adını tam olarak ince metal levhadan almaktadır. İnce metal levhalara uygulanacak minimum kuvvete karşı istenilen şeklin elde edilmesi için kullanıldıklarından bu şekilde adlandırılmışlardır (Sibson ve Stone, 1991). Bu yöntemin sağladığı interpolasyon ile daha pürüzsüz bir yüzey elde etmek mümkün olmaktadır.

2.1.3. Çok Ölçekli Eğrilik Sınıflandırması

Çok ölçekli eğrilik sınıflandırmasının öncelikli amacı sınıflandırılmamış ham nokta bulutu verilerinin yer ve yer olmayan olmak üzere sınıflandırılmasıdır. Bunu mümkün olduğunca yüksek doğrulukla yapmayı amaçlamaktadır. Bu nedenle sınıflandırma yaparken ince plaka spline interpolasyonundan yararlanmaktadır.

Lidar verileri farklı formatlarda şifrelenebilmekte veya sıkıştırılabilmektedirler. Ticari kar amaçlı yazılımlar bunu kendi yazılımları dışında kullanılmaması için yapmaktadır. Çok ölçekli eğrilik sınıflandırmasının avantajı ise farklı Lidar formatlarının desteklenmesidir. Gereksinim olarak sadece x, y ve z eksenlerine ihtiyaç duymaktadır.

Bir diğer özelliği ise sınıflandırma işleminin tam dinamik olmasıdır. Diğer filtreleme ve sınıflandırma algoritmaları kullanıcının belli aşamalarda elle müdahale etmesine ihtiyaç duymaktadır. İşlem sırasında herhangi bir girdiye, düzenlemeye, elle müdahaleye gerek duymadan tüm işlem süreci otomatik bir şekilde ilerlemektedir (Evans ve Hudak, 2007). Bu çalışmada çok ölçekli eğrilik sınıflandırma algoritmasından faydalanılmıştır.

2.1.4. CUDA Mimarisi

GPU'lar CPU'lara oranla çok sayıda Arithmetic Logic Unit (ALU) barındırırlar. Aynı zamanda GPU'lar CPU'lara oranla içlerinde çok sayıda önbellek barındırırlar. Fakat bunun dezavantajı ise önbellek boyutlarının oldukça düşük olmasıdır. Önbellek sayısının fazla olması ve çekirdeklere yakın olması problemin çekirdeklere paylaştırılması sırasında belleklerden çok daha hızlı veri okunup yazılabileceği anlamına gelmektedir (Garland vd., 2008). Ancak aynı zamanda boyutlarının düşük olması nedeniyle işlenecek ve bellekte kullanılacak olan verilerin oldukça akıllı bir şekilde tahsis edilmesi gerekmektedir. GPU programlamanın en temel problemlerinden biri de budur.

Herşeyden önce problem paralelleştirilmektedir. Daha sonrasında paralelleştirilmiş problem için gerek duyulan iş parçacığı sayısı hesaplanmaktadır (Cook, 2012). Ardından problem bu iş parçacıklarına dağılmaktadır. Ve son olarak iş parçacıkları tarafından problem hesaplanmaktadır.

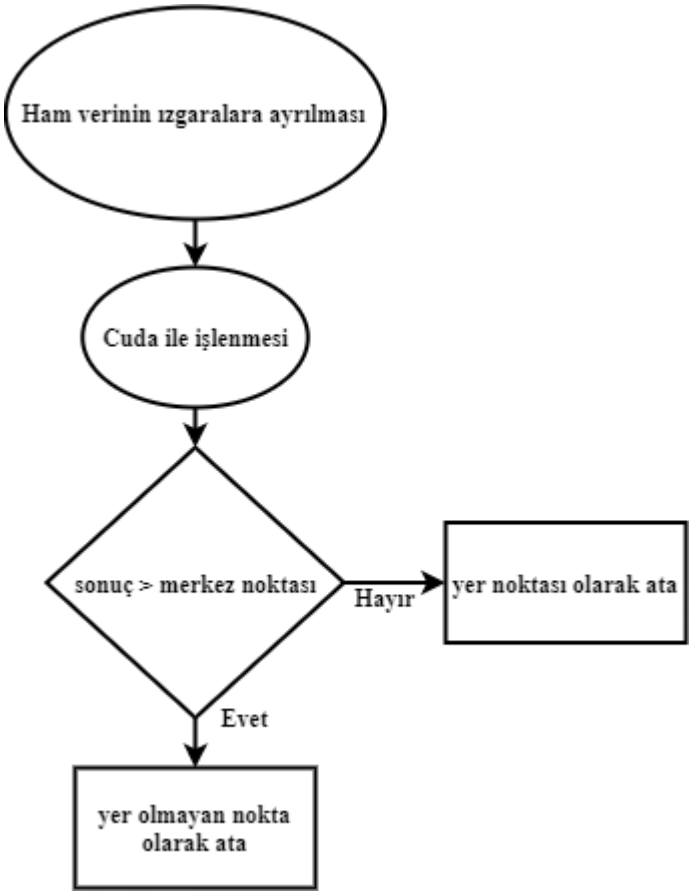
Tüm dağılımlar yapıldıktan ve tüm bellekler tahsis edildikten sonra artık GPU çalışmaya hazır bulunmaktadır. Bu noktada CPU'dan gelecek tek bir komut ile birden fazla iş parçacığı, neredeyse aynı anda, üstlerine düşen hesaplamalarını yapmaktadırlar.

GPU'lar birden fazla İş Parçacığı, Blok ve Izgara barındırmaktadırlar (Soyata, 2018). İş Parçacıkları bir araya gelerek blok denilen yapıyı oluşturmaktadırlar. Bloklar bir araya gelerek ise Izgara denilen yapıyı oluşturmaktadırlar.

Block'ların ve iş parçacıklarının kendilerine has eşsiz birer ID'si bulunmaktadır. Bu ID'ler threadId ve blockId olarak isimlendirilmektedirler. Burda dikkat edilmesi gereken yapının 3 boyutlu olmasıdır. Yani threadIdx.x, threadIdx.y, threadIdx.z ile 3 farklı boyutta iş parçacağına erişilebilmektedir. Aynı zamanda blockId içinde blockIdx.x, blockIdx.y ve blockIdx.z ile 3 farklı boyutta blok'a erişilebilmektedir (Cheng vd., 2014).

2.2. Yöntem: CUDA ile İşleme

Öncelikle .las formatındaki dosya okunmaktadır. Okunan dosya ızgaralara ayrılmaktadır. Izgaralara ayrılan veri CUDA ile paralelleştirilerek işlenmektedir. Sonucunda elde edilen sonuç verisi ızgaraların merkez noktaları ile karşılaştırılmaktadır. Karşılaştırma sonucunda Şekil 4'te gösterildiği gibi sonuç verisi merkez noktadan yüksek ise yer olmayan, düşük ise yer noktası olarak sınıflandırılmaktadır. Son olarak yeni .las dosyası kaydedilmektedir.



Şekil 4. Problem Akış Şeması

Proje kapsamında nokta bulutu verisinin sınıflandırılması hedeflenmiştir. Bu amaç doğrultusunda bahsedildiği üzere İnce plaka Spline'ı algoritması uygulanmıştır.

CPU ile çalışmada öncelikli olarak veri ızgaralara yani hücrelere ayrılmaktadır. Daha sonra bu hücrelerin merkez noktaları alınmaktadır. Ardından hücreler TPS algoritmasına girmektedir. TPS sonucu o hücrenin interpolasyon sonucu z verisi, yani yükseklik verisi hesaplanmaktadır. Ardından hücrenin merkez noktasının z değeri ile interpolasyon sonucu karşılaştırılmaktadır. Eğer sonuç hücrenin merkez noktasının z değerinden yüksekse o bölgedeki noktalar yer verisi olmayan olarak sınıflandırılmaktadırlar. Eğer düşükse yer verisi olarak sınıflandırılmaktadırlar. Bu şekilde doğrusal olarak bir döngü içinde bütün hücreler tek tek hesaplanmaktadır.

GPU ile çalışmada öncelikli olarak veri hücrelere ayrılmaktadır. Daha sonra bu hücreler 3 boyutlu matris haline getirilmektedir. Ardından 3 boyutlu matris GPU'ya allocate edilmekte ve kopyalanmaktadır. GPU içindeki TPS algoritması ile bu hücreler iş parçacıklarına dağılarak hesaplanmaktadır. Hesaplama sonuçları yine 3 boyutlu bir diziye aktarılarak tekrar RAM'e kopyalanmaktadır. Daha sonra CPU ile karşılaştırma işlemi yapılarak veri kaydedilmektedir. Şekil 5'te, CUDA ile işleme şeması ile akış detayları gösterilmiştir.

Şekil 5. CUDA ile İşleme Akışı

2.2.1. Verinin hazırlanması

Yukarıdaki genel anlatımda bahsedildiği gibi veri CPU'nun doğrusal çalışma mantığı gereği sırayla hücrelere ayrılmaktadır. Her bir hücre sırayla CPU'ya oturmaktadır. Proje kapsamında hücrelerin hepsine aynı anda erişebilmek için 3 boyutlu matris kullanılmıştır. Matrisin birinci boyutu hücreyi, ikinci boyutu hücre içindeki noktayı, üçüncü boyutu ise nokta içindeki x, y, z ve s değerlerini barındırmaktadır. S değeri ikinci matrisin boyutudur. GPU kodu içinde hazır bir fonksiyon kullanmak mümkün olmadığı için dizinin büyüklüğünü fonksiyon ile erişmekte mümkün değildir. Bundan dolayı üçüncü indis içinde fazladan bir veri ile bu görev gerçekleştirilmiştir.

2.2.2. Verinin GPU belleğine kopyalanması

Veri artık GPU'ya kopyalanmaya hazır hale gelmiştir. Ancak burada bir sorun vardır. GPU belleğinde yer tahsis etmek tek boyutlu matris ile mümkün olmaktadır. Yani tek boyutlu bir matrisi tahsis etmek için Cudamalloc komutu kullanılmaktadır. Ancak çalışmada kullanılan matris 3 boyutlu olduğu için farklı bir yaklaşım uygulanması gerekmiştir.

Öncelikli olarak noktalar tek tek tahsis edip kopyalanmıştır. Ardından bu tek boyutlu matrise ikinci matris oluşturup bağlanmıştır. Bu işlem cudaMemcpy fonksiyonu ile gerçekleştirilmiştir. Son olarak ikinci matrise üçüncü matris oluşturup bağlanmıştır. Sonuçta 3 boyutlu matrise tüm İş Parçacıkları tarafından aynı anda erişilebilir hale gelmiştir.

2.2.3. Verinin İşlenmesi

Kopyalanan veriyi işleyebilmek için öncelikle yeterli İş Parçacığı ve Blok sayısı bulunması gerekmektedir. Bunun için ufak bir hesap ile bunlar hesaplanmaktadır. Daha sonra ise artık

GPU kodu çalıştırılmaya hazır hale gelmektedir. GPU kodu çalıştırıldığı anda tüm İş Parçacıkları yaratılıp başlatılmaktadır. GPU fonksiyonu içinde iş parçacığı ID'si bulunması gerekmektedir. Bunun içinde yine ufak bir hesaplama yapılmaktadır. Ardından threadID ile matrisin birinci indisi ile hücrelere erişip 1 İş Parçacığı ile 1 hücrenin hesaplanmasını sağlanmaktadır.

2.2.4. Verinin Geri Kopyalanması

İşlenen veri fonksiyonun en sonunda 3 boyutlu bir matrise kopyalanmaktadır. Bütün İş Parçacıklarının hesaplaması bittikten sonra verinin tekrar RAM'e kopyalanması gerekmektedir. Bunun için benzer şekilde geri kopyalama işlemi gerçekleştirilmektedir. Artık CPU'ya dönen kod, gelen sonuç verisi ile merkezleri karşılaştırır ve sınıflandırmayı yapmaktadır. Ve son olarak sonuçlar kaydedilmektedir.

3. Testler

3.1. Sık Bölge Testi

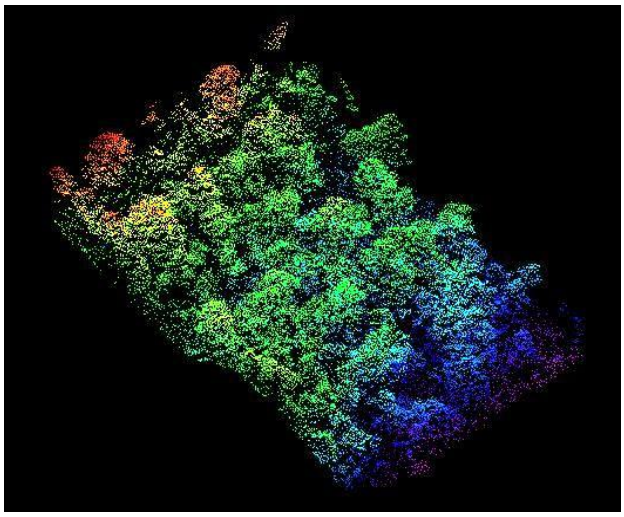
Bu testte uygulama sık ormanlık ve eğimli bir arazide test edilmiştir. Şekilde 6'da test edilen verinin ham hali gösterilmiştir. Şekil 7'de ise test edilen verinin filtrelenmiş hali gösterilmiştir. Şekil 8'de CPU ve GPU performansları karşılaştırılmıştır.

3.2. Seyrek Bölge Testi

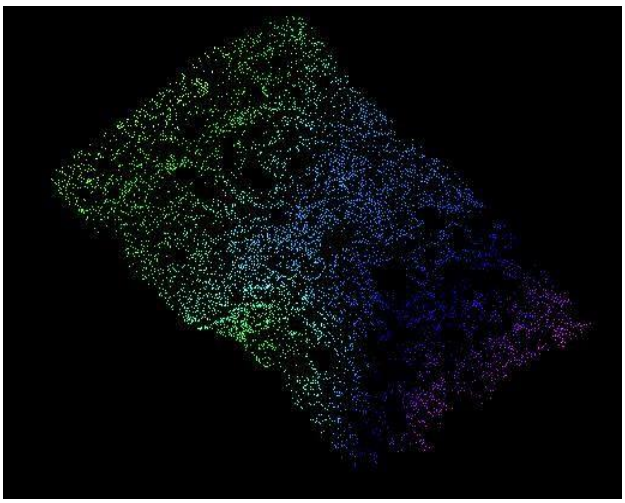
Bu testte ise uygulama daha az seyrek ve daha düz bir arazi üzerinde test edilmiştir. Şekil 9'da test edilen verinin ham hali gösterilmiştir. Şekil 10'da ise test edilen verinin filtrelenmiş hali gösterilmiştir. Şekil 11'de CPU ve GPU performansları karşılaştırılmıştır.

4. Sonuç

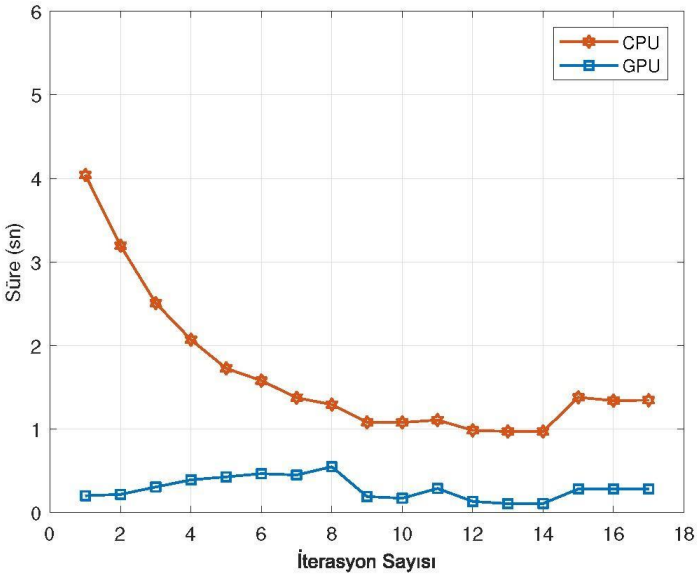
Bu çalışmada CPU üzerinde çalışmakta olan nokta bulutu sınıflandırma algoritması paralleştirilmiştir. Paralelleştirilen problemin CUDA üzerinde uygulaması yapılmıştır. 2 farklı nokta bulutu üzerinde test yapılmıştır. Birinciside daha sarp ve yoğun ağaçlıklı bir bölge seçilmiştir. İkincisinde ise daha düz ve seyrek bir arazi seçilmiştir. Bu 2 farklı nokta bulutu verisi üzerinde yapılan testler de görülmüştür ki CPU'ya oranla GPU üzerinde CUDA ile programlama ile 21 kat hızlanma elde edilmiştir. Sonuç verisinde CPU ile elde edilen sonuç ile GPU ile elde edilen sonuç arasında hiçbir fark veya kayıp yoktur. Tüm veriler sağlıklı bir şekilde korunmuştur. GPU, CPU ile aynı sonucu üretecek şekilde kodlanmıştır.



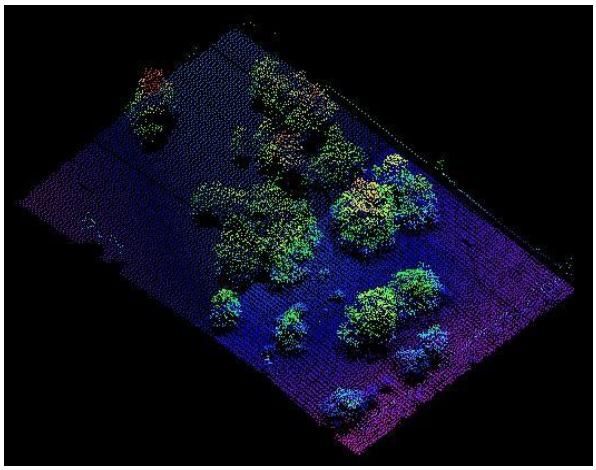
Şekil 6. Eğimli ve Sık Bölge



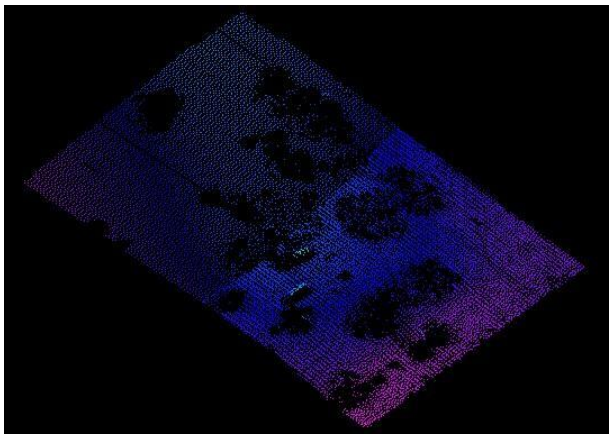
Şekil 7. Eğimli ve Sık Bölge Sonucu



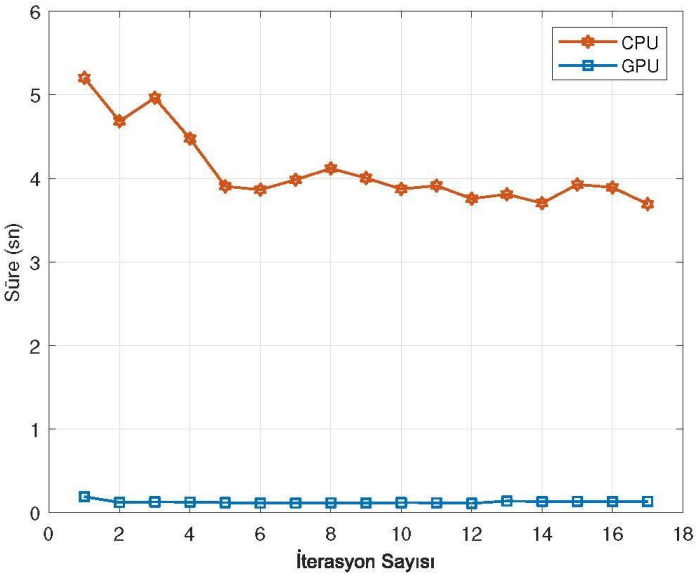
Şekil 8. Sık Bölge CPU ve GPU Performans Karşılaştırması



Şekil 9. Düz ve Seyrek Bölge



Şekil 10. Düz ve Eğimli Bölge Sonucu



Şekil 11. Seyrek Bölge CPU ve GPU Performans Karşılaştırması

Kaynakça

Chen, Q., Wang, H., Zhang, H., Sun, M., & Liu, X. (2016). A Point Cloud Filtering Approach to Generating DTMs for Steep Mountainous Areas and Adjacent Residential Areas. Remote Sensing, 8(1), 71. doi:10.3390/rs8010071

Chen Z, Gao B, Devereux B. State-of-the-Art: DTM Generation Using Airborne LIDAR Data. Sensors. 2017; 17(1):150. https://doi.org/10.3390/s17010150

Cheng, J., Grossman, M., & McKercher, T. (2014). Professional CUDA C Programming (1st ed.). Wrox.

Cook, S. (2012). CUDA Programming: A Developer's Guide to Parallel Computing with GPUs (Applications of Gpu Computing) (1st ed.). Morgan Kaufmann.

Garland, M., le Grand, S., Nickolls, J., Anderson, J., Hardwick, J., Morton, S., Phillips, E., Zhang, Y., & Volkov, V. (2008). Parallel Computing Experiences with CUDA. IEEE Micro, 28(4), 13–27. https://doi.org/10.1109/mm.2008.57

J. S. Evans and A. T. Hudak, "A Multiscale Curvature Algorithm for Classifying Discrete Return LiDAR in Forested Environments," in IEEE Transactions on Geoscience and Remote Sensing, vol. 45, no. 4, pp. 1029-1038, April 2007, doi: 10.1109/TGRS.2006.890412.

Keqi Zhang, Shu-Ching Chen, Whitman, D., Mei-Ling Shyu, Jianhua Yan, & Chengcui Zhang. (2003). A progressive morphological filter for removing nonground measurements from airborne LIDAR data. IEEE Transactions on Geoscience and Remote Sensing, 41(4), 872–882. https://doi.org/10.1109/tgrs.2003.810682

Meng, X., Lin, Y., Yan, L., Gao, X., Yao, Y., Wang, C., & Luo, S. (2019). Airborne LiDAR Point Cloud Filtering by a Multilevel Adaptive Filter Based on Morphological Reconstruction and Thin Plate Spline Interpolation. Electronics, 8(10), 1153. https://doi.org/10.3390/electronics8101153

Mongus, D., & Zalik, B. (2014). Computationally Efficient Method for the Generation of a Digital Terrain Model From Airborne LiDAR Data Using Connected Operators. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 7, 340-351.

Otepka, J., Ghuffar, S., Waldhauser, C., Hochreiter, R., & Pfeifer, N. (2013). Georeferenced Point Clouds: A Survey of Features and Point Cloud Management. ISPRS International Journal of Geo-Information, 2(4), 1038–1065. https://doi.org/10.3390/ijgi2041038

Sibson, R., & Stone, G. (1991). Computation of Thin-Plate Splines. SIAM Journal on Scientific and Statistical Computing, 12(6), 1304–1313. https://doi.org/10.1137/0912070

Sithole, G., & Vosselman, G. (2004). Experimental comparison of filter algorithms for bare-Earth extraction from airborne laser scanning point clouds. ISPRS Journal of Photogrammetry and Remote Sensing, 59(1–2),85–101. https://doi.org/10.1016/j.isprsjprs.2004.05.004

Soyata, T. (2018). GPU Parallel Program Development Using CUDA (Chapman & Hall/CRC Computational Science) (1st ed.). Chapman and Hall/CRC.

Vosselman, G., & Maas, H.-G. (2010). Airborne and terrestrial laser scanning. Dunbeath, Scotland: Whittles.