KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HARİTA MÜHENDİSLİĞİ ANABİLİM DALI

HAVA LiDAR VERİLERİNDEN REZERVUAR ALANININ OTOMATİK ÇIKARIMI: KESTEL BARAJI ÖRNEĞİ

YÜKSEK LİSANS TEZİ

Tolga ODABAŞ

MAYIS-2018
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HARİTA MÜHENDİSLİĞİ ANABİLİM DALI

HAVA LiDAR VERİLERİNDEN REZERVUAR ALANININ OTOMATİK ÇIKARIMI: KESTEL BARAJI ÖRNEĞİ

Tolga ODABAŞ

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"HARİTA YÜKSEK MÜHENDİSİ"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 10 /05/2018
Tezin Savunma Tarihi : 31/05/2018

Tez Danışmanı : Prof . Dr. Fevzi KARSLI

Trabzon 2018

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Harita Mühendisliği Ana Bilim Dalı
Tolga ODABAŞ Tarafından Hazırlanan

HAVA LiDAR VERİLERİNDEN REZERVUAR ALANININ OTOMATİK ÇIKARIMI: KESTEL BARAJI ÖRNEĞİ

başlıklı bu çalışma, Enstitü Yönetim Kurulunun / / gün ve sayılı kararıyla oluşturulan jüri tarafından yapılan sınavda
YÜKSEK LİSANS TEZİ
olarak kabul edilmiştir.

Jüri Üyeleri

Başkan : Prof. Dr. Fevzi KARSLI …...……...……

Üye : Doç. Dr. Faik Ahmet SESLİ …...……...……

Üye : Dr. Öğretim Üyesi Mustafa DİHKAN ……...……...……

Prof. Dr. Sadettin KORKMAZ
Enstitü Müdürü

TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum “Hava LiDAR Verilerinden Rezervuar Alanının otomatik Çıkarımı: Kestel Barajı Örneği” başlıklı bu çalışmayı baştan sona kadar danışmanım Prof. Dr. Fevzi KARSLI‘nın sorumluluğunda tamamladığımı, verileri kendim topladığımı, analizleri ilgili programlarda yaptığımı başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 31/05/2018

Tolga ODABAŞ

ÖNSÖZ

“Hava LiDAR Verilerinden Rezervuar Alanının Otomatik Çıkarımı: Kestel Barajı Örneği” başlıklı bu çalışma, Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Harita Mühendisliği Anabilim Dalında, yüksek lisans tezi olarak gerçekleştirilmiştir.

Eğitim hayatımın her evresinde destek olan tüm öğretmenlerime, Karadeniz Teknik Üniversitesi’ndeki lisansüstü öğrenimim boyunca bilgi, tecrübe ve zamanını esirgemeyen ayrıca bu günlere ulaşmamda katkı sağlayan saygıdeğer tez danışmanım Prof. Dr. Fevzi KARSLI’ya teşekkürlerimi borç bilirim. Bilimsel anlamda bilgilerini paylaşarak çoğaltmayı ve büyütmeyi temel edinmiş, Artvin Çoruh Üniversitesi, Harita Mühendisliği Bölümü öğretim üyelerinden Doç. Dr. Halil AKINCI’ya da yardımlarından dolayı çok teşekkür ederim.

Tüm hayatım boyunca maddi ve manevi desteklerini esirgemeyen aileme, zor zamanlarda her zaman yanımda olan sevgi ve saygı dolu eşime, her an büyük içtenlikle yardımıma koşan dostlarıma sonsuz teşekkürlerimi ve şükranlarımı sunarım.

Tolga ODABAŞ

Trabzon 2018

İÇİNDEKİLER

Sayfa No

Sayfa No

Yüksek Lisans

ÖZET

HAVA LiDAR VERİLERİNDEN REZERVUAR ALANININ OTOMATİK ÇIKARIMI: KESTEL BARAJI ÖRNEĞİ

Tolga ODABAŞ

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Harita Mühendisliği Anabilim Dalı
Danışman: Prof. Dr. Fevzi KARSLI
2018, 77 Sayfa, 96 Ek Sayfa (cd)

Lazer darbeleri kullanılarak bir nesne veya yüzeyin uzaklığını ölçmeye yarayan hava lazer tarama teknolojisi (LiDAR), sağlamış olduğu verilerle harita mühendisliği disiplini açısından büyük önem kazanmıştır. Bu teknoloji sayesinde elde edilen üç boyutlu (3B) nokta bulutundan istenilen sonuçları elde edebilmek ya da harita ürünü olarak değerlendirilebilecek objeleri otomatik olarak çıkarmak için LAStools, FugroViewer ve MATLAB gibi farklı yazılım ve programlama dillerinden yararlanılmaktadır. Bu çalışmada, İzmir ili, Bergama ilçesi yakınında Kestel Çayı üzerine kurulmuş Kestel Barajı rezervuar sınırlarının aynı alanı içeren ve iki farklı LiDAR sensörü ile üretilmiş hava LiDAR verileri kullanılarak çıkarılması amaçlanmıştır. 3B nokta bulutu verilerinin çok yoğun hacimde olması ve işlenmesi noktasında yüksek konfigürasyon gerektirmesi sebebiyle, söz konusu LiDAR verisi yani nokta bulutu içerisinden öncelikle çalışma alanı olan Kestel Barajı bölgesi, diğer noktalardan ayıklanarak veri yoğunluğu azaltılmıştır. Baraj bölgesini içeren iki farklı sensörle alınmış nokta bulutu içerisinden baraj rezervuar alanı ve sınırları MATLAB programlama dili ortamında üretilen bir yazılımla otomatik olarak çıkarılmıştır. Böylece LiDAR verilerinin üretildiği tarih itibariyle Baraj rezervuar alanı sınırları elde edilmiş, bu sınırlar LiDAR verisinin elde edildiği tarihe yakın bir zamanda üretilmiş bölgeye ait hâlihazır harita üzerinden çıkarılan Kestel Barajı rezervuar alan sınırı referans verisi ile karşılaştırılmıştır. ArcGIS yazılımı ile gerçekleştirilen bu karşılaştırma sonrası LiDAR verilerinden üretilen baraj rezervuar sınırı ile referans veriden (Hâlihazır Harita) alınan sınırların büyük oranda (%95-%99 arasında) çakıştığı tespit edilmiştir. Bu çalışma ile LiDAR verileri kullanılarak baraj rezervuar alanı gibi su ile çevrili sınırların yüksek doğrulukla otomatik olarak hızlı ve güvenilir bir şekilde çıkarılabileceği, ayrıca LiDAR verilerinden yüksek doğrulukta planimetrik harita bilgilerinin de ortaya çıkabileceği sonucu ortaya konmuştur.

Anahtar Kelimeler: *LiDAR, Baraj Sınır Çıkarımı, MATLAB, ArcGIS*

Master Thesis

SUMMARY

AUTOMATIC REMOVAL OF RESERVOIR FIELD FROM AIRCRAFT LIDAR DATA: EXAMPLE OF KESTEL DAM

Tolga ODABAŞ

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Geomatics Engineering Graduate Program
Supervisor: Prof. Dr. Fevzi KARSLI
2018, 77 Pages, 96 Appendix Pages (cd)

Air laser scanning technology (LiDAR), which is used to measure the distance of an object or surface using laser pulses, has gained great importance in terms of discipline in map engineering. Different software and programming languages such as Lastools, FugroViewer and Matlab are used to obtain the desired results from the three-dimensional (3D) point cloud obtained by this technology or to automatically extract objects that can be evaluated as a map product. In this study, Kestel Dam, which is located on the Kestel Stream in the vicinity of Bergama county, Izmir, is intended to be extracted using two different LiDAR sensor generated air LiDAR containing the same area of the reservoir borders. Since the 3D point cloud data is in a very dense volume and requires high configuration at the processing point, the data density is reduced by the LiDAR data, ie the point cloud, by first extracting the working area Kestel Dam region from other points. The dam reservoir area and its boundaries were automatically extracted from the point cloud with two different sensors including the dam area by software produced in the Matlab programming language environment. Thus, as of the date of the production of the LiDAR data, the dam reservoir area boundaries have been obtained, which are compared with the Kestel Dam reservoir area reference datum, which is derived from the current map of the area produced near the time of obtaining the LiDAR data. This comparison with ArcGIS software found that the limit of the dam reservoir generated from the LiDAR data exceeded the boundaries of the reference map (from the current map) in large amounts (between %95 and %99). Using this data, it is shown that water-enclosed boundaries such as dam reservoir area can be automatically extracted quickly and reliably with high accuracy, and also high accuracy planimetric map information can be obtained from LiDAR data.

Key Words: LiDAR, Dam Boundary Inference, MATLAB, ArcGIS

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

KISALTMALAR DİZİNİ

BARKOK : Bilimsel Araştırma ve Koordinasyon Komisyonu

BHİKPK : Bakanlıklararası Harita İşlerini Koordinasyon ve Planlama Kurulu

GPS : Global Positioning System

HGK : Harita Genel Komutanlığı

HLS : Horizontal Line Sampling

IMU : Inertial Measurement Unit

INS / IMS : Inertial Navigation System / Institute for Microelectronic System

LiDAR : Light Detection And Ranging

MATLAB : Matrix Laboratory

RADAR : Radio Detection And Ranging

RANSAC : Random Sample Consensus

SYM : Sayısal Yükseklik Modeli

TLS : Terrestrial Laser Scanning

3B : Üç Boyutlu

1. GENEL BİLGİLER

1.1. Giriş

Türkiye, üç tarafı denizlerle çevrili ender ülkelerden biri olduğu gibi aynı zamanda gölleri, barajları, akarsuları ve kaynak sularıyla da bölgesinde önemli bir konumdadır. Ülkemizin yaklaşık 8300 km uzunluğunda kıyısı ve yaklaşık 178.000 km uzunluğunda akarsuyu, 906.118 ha doğal gölalanı ve 342.377 ha baraj gölalanı bulunmaktadır (Kalkan, 2005).

Türkiye için yıllık ortalama yağış yaklaşık 501 milyar m^3 olarak verilmektedir (Turfan, 2002; URL1). Bu suyun yarıdan fazlası (274 milyar m^3) buharlaşarak tekrar atmosfere dönmekte, 69 milyar m^3'lük kısmı yeraltı suyunu beslemekte ve geriye kalan 158 milyar m^3'lük kısmı ise akışa geçerek çeşitli büyüklükteki akarsular ile deniz veya göllere boşalmaktadır. Ayrıca, komşu ülkelerden gelen 7 milyar m^3 ile tekrar yer yüzeyine dönen 28 milyar m^3 suyun ilavesi ile bu miktar 193 milyar m^3'e ulaşmaktadır. Türkiye'nin toplam yenilenebilir su potansiyeli 234 milyar m^3'tür. Buna göre, ülkemiz "su zengini" olmayıp "su azlığı" çeken ve bu nüfus artışıyla pek yakın gelecekte "su fakiri" konumuna düşebilecek bir ülke durumundadır (Kalkan, 2005).

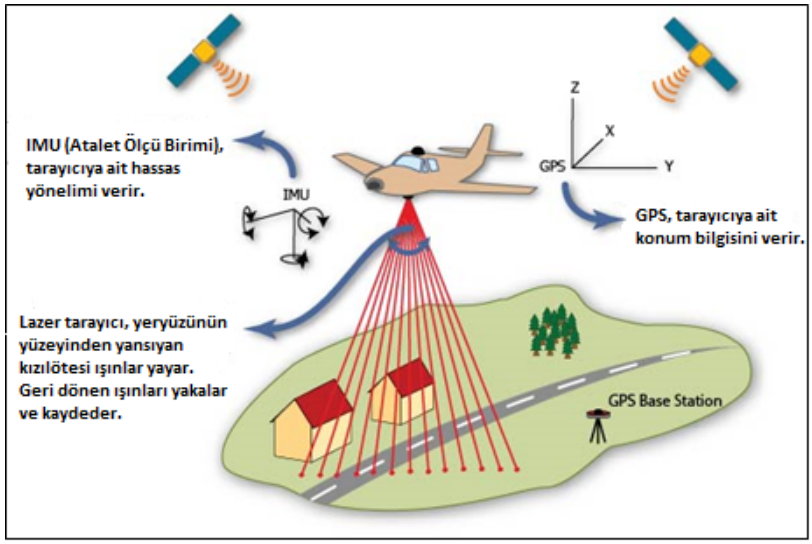
Günümüzde küresel ısınma sonucu meydana gelen kuraklık etkisi göllerde su seviyelerini önemli oranlarda etkileyebilmektedir. Bu yüzden karasal ekosistem üzerinde çok önemli bir yere sahip olan göllerde meydana gelen zamana bağlı değişimler sürekli olarak izlenmeli ve değişimler tespit edilerek gerekli önlemler alınmalıdır. Çeşitli hava araçları üzerine monte edilmiş LiDAR (Light Detection And Ranging) alıcılar kullanılarak istenilen zamanda her hangi bir bölgeye ilişkin doğrudan ve hızlı biçimde üç boyutlu (3B) veri toplanabilmektedir (Canaz vd, 2015).

Hava LiDAR sistemleri bir hava aracının altına monte edilen tarayıcıdan gönderilen lazer ışının gidiş-dönüş süresini kaydederek yer objeleri ve algılayıcı arasındaki mesafeyi hesaplayan aktif bir ölçme sistemidir. Sistem dâhilinde bulunan GPS ve INS/IMU sistemlerinden alınan konum ve dönüklük verisi ile beraber taranan objeye ait konum hesaplanır. Bu şekilde yeryüzüne binlerce lazer ışını gönderebilen sistem, fiziksel yeryüzüne ait yüksek yoğunluklu ve referanslandırılmış 3B veri doğrudan elde edilmektedir.

Kıyı bölgeleri doğa olayları ve insan etkileri açısından çevresel olarak hassas yerlerdir. Bu bölgelerdeki arazi ile su etkileşimi zamanla değişmekte, bu değişimin algılanması ve izlenmesi gerekmektedir. Su yüzeyinin tespiti uzaktan algılama teknikleri kullanılarak kolaylıkla tespit edilebilecek bir çalışma değildir fakat LiDAR teknolojisinin multispectral (çok bantlı) kullanımı sayesinde farklı dalga boylarına sahip lazer ışınlarıyla yapılacak olan uygulamalar bu tespiti kolaylaştırmaktadır.

LiDAR sistemleri de radar gibi aktif uzaktan algılama sistemleridir. RADAR'dan farklı olarak mikro dalga yerine, lazer sinyallerini kısa elektromanyetik dalgalar halinde yayarak veri elde etmektedir. Bu sistemlerle yeryüzüne saniyede binlerce sinyal gönderilmekte ve GPS/INS teknolojisi ile entegre çalışarak 3B veri doğrudan elde edilmektedir. LiDAR, yüksek yoğunluklu ve geometrik özellikli sayısal yükseklik verisini, yersel ölçmeler ile yaklaşık aynı doğrulukta, bununla birlikte sayısal hava fotogrametrisinden daha hızlı şekilde oluşturmaya yatkın bir teknolojidir.

Ayrıca bu sistemle, daha fazla otomasyon, hava ve ışıktan bağımsızlık, yer noktalarında daha az kontrol imkânı ve verinin dijital ortamda elde edilmesi sağlanmaktadır. LiDAR, yersel ölçme yöntemleri ve sayısal hava fotogrametrisi ile karşılaştırıldığında daha az arazi çalışması ve değerlendirme maliyeti gerektirir. Bu durum, düşük maliyetli, nokta yoğunluğu yüksek ve istenilen doğrulukla referanslandırılmış sayısal yükseklik verisine ihtiyaç duyan kullanıcılar için, LiDAR'ı çekici bir teknoloji yapmıştır. Burada, referanslandırma terimi sayısal bir görüntünün geometrik düzeltmesinin yapılması ve bir projeksiyon sistemine oturtulması anlamında kullanılmaktadır**.** (Ekercin ve Üstün, 2004)



Şekil 1. LiDAR Sistem Bileşenleri (URL 2)

Birçok ölçme uygulamalarında uçaktan lazerle tarama teknolojisi, diğer bilinen algılayıcıları içeren dijital kameralar, çok spektrumlu (hiperspektral) tarayıcılar ve termal kameralarla birlikte kullanılmaktadır.

Spektral özellikler açısından lazerler, görünür ve yakın kızılötesinden, 50-30000 nm gibi çok daha geniş spektrumlarda var olmaktadırlar. Ancak LiDAR'da kullanılanlar yakın kızılötesi bölgeye kadar sınırlıdır. Helikopter veya uçağa monte edilerek kullanılan LiDAR sistemleri; gölgede, gece gündüz veya bulutlar arasında veri toplayabilmek için kızılötesine en yakın elektromanyetik ışık spektrumu kullanır (Wehr ve Lohr, 1999).

LiDAR verilerinde, yükseklik bilgilerinin mutlak doğruluğu 15 cm, bağıl doğruluğu 5 cm'den daha az olabilir. Planimetrik (X ve Y) verilerin mutlak doğruluğu uçuş yüksekliğine bağlı olmakla birlikte genellikle 10 cm ile 1 m arasındadır. Yükseklik bilgileri, saniyede 1000 nokta alacak şekilde oluşturulur ve sonuçtaki nokta yoğunluğu, yersel ve fotogrametrik ölçme yöntemleriyle elde edilenden çok daha fazladır (Ekercin ve Üstün, 2004).

Yükseklik bilgisini içeren modeller jeoloji, hidroloji, harita mühendisliği, şehircilik, risk yönetimi ve askeri uygulamalar gibi birçok alanda kullanılmaktadır. Laser Ranging, Laser Altrimetry, Laser Scanning ve Laser Detection and Ranging olarak da bilinen sistem, Radar benzeri bir mantıkla çalışmaktadır fakat radar sistemlerinden farklı olarak radyo dalgaları yerine lazer ışınları kullanarak gerekli yükseklik bilgisini elde eder (Jiang vd., 2014).

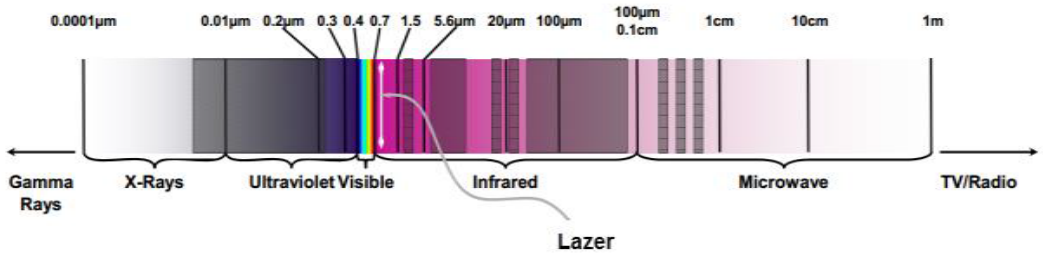
1960'ların sonlarında geliştirilen sistemin ilk ticari LiDAR haritalama sistemi 1993'te geliştirilerek topografik harita üretiminde kullanılmıştır (Liadsky, 2007; NOAA, 2012). Lazer tarayıcı sistemleri kullanılarak SYM (Sayısal Yükseklik Modeli) üretimi metodu üzerine tartışmalar ilk kez 1994 yılında Almanya'da Ölçme ve Haritalama Birimi (Surveying and Mapping Agency-SMA) tarafından başlatıldı (Petzold vd., 1999). Bu konuda tartışmalar sürerken 1996 yılından sonra birçok şirket tarafından çeşitli amaçlarda kullanılmak üzere üretilerek haritalar için LiDAR sistemleri oluşturulmaya başlandı (Jiang vd., 2014). Son yıllarda LiDAR teknolojisinin hızla gelişmesini sağlayan birçok yatırım ve araştırma da yapılmıştır.

Aktif uzaktan algılama teknolojisi olan bu sistem yüksek enerjili ve çok dar bir ışık dalgası üreterek hedefe gönderir ve dönen dalgayı kaydeder. Gece ve gündüz kullanılabilir fakat güneş ışığı etkilerinden korunmak için gece çalışması daha verimlidir. Piyasada aktif

satışı yapılan lazer tarayıcılara bakıldığında kullandıkları dalga boyu 810 nm ile 1540 nm arasında değişmektedir. (Morsy vd., 2018)

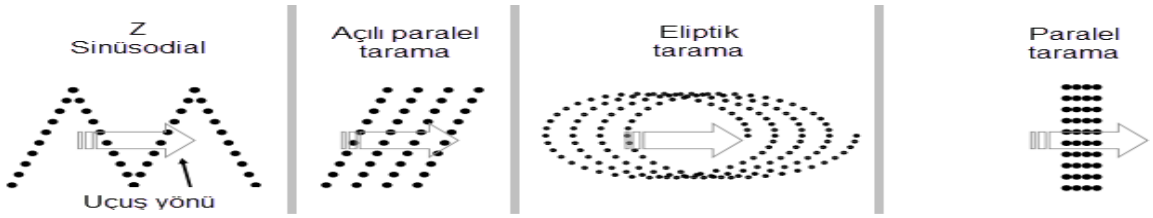
LiDAR sistemleri genellikle nesnelerden yansıyan enerjinin (yoğunluk) aralığını ve gücünü ölçen tek renkli bir dalga boyunda çalışır. Son zamanlarda, farklı dalga boylarında veri elde eden multispektral LiDAR sensörleri ortaya çıkmıştır. Multispektral karasal lazer taraması (Terrestrial Laser Scanning (TLS)) gelişmekte olan bir teknolojidir. Çift veya üç dalga boyuna sahip hava lazer tarayıcılarını kullanmaktadır. Hem spektral bilgileri hem de 3B koordinat verilerini eş zamanlı olarak tespit ettiği için birçok uygulamada sınıflandırma işlemi için multispektral LiDAR tercih edilmiştir.

Gelişen havadan lazer tarama sistemleriyle birlikte, su altı (batimetrik) ölçümlere de kolaylık sağlanmıştır. Yakın kızıl ötesi (1064nm ve 1550nm) gibi tek renk dalga boyuna sahip LiDAR ölçümlerinde su yüzeyine çarpan lazerler soğrulduğu için veri temini sağlanamazdı. Bu yüzden yeşil ve mavi (532nm) ışık rengine karşılık gelen frekansta lazer kullanılarak suyun altındaki topoğrafyaya da ulaşım sağlandı. Her iki dalga boyuna sahip sensörler yardımıyla su altı ölçümlerin gerçekleşmesine olanak sağlanmıştır.



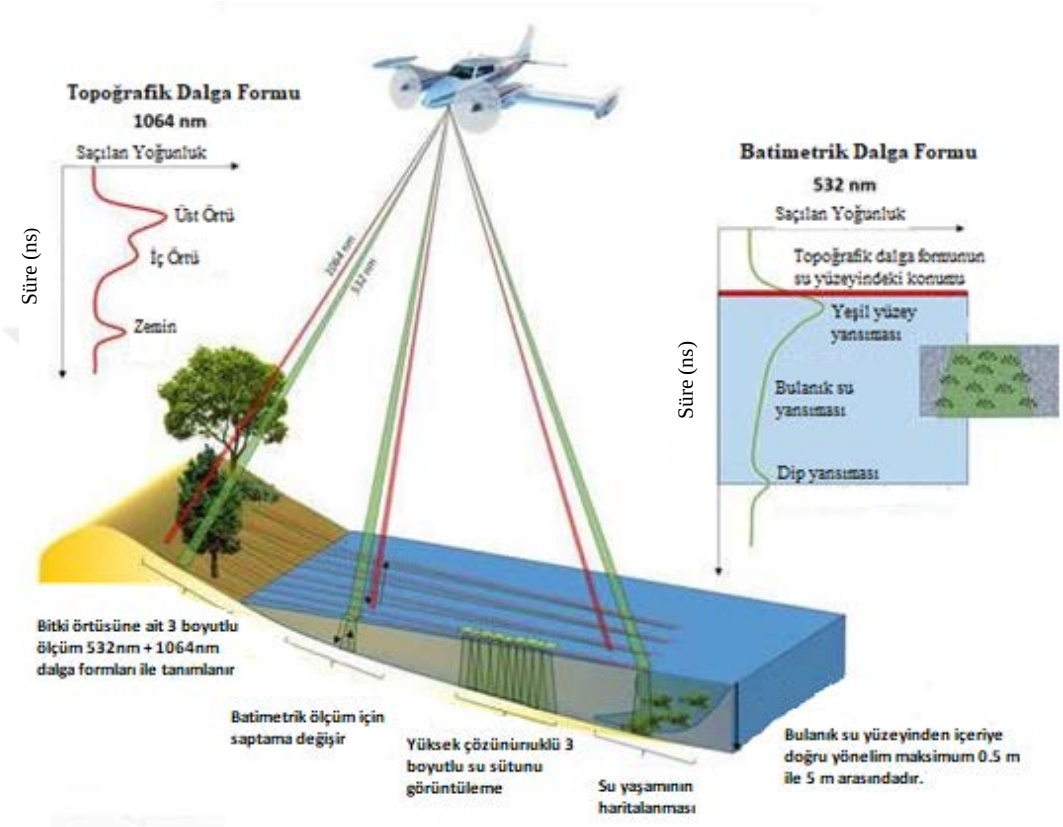
Şekil 2. Lazerin elektromanyetik spektrumdaki yeri (Polat ve Uysal,2016)

Tarayıcıların veri toplama şeklide farklılık göstermektedir. Şekil 3' de Z şekli, paralel, eğik paralel ve eliptik olmak üzere farklı tarama çeşitleri gösterilmektedir.



Şekil 3. Lazer tarayıcılara bağlı olarak değişen farklı tarama çeşitleri (Polat ve Uysal,2016)

Lazerin yerdeki ayak izi yaklaşık olarak bir çemberdir ve bunun yarıçapı lazerin tarama açısına, uçuş yüksekliğine, uçağın hızına ve topografyaya bağlıdır. Örneğin 1000 m yüksekten çıkan 0.1 mrad' lık lazer ışının t uçuş süresi 6.7 µs ve yerdeki karşılığı yaklaşık olarak 10 cm'ye karşılık gelmektedir. Tek bir lazer pulsu, tek veya çoklu şekilde dönebilir.



Şekil 4. Lazer ışınının yer yüzeyine ve su altına nüfuz etmesi (URL2)

Yaklaşık 20 yıl önce ticari olarak kullanılmaya başlanan ve 2000'li yılların başından itibaren ABD ve Avrupa'da çok farklı uygulama alanlarında yoğun olarak kullanılan Havadan LiDAR Sistemleri (HLS), son yıllarda ülkemizde pek çok kurum, kuruluş ve kişinin ilgisini çekmeye başlamıştır. Kısa zaman içerisinde, oldukça yoğun ve yüksek doğrulukla 3B nokta bulutu üretme, ağaç üst bitki örtüsünün altındaki nesneleri ve zemini tespit edebilme yeteneklerine sahip HLS ile topoğrafyanın ve nesnelerin 3B sayısal gösterimleri hassas ve doğru olarak elde edilebilmektedir. LiDAR'ın en büyük avantajı, nokta koordinatlarının yanı sıra sinyalin geri yansıdığı yüzeyin fiziksel özelliklerine (dönen sinyalin genişliği ve şiddeti) ait bilgiler içermesi ve bazı nesnelerden birden fazla sayıda geri dönüş elde edebilmesidir (Şehsuvaroğlu vd, 2015).

1.2. Problemin Tanımı

Göllerin karasal ekosistem üzerinde çok önemli etkisi bulunmaktadır. Göller, sulama, tatlı su kaynağı, akuatik yaşam alanı oluşturma gibi faaliyetler açısından çok önemlidir. Günümüzde ise küresel ısınmadan dolayı birçok su kaynağı gibi, göller de kurumaya başlamıştır. Ekosistemin düzenli bir şekilde devamı için göllerin yüzey alanındaki değişimlerini gözlemek, buna bağlı önlemler almak gerekir. Bu sebeple göllerin zamanla değişimini izlemek şarttır (Mostafa ve Soussa, 2006).

Ülkemiz, binlerce kilometre uzunluğunda denize kıyısı, irili ufaklı çok sayıda göl, gölet ve baraja sahip olan bir ülkedir. Bununla birlikte, yarı kurak iklim kuşağında yer almamız ve artan nüfus ve yükselen hayat standardına bağlı olarak su tüketiminin artması ve kirleticilerin çoğalması mevcut su kaynaklarımızın daha etkin ve daha dikkatli kullanımını gerekli kılmaktadır. Sahip olduğumuz bu servetin kıymetini bilmek ve en verimli şekilde kullanmak, en fazla istifadeyi sağlamak temel amaçtır. Bunun için, kıyılarımızın korunması, kirliliğin önlenmesi, kaynakların yeterince kontrol altına alınması ve boşa akıp giden suların mümkün olduğunca baraj/gölet gibi ortamlarda biriktirilmesi ve bunların var olan enerjilerinin kontrollü ve faydalı bir şekilde açığa çıkarılması önemli konulardandır. (Kalkan ve Alkan,2005).

Baraj, göl ve kıyılarımızdaki değişen su seviyelerini izlemek ve yorumlayabilmek adına analiz ve tespitlerin yapılması bu tez çalışmasının çıkış noktası olmuştur. Bu amaç doğrultusunda "LiDAR verilerinin kullanımının ne kadar sağlıklı olacağı ?" sorusu akla gelmektedir. Tezde ana hedef bu sorunun cevabını Kestel Barajı örneğinde ele alarak sonuçlarıyla beraber ortaya koymaktır.

Kullanılan LiDAR verisinin özellikleri de ulaşılmak istenen sonuç için bilinmesi gereken bir faktördür. Bu çalışmada kullanılan LiDAR verisi kızılötesi (kırmızı görünür ışığın en uzun dalga boyuna sahip rengidir, kızılötesi ışınımın dalga boyu 750 nm ile 1 μm arasındadır) ışınlar ile elde edilen nokta bulutundan oluşmaktadır. Bu şekilde elde edildiği için göl yüzeyini temsil eden noktalardan yararlanılamamaktadır. Eğer kullanılan ışının dalga boyu daha küçük olsaydı mor veya mavi (400-490 nm) gibi elde edilen nokta bulutu daha farklı olacak ve uygulanacak metot buna göre şekil alabilecektir.

Literatüre bakıldığında benzer çalışmalarda iki farklı yoldan çözüme ulaşıldığı görülmüştür. Birincisi, çalışma alanına ait görüntülerle ya da görüntü verisi elde ederek, ikili görüntü (binary image) yaparak, su yüzeyini ve karayı ayırt etmek için piksel değerlerinden faydalanarak kıyı çizgisini belirlemeye çalışmışlardır. Tercih edilen diğer yol ise, çalışma alanına ait sayısal arazi modeli elde edilip, su yüzeyinin sahip olduğu yükseklik değeri sabit olacağından ve karaya geçiş yaparken yükselti farkının bir anda değişmesinden kaynaklı özellikleri kullanarak kıyı çizgisini belirlemeye çalışmışlardır. Bu çalışmada ise, salt nokta kümesi içerisinden Kestel Barajına ait rezervuar alanı tespit edilmiştir. Bölgeye ait herhangi bir görüntü veya sayısal arazi modelinden yardım alınmamıştır. Sadece nokta bazlı bir çalışmanın söz konusu rezervuar alanını ve çevresini belirleyebilmede yeterli olup, olamayacağı sorusuna cevap aranmıştır. Aynı zamanda matlab programlama dili ile bu işlemi otomatik bir hale nasıl getirilebileceğini ve bunun için bu nokta bulutunun hangi işlemlere tabi tutulması gerektiği gibi sorulara çözüm aramak bu çalışmanın hedefleri arasındadır.

1.3. Çalışmanın Amacı

Bu çalışmada, Bergama LiDAR test alanına dair elde edilen nokta bulutu kullanılarak, Kestel Barajı rezervuar alanına ait sınırın otomatik olarak tespit edilmesi amaçlanmıştır. Aynı zamanda LiDAR uçuşunun yapıldığı tarihe yakın bir zamanda üretilen hâlihazır haritada mevcut rezervuar sınırı ile nokta bulutundan elde edilen baraj rezervuar sınırı kıyaslanarak doğruluk analizi yapılması amaçlanmıştır. Buna ilaveten, kullanılan iki farklı LiDAR sisteminin birbiriyle olan kıyası ve yükseklik faktörünün doğruluğa etkisi de incelenmiştir.

1.4. Metodoloji

Bu çalışmada, baraj, göl ve kıyı kenarları gibi su yüzeylerinin sınırlarının tespitinde lazer tarama teknolojisi ile üretilen 3B nokta bulutu verisinin ne kadar sağlıklı bir biçimde kullanılabileceğini tespit etmek için Kestel Barajı gölet alanı ele alınarak baraj rezervuar sınırının otomatik çıkarımı için aşağıdaki işlem adımları izlenmiştir.

- Bergama LiDAR test alanına ait elde edilen veri kümesinden, 156 km^2'ye karşılık gelen alandan yaklaşık olarak 1,5 milyar nokta içerisinden çalışma alanı olan Kestel Barajı çıkarılmıştır.
- Kestel Barajı ve çevresine ait nokta bulutu verisinin (yaklaşık 100 milyon) yoğun olması sebebiyle, Optech_1200 verisi; Optech_1200_38, Optech_1200_53, Optech_1200_54, Optech_1200_55, Optech_1200_70 ve Optech_1200_71 olarak alt çalışma alanlarına bölünerek filtreleme uygulamaları yapılmıştır.
- Gürültü filtrelemesi ve çakışan veri filtrelemesi işlemleri gerçekleştirilen nokta bulutu verisinde parlaklık (intensity) değerleri dikkate alınarak veri kümesindeki nokta sayısı bir miktar daha azaltılmıştır ve sadece baraj rezervuar alanı kalacak şekilde çalışma alanı boyutları küçültülmüştür.
- Elde edilen veri kümesindeki en düşük kot bilgisinden yararlanarak kıyı çizgisi 20-30 cm aralığında belirlenmiştir.
- RANSAC algoritması ile veri kümesinden düzlem geçirilerek kıyı çizgisini temsil eden noktalar tespit edilmiştir ve Kestel Barajına ait rezervuar alanı çıkarılmıştır.
- Bergama test bölgesinde gerçekleşen LiDAR ölçümleri iki farklı sistem ve iki farklı yükseklikten elde edilmiştir. Dört farklı veri için de aynı uygulamalar yapılmış ve rezervuar alanına ait sonuçlar elde edilmiştir.
- Kestel bölgesine ait ortofoto görüntüsünden elde edilen baraj sınırı ile otomatik olarak elde edilen sınırlar karşılaştırılmıştır.

1.5. RANSAC Algoritması

Rastgele Örnek Konsensüsü (RANSAC: RANdom SAmple Consensus) 1981 yılında Fisher ve Bolles tarafından geliştirilmiş bir model yakalama algoritmasıdır. Hough dönüşümü gibi dijital görüntü işleme alanında görüntüler üzerinde yer alan matematiksel olarak ifade edilebilen çizgi ve daire gibi nesnelerin çıkarımında kullanılan bir algoritmadır. RANSAC yüksek derecede gürültü ve aykırı değer içeren bir veri seti içinden geometrik nesnelere ait model kestiriminde kullanılmaktadır (Fischler and Bolles 1981; Hartley ve Zisserman 2000).

RANSAC yöntemi geleneksel model kestirim yöntemlerinin tersine bir algoritmadır. Geleneksel model kestirim algoritmalarında, başlangıç olarak olabildiğince fazla girdi veri

seçilip daha sonra geçersiz verileri model içinden elenirken RANSAC algoritmasında olabildiğince az sayıda girdi verisi kullanıp bu girdi veriyi modele uygun oluncaya kadar artırmaktadır. Örneğin 2 boyutlu bir nokta verisi seti içinde bir dairenin yayı modeli yakalamak için 3 nokta (daireyi oluşturan parametreleri hesaplayabilmek için) seçilir. Seçilen noktalar ile dairenin merkezi ve yarıçapı belirlenir. Daha sonra belirlenen bir eşik değeri altında kalan noktalar daireye atanır. Bu şekilde uygun model bulunana kadar işlem devam ettirilir (Fischler and Bolles 1981).

RANSAC algoritmasını çalışma prensibi şu şekilde ifade edilmiştir (Fischler and Bolles 1981; Hartley ve Zisserman 2000).

Amaç: Çok sayıda aykırı değer içeren S veri seti içinden uygun modelin yakalanması

Algoritma:

i. S veri seti içinden rastgele s sayıda örnek veri noktası seç ve model parametrelerini hesapla.
ii. Model belirlenen bir eşik değeri t (threshold) uzaklığı içinde kalan noktaları modele ata ve atanan bu noktalar için S_i veri setini belirle. S_i örnek verinin konsensüsüdür ve S veri seti için uygun modelin girdi noktalarını (inlier) tanımlar.
iii. Eğer S_i'nin boyutu (uygun nokta sayısı) model için yeterli nokta sayısını gösteren T eşik değerinden daha büyük ise, yeniden modeli S_i içindeki tüm noktaları kullanarak yeniden tahminle ve işlemi bitir.
iv. Eğer S_i'nin boyutu (uygun nokta sayısı) model için yeterli nokta sayısını gösteren T eşik değerinden daha küçük ise, yeniden s sayıda örnek veri noktası seç ve yukarıda anlatılan işlemi tekrar et.
v. N sayıda deneme sonucunda en büyük S_i konsensüsü S veri seti içinde aranan modele uygun konsensüsü olarak seçilir. Model S_i veri seti içinde yer alan tüm noktalara kullanılarak yeniden hesaplanır ve işlem bitirilir.

RANSAC algoritmasında seçilen s sayıda örnek veri noktası S veri seti içinde çıkarılacak modele ait parametrelerin hesaplanması için kullanılmaktadır. Model parametreleri çıkarılacak modelin geometrik yapısına göre farklılıklar göstermektedir.

Örneğin düz bir çizgi model yakalama için başlangıçta 2 adet örnek veri noktası seçilirken, düzlem model için 3 adet noktaya ihtiyaç duyulmaktadır.

RANSAC algoritması kullanılarak yapılan model yakalama işlemlerinde belirlenmesi gereken en önemli değerlerden bir tanesi eşik değeri (t) mesafesidir. Çünkü bir veri seti içindeki bir noktanın belirlenen bir α olasılığında girdi veri olarak atanabilmesi için bir t değerinin belirlenmesi gerekmektedir. Pratikte genellikle t eşik değeri mesafesi deneysel olarak belirlenmektedir (Fischler and Bolles 1981; Hartley ve Zisserman 2000).

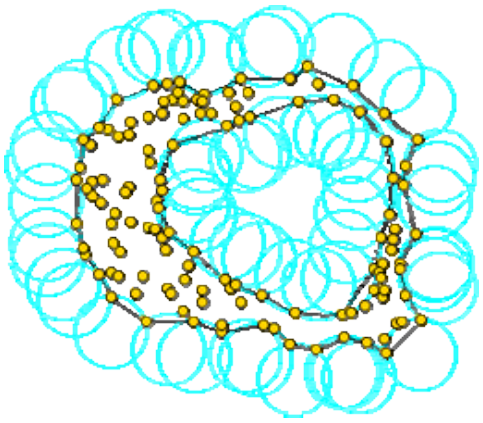
Veri seti içinde uygun modelin seçilmesi için tüm noktaların denenmesi hesaplama açısından uygun değildir. Bunun yerine (1) nolu formülde belirlenecek bir p olasılığındaki değere göre N defa s sayıda örnek nokta seçimi yapılarak modelin hesaplanması sağlanır. RANSAC algoritması için N deneme sayısı eşitlik 1'de gösterildiği gibi hesaplanmaktadır (Fischler and Bolles 1981; Hartley ve Zisserman 2000).

$$N = log(1-p) / log(1-(1-\varepsilon)^s) \quad (1)$$

Burada p doğru modelin seçilme olasılığı, ε seçilen noktaların modele aykırı olma olasılığı, s ise model parametrelerini hesaplamak için rastgele seçilmesi gereken minimum nokta sayıdır. Doğru modelin seçilme olasılığı olan p genellikle 0.99 olarak seçilmektedir (Fischler and Bolles 1981; Hartley ve Zisserman 2000).

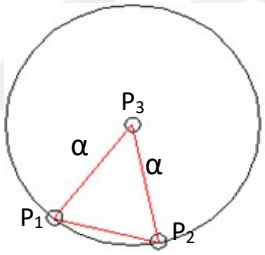
1.6 Alpha Shapes Algoritması

Düzensiz nokta bulutu verileri içinden istenen sınır değerini yakalayarak kapalı bir alan elde etmek için kullanılan algoritma olarak tarif edilebilir "*alpha shapes*" algoritması. Nokta bulutuna dair istenen sınır değerini elde etmek için algoritmaya ait en önemli parametre çap (radius) değeridir. Belirlenen çap değerine göre bu nokta bulutu etrafında daireler oluşturarak uygun noktaları tespit eder ve sınır çizimini sağlar (Şekil 5). Algoritma için eşik değeri olan çap (radius) değeri ne kadar büyük olursa oluşacak daireler büyür ve nokta bulutu etrafında dolaşarak oluşacak sınır konkav (dışbükey) görünür. Çap değerinin küçük tutulması ise oluşacak sınır değerinde nokta bulutuna ait her nokta dahil olabilir ve konveks (içbükey) görünüm elde edilir. Bu yüzden belirlenen çap değerinin oluşacak sınırın şeklindeki rolü büyüktür.



Şekil 5. "*alpha shapes*" algoritmasının çalışma prensibi

"*alpha shapes*" algoritması prensibi gereği nokta bulutu etrafında dolandırılan dairelerin oluşturulmasında izlenen matematik modelde; dairenin merkezindeki $P_3(X_3,Y_3)$ noktasının belirlenen çap (α) değeri kadar aynı uzaklığa sahip olan $P_1(X_1,Y_1)$ ve $P_2(X_2,Y_2)$ noktalarını (2), (3), (4) ve (5) nolu formüller ile tespit edilmiştir (Şekil 6).



Şekil 6. Uzaklık kesişme algoritması

Uzaklık kesişme algoritması (Zhang Hong, 2006):

$$x_3 = x_1 + \frac{1}{2}(x_2-x_1) + H\,(y_2-y_1) \qquad (2)$$

$$y_3 = y_1 + \frac{1}{2}(y_2-y_1) + H\,(x_1-x_2) \qquad (3)$$

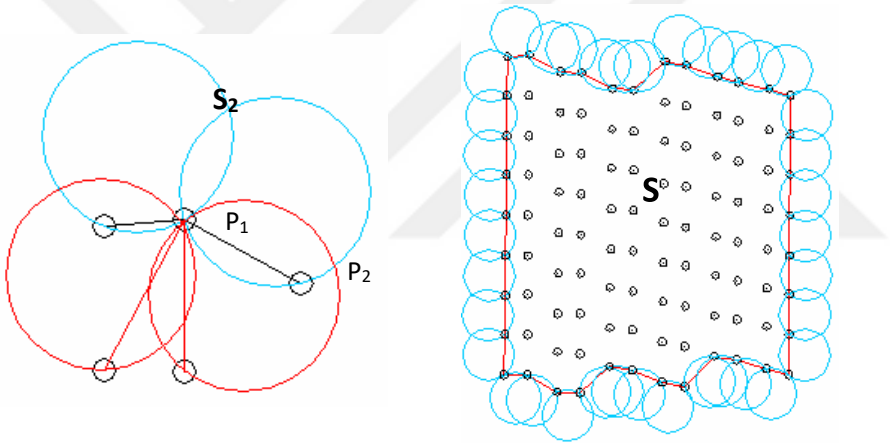
$$H = \sqrt{\frac{\alpha^2}{S^2_{P1P2}} - \frac{1}{4}} \qquad (4)$$

$$S^2_{P1P2} = (x_1 - x_2)^2 + (y_1 - y_2)^2, \quad a = \text{çap (radius)} \qquad (5)$$

"*alpha shapes*" algoritması uygulanırken sınır çizimine hangi noktaların dahil edilmesi gerektiği kararı ve sonucu şu şekilde alınır:

Karar; Belirlenen çap değeri ile nokta bulutu etrafında gezdirilen dairelerin içinde her hangi bir nokta bulunmaması gerekir. Bu yüzden noktaların yoğun olduğu iç bölgelerden değil dış bölgelerden seçilen çap değerine göre yolunu izlemektedir (Şekil 7-a). Nokta bulutunun içinde oluşan bir boşluk varsa seçilen çap değerine göre oluşturulan daire için uygun ise o alanda da sınır oluşturulur.

Sonuç; Belirlenen çap değeriyle oluşan daireyi temsil eden noktalar birbirine bağlanarak sınır hattı oluşur eğer aynı daireyi ikiden fazla nokta temsil ediyorsa bu sefer birbirlerine en uzak olan iki nokta arasında sınır hattı çizilir ve bu prensip nokta bulutunun tamamında uygulandıktan sonra sınır çizimi tamamlanmış olur (Şekil 7-b).

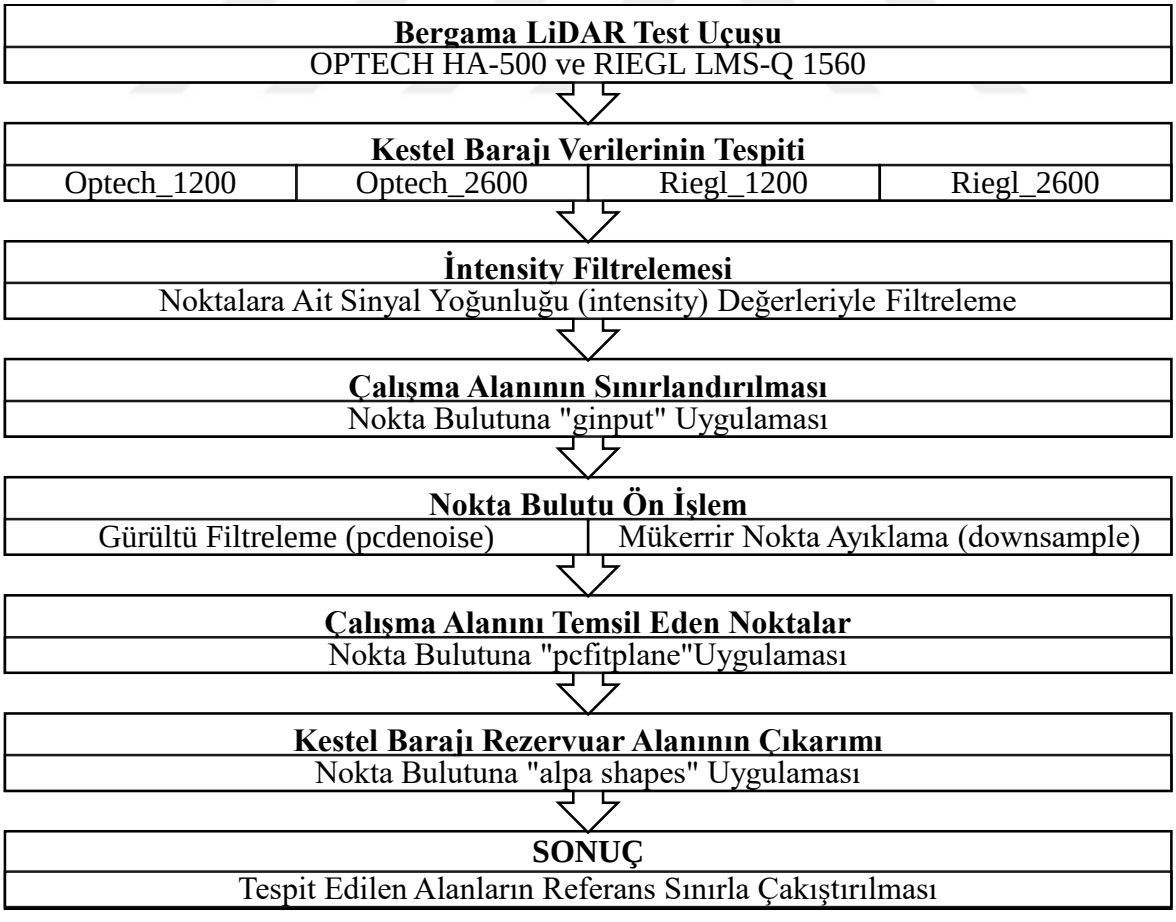


(a) (b)

Şekil 7. "alpha shapes" algoritması uygulanırken alınan karar ve sonuç

2. YAPILAN ÇALIŞMALAR

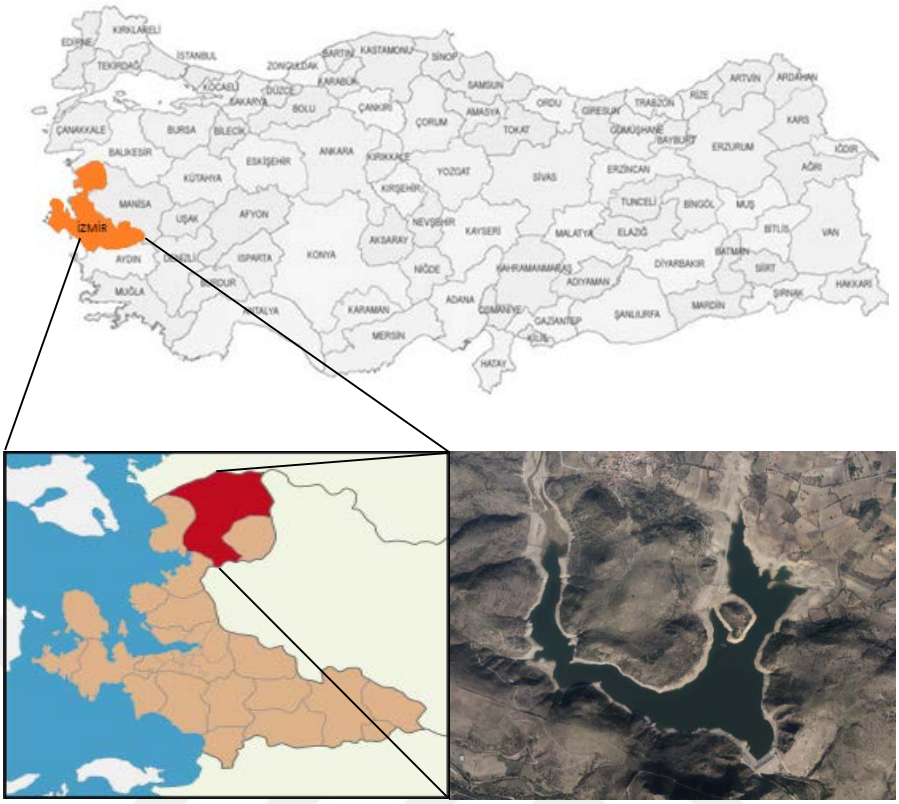
Yapılan çalışmalara dair işlem sırası iş akış diyagramı şeklinde gösterilmiştir (Şekil 8). Kullanılan dört farklı veri kümesi (Optech_1200, Optech_2600, Riegl_1200 ve Riegl_2600) içerisinden Optech_1200 verisi örnek olarak ele alınmış ve bütün aşamalar bu veri seti üzerinden anlatılmıştır. Optech_1200 veri setinin haricindeki veriler içinde aynı uygulamalar yapılmış fakat tekrara girmemesi için sadece sonuç kısımları ele alınmıştır. İlerleyen aşamalarda bu dört veri setinden elde edilen sonuçlar 2013 yılında hava fotoğraflarından elde edilmiş olan hâlihazır haritadan elde edilen Kestel Barajına ait bir referans sınırı ile karşılaştırılarak alansal benzerlik ve farklılık tespit edilmiştir. Aynı zamanda tespit edilen sınırlar Google Earth görüntüleriyle çakıştırılmış fakat LiDAR uçuşunun yapıldığı tarihe ait görüntü olmadığı için çok sağlıklı yorum yapılamamış fakat Google Earth üzerinden alınan konum bilgileri ile tespit edilen sınırların polinom eğri katsayıları karşılaştırılarak yapılan çalışmanın doğruluğu hakkında bilgi sahibi olunarak çalışma sonuçlandırılmıştır.



Şekil 8.Yapılan çalışmalara ait iş akış diyagramı

Kestel Barajı, İzmir, Bergama kenti yakınındaki Kestel Çayı üzerinde sulama amacıyla 1983-1988 yılları arasında inşa edilmiştir (Şekil 9). Bu tez kapsamında yapılan çalışmalar, öncelikle Bergama test alanına ait LiDAR verileri içerisinden Kestel Barajına ait verileri FugroViwer programı yardımıyla belirleyerek hangi alt çalışma alanlarından oluşturulduğu tespit edilmiştir. Bu tespitin ardından söz konusu nokta bulutu için MATLAB programlama dili yardımıyla noktalara ait parlaklık (intensity) değerleri dikkate alınarak filtreleme işlemi yapılmıştır. Akabinde nokta bulutu verileri için ön işlem uygulamalarından olan gürültü filtreleme (pcdenoise) ve mükerrir noktaların tespiti ve elenmesi için (downsample) filtreleme işlemleri yapıldıktan sonra çalışma alanının küçültülerek sadece baraj rezervuar alanını ilgilendirecek kısımlara indirgenmesi sağlanmıştır. Sonrasında rezervuar alanını temsil eden nokta bulutları ile alan ve şekil çıkarım işlemi MATLAB programlama dili yardımıyla yapılmıştır. Referans olarak tespit edilen Kestel Baraj sınırı ile çakıştırma işlemi gerçekleştirilmiştir. Sonuç olarak elde edilen sınırları ve referans sınırı ile çakıştırma işlemi sonucu ortaya çıkan farkları irdeleyerek yapılan çalışmanın doğruluğu ve uygulanabilirliği noktasında fikir sahibi olunmuştur.

Harita ve harita bilgisi üretimlerinde kullanılan LiDAR sistemlerinin ülkeye kazandırılması amacıyla, Harita Genel Komutanlığı'nın başkanlığını yürüttüğü Bakanlıklar Arası Harita İşlerini Koordinasyon ve Planlama Kurulu'nun (BHİKPK) Bilimsel Araştırma ve Koordinasyon Komisyonuna (BARKOK) 2014-2015 yılı döneminde "LiDAR Test Uçuşunun Gerçekleştirilmesi" görevi verilmiştir. Bu kapsamda 23 Ekim–06 Kasım 2014 tarihleri arasında Bergama test bölgesinde iki ayrı yükseklikten (1200 m ve 2600 m) Optech firmasının Pegasus HA-500 ve Riegl firmasının LMS-Q1560 LiDAR sistemlerinin Türkiye temsilcisi NİK İnşaat ve SEZA Elektronik LTD.Şti.'lerinin aracılığıyla havadan "LiDAR Test Uçuşları" gerçekleştirilmiştir. İki farklı yükseklikte ve nokta sıklığında alınan LiDAR test verisi firmaların kendi geliştirdikleri ve sırasıyla isimleri LMS (LiDAR Mapping Suite) ve RiProcess (Riegl Process) olan yazılımlar kullanılarak işlenmiş ve test bölgesine ilişkin nokta bulutları elde edilmiştir (Kayı vd, 2015).



Şekil 9. İzmir ili Bergama ilçesi ve Kestel Barajının haritadaki konumu

2.1. LiDAR Test Uçuşu

LiDAR test uçuşu, NİK İnşaat Ticaret Ltd.Şti.'nin Türkiye temsilciliğini yürüttüğü Optech firmasının Pegasus HA-500 LiDAR sistemi ve SEZA Teknik Cihazlar Ltd.Şti.'nin temsilciliğini yürüttüğü Riegl firmasının LMS-Q1560 LiDAR sistemiyle 13 Ekim - 07 Kasım 2014 tarihleri arasında gerçekleştirilmiştir. Bergama ilçesini merkeze alacak şekilde, başta su, orman ve tarım alanları olmak üzere diğer detayları da kaplayan test bölgesinin alanı yaklaşık 156 km^2'dir. Bölgede 51 adet yer kontrol noktası Tusaga-Aktif Uyumlu RTK GNSS alıcısı kullanılarak ölçülmüştür (Kayı vd, 2015).

2.1.1. Optech Pegasus HA-500 Test Uçuşu

NİK İnşaat Ticaret Ltd.Şti.'nin Türkiye temsilciliğini yürüttüğü Optech firması LiDAR testine Pegasus HA-500 LiDAR sistemi ile katılmıştır. Pegasus HA-500 sistemi ile ilgili ayrıntılı bilgi Tablo 1'de verilmiştir (Optech, 2014).

Tablo 1. Optech Pegasus HA-500 teknik özellikleri

Uçuş Yüksekliği	150-5000 m
Efektif Lazer Frekansı	100-500 kHz
Tarama Açısı	0-75º Ayarlanabilir
Nokta Koordinatı Ölçme Doğruluğu (KOH)	≤ 5-20 cm.
Tarama Mekanizması	Salınımlı

Bu sistem ile 1200 m uçuş yüksekliğinde testte talep edilene uygun olarak metrekarede en az 8 nokta olacak şekilde, %25 bindirmeli 32 kolon planlanmıştır. Bu yükseklik için tarama açısı 35º olarak ayarlanmış ve tarama genişliği 580 m olmuştur. İkinci uçuş 2600 m yükseklikten testte talep edilen 2 nokta/m² yoğunluğun uygun olacak şekilde planlanmış, tek geçişte bu talebin karşılanamaması sebebiyle %50 bindirmeli 18 uçuş kolonu planlanarak istenilen yoğunluk karşılanmıştır (Kayı vd, 2015).

20-21 Ekim 2014 tarihleri arasında Bergama bölgesinde planlandığı şekilde iki farklı yükseklikte test uçuşu gerçekleştirilmiştir. Uçuşlara ilişkin ayrıntılı bilgi Tablo 2'de verilmiştir (Kayı vd, 2015).

Tablo 2. Optech Pegasus HA500 LiDAR sistemi test uçuşu

Uçuş Yüksekliği	1200 m	2600 m
Açı	35	20
Hız (knots)	150	150
Yoğunluk (nokta/m^2)	≥8	≥2
Kolon Sayısı	32	18
Bindirme	%25	%50

Uçak üzerindeki uçuş esnasında kaydedilen GNSS\IMU verileri, test bölgesine 65 km uzaklıkta bulunan AYVL ve 45 km uzaklıkta olan KIKA Tusaga-Aktif istasyonlarının 1 sn aralıklı RINEX formatındaki verilerine göre POSPac MMS yazılımı ile işlenmiştir.

LiDAR verileri LMS yazılımı ile işlenmiş ve 51 adet kontrol noktasının 26 adedi kontrol noktası, 25 adedi ise denetleme noktası olarak kullanılmıştır. Denetleme noktalarına göre karesel ortalama hata ± 0.07 m olarak hesaplanmıştır (Optech, 2014-a).

2.1.2. Riegl LMS-Q 1560 Test Uçuşu

SEZA Teknik Cihazlar Ltd.Şti.'nin temsilciliğini yürüttüğü Riegl firmasının LMS-Q1560 LiDAR sistemi ile teste katılmıştır. RIEGL LMS-Q1560 sistemi ile ilgili ayrıntılı bilgi Tablo 3'de verilmiştir (Kayı vd, 2015).

Tablo 3. Riegl LMS-Q1560 teknik özellikleri

Uçuş Yüksekliği	400-4700 m
Efektif Lazer Frekansı	200-800 kHz
Tarama Açısı	58/60 º
Nokta Koordinatı Ölçme Doğruluğu (KOH)	2 cm
Tarama Mekanizması	Döner

Bu sistemle 1200 m uçuş yüksekliğinde metrekarede en az 8 nokta olacak şekilde % 50 bindirmeli, 35º tarama açısıyla 29 kolon planlanmıştır. 2600 m yükseklikte metrekarede en az 2 nokta gelecek şekilde %30 bindirmeli 11 uçuş kolonu planlanmıştır. Sistem kapasitesinin 1200m yükseklikten metrekarede 6 nokta elde etmesi nedeniyle istenilen ölçüt (8 nokta/m^2) bindirme oranı ile karşılanmıştır.

03-05 Kasım 2014 tarihleri arasında Bergama bölgesinde iki farklı yükseklikte test uçuşu gerçekleştirilmiştir. Uçuşlara ilişkin ayrıntılı bilgi Tablo 4'de verilmiştir (Kayı vd, 2015).

Tablo 4. Riegl LMS-Q1560 LiDAR sistemi test uçuşu

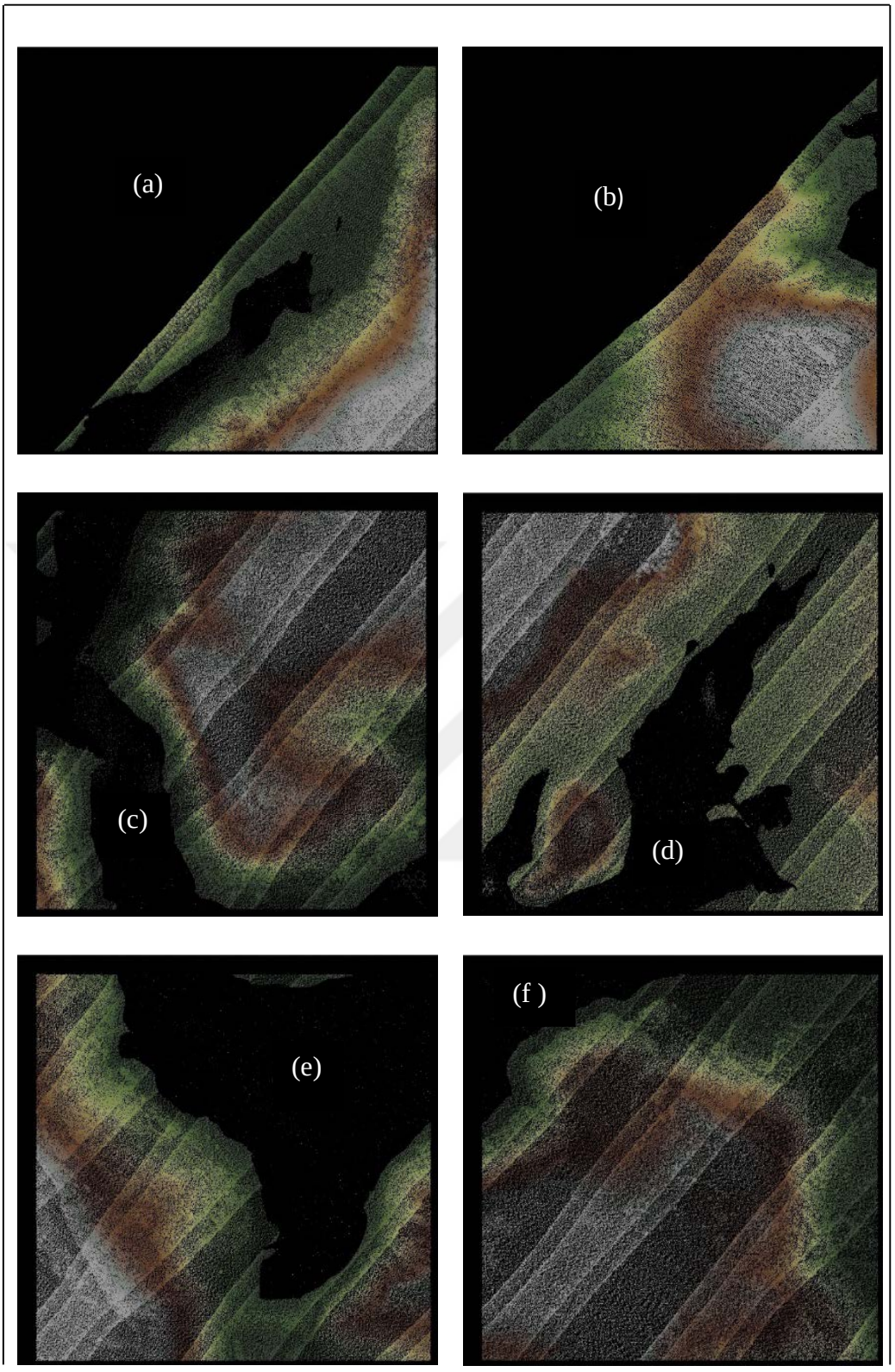
Uçuş Yüksekliği	1200 m	2600 m
Açı	30	30
Hız (knots)	150	150
Yoğunluk (nokta/m^2)	≥8	≥2
Kolon Sayısı	29	11
Bindirme	%50	%30

GNSS\IMU verileri test bölgesine 65 km uzaklıkta bulunan AYVL ve 45 km uzaklıkta olan KIKA Tusaga-Aktif istasyonlarının 1 sn aralıklı rinex formatındaki verileriyle POSPac MMS yazılımı ile işlenmiştir. LiDAR yer kontrol noktaları alan detay olmadığından dolayı Riegl sisteminde etkin olarak kullanılamamış, arazide yükseklik değişimini olmadığı varsayılarak noktalar 2×2 boyutunda alan detaylara dönüştürülmüş fakat sağlıklı olarak test edilememiştir. Tüm verinin bağıl standart sapması ± 0.03m olarak hesaplanmıştır (RIEGL,2014-a).

2.1.3. LiDAR Verilerinin Görüntülenmesi

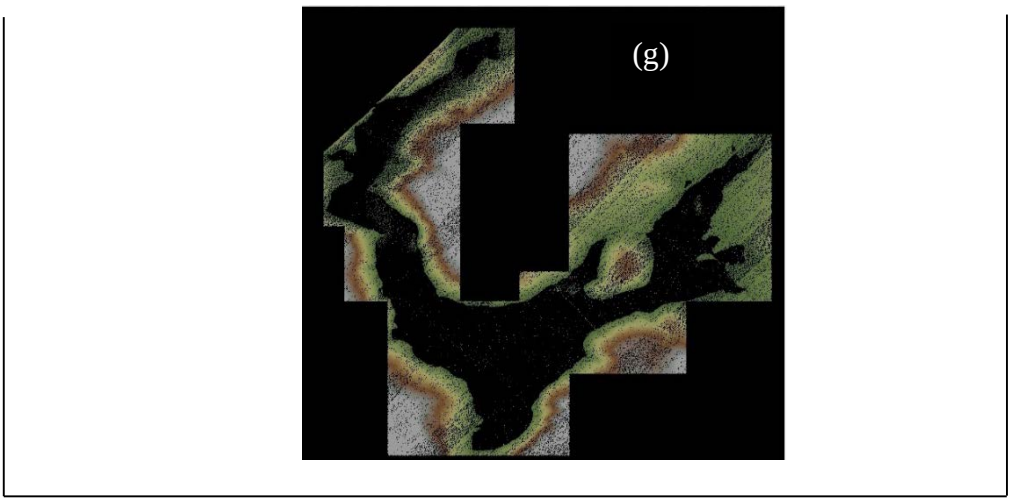
Bergama test alanında gerçekleştirilmiş olan dört farklı LiDAR uçuşu sonucunda elde edilen verilerden, söz konusu Kestel Baraj rezervuar sınırının tespiti için ilgili alanlar parça parça tespit edilmiştir. Çalışma alanındaki nokta bulutuna ait veri fazlalığından dolayı parça parça işleme tabi tutulan veriler sonrasında Lastools programı yardımıyla birleştirilerek işleme devam edilmiştir (Şekil 10, 11, 12, 13).

Optech HA-500 LiDAR sistemi ile 1200 m. yükseklikten gerçekleştirilen uçuş sonucu elde edilen görüntülerden Kestel Barajı ve yakınındaki nokta bulutları FugroViwer programı aracılığı ile görüntülenerek çalışma alanı tespit edilmiştir. Söz konusu çalışma alanı 6 farklı *.*las* dosyasından oluşmaktadır.

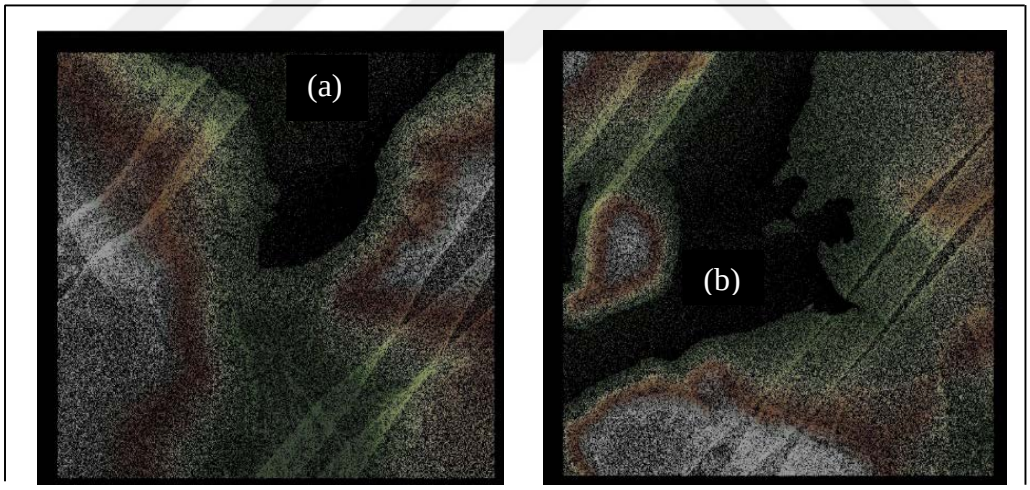


Şekil 10. Optech_1200 verisine ait nokta bulutlarının parçalar halindeki alt çalışma verilerine ait görüntüleri (a,b,c,d,e ve f) ve aynı verilerin birleştirilmiş görüntüsü (g)

Şekil 10'un devamı

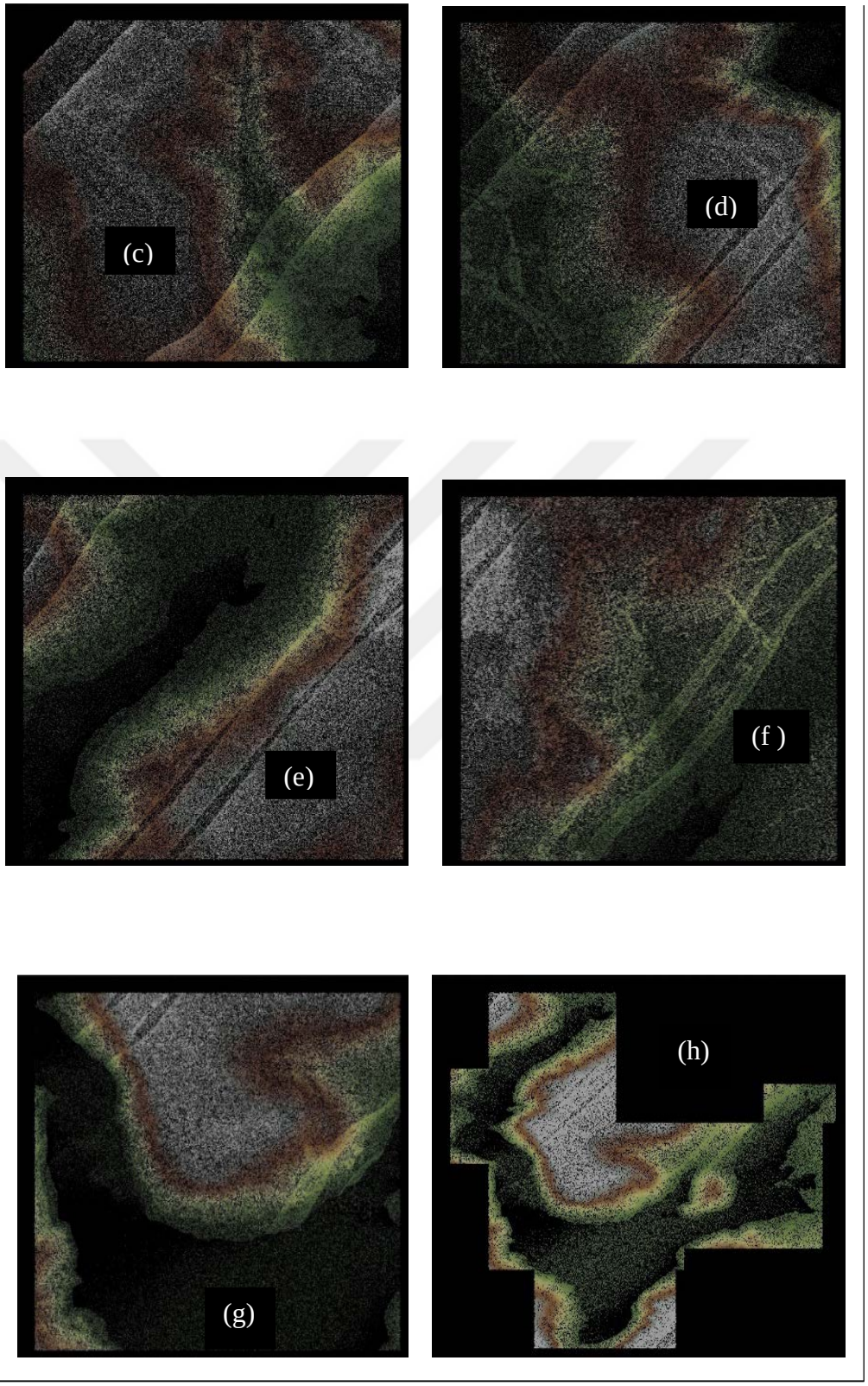


Optech HA-500 LiDAR sistemi ile 2600 m. yükseklikten gerçekleştirilen uçuş sonucu elde edilen 7 farklı *.las uzantılı dosya FugroViewer programı aracılığı ile görüntülenerek Kestel Barajı çevresine ait nokta bulutu üretilmiştir.

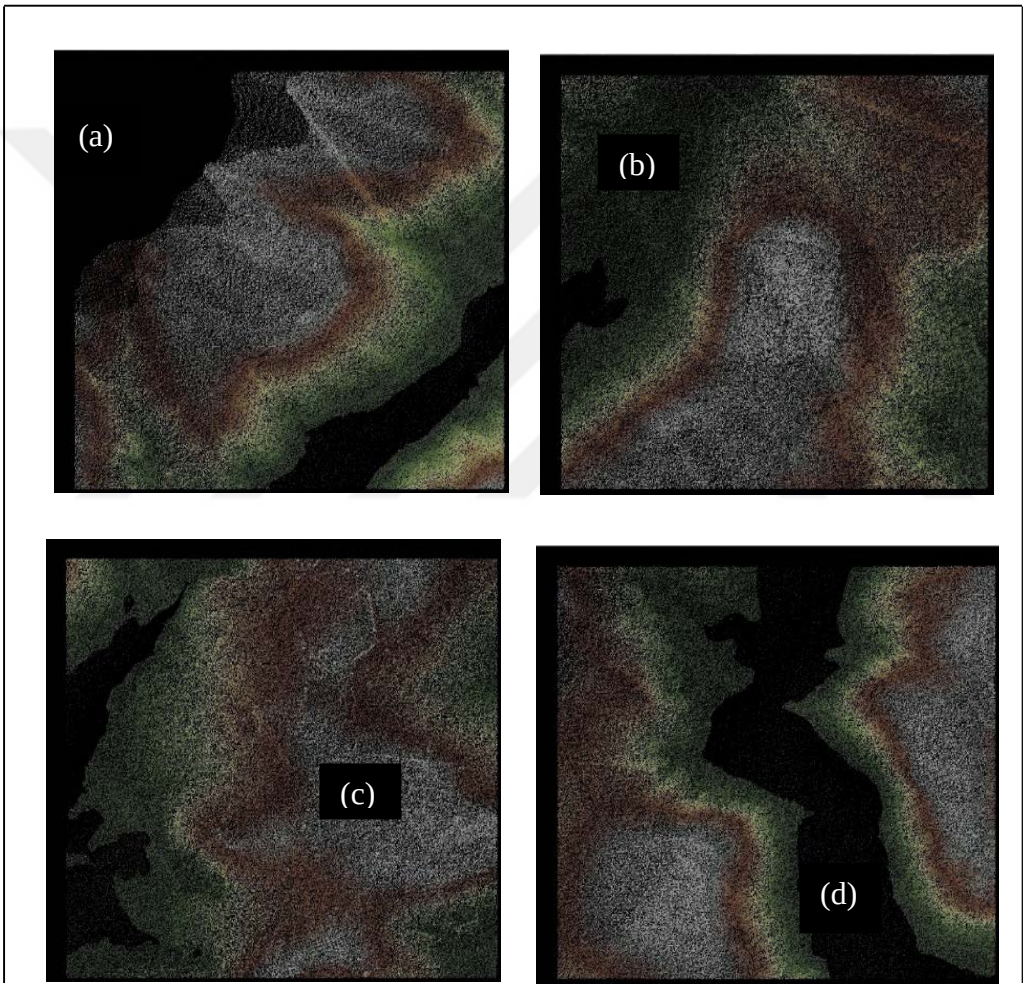


Şekil 11. Optech_2600 verisine ait nokta bulutlarının parçalar halindeki alt çalışma verilerine ait görüntüleri (a,b,c,d,e,f ve g) ve aynı verilerin birleştirilmiş görüntüsü (h)

Şekil 11’in devamı

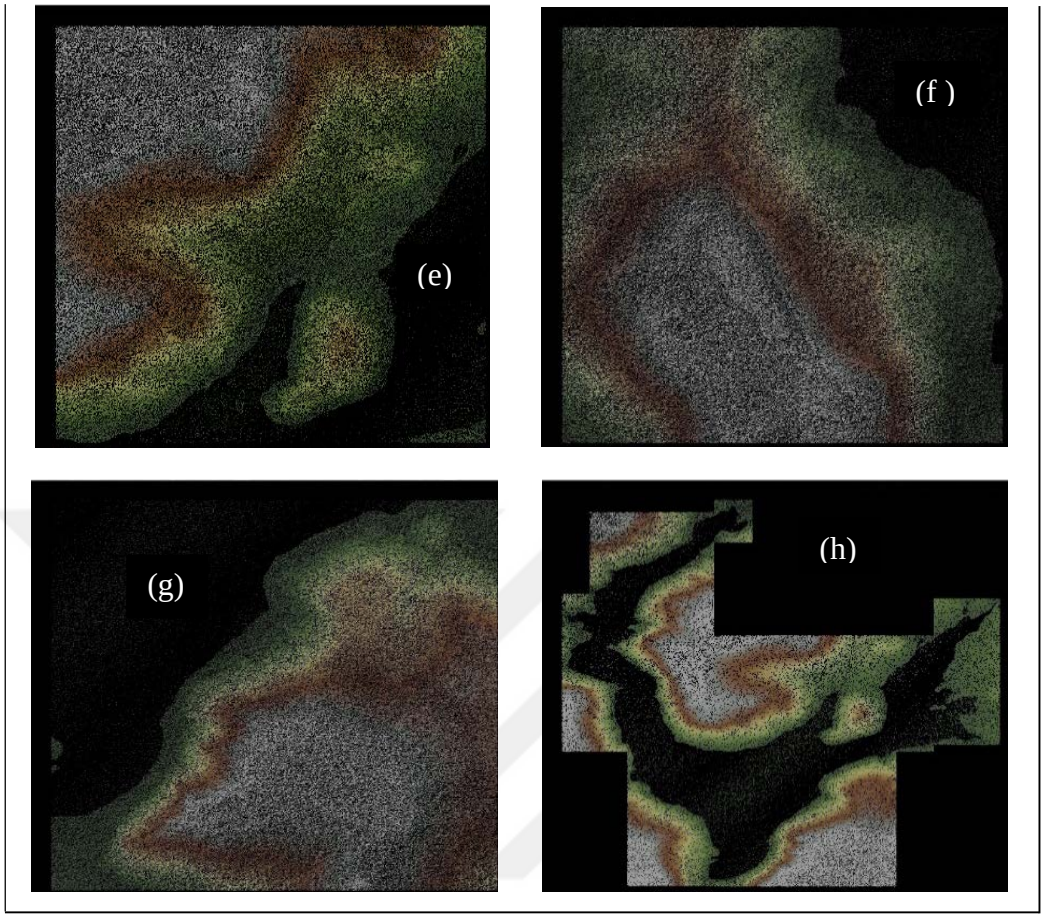


Riegl LMS-Q 1560 LiDAR sistemi ile 1200 m. yükseklikten gerçekleştirilen uçuş sonucu 7 farklı **.las* dosyasından oluştuğu tespit edilmiştir. Bu veriler FugroViwer programı aracılığı ile görüntülenerek hangi nokta bulutlarının Kestel Barajı ve etrafını gösterdiği tespit edilmiştir. Çalışılan nokta bulutu verisi içerisinden istenilen alanı belirlerken *"ginput"* komutu ile işlem yapılır. Bu işlem sırasında bölgeye dair seçilen noktaların içerisinden en küçük ve en büyük koordinat değerlerini (x,y) tespit ederek bu koordinatların sınırlandırdığı alanı belirlemiş olur.

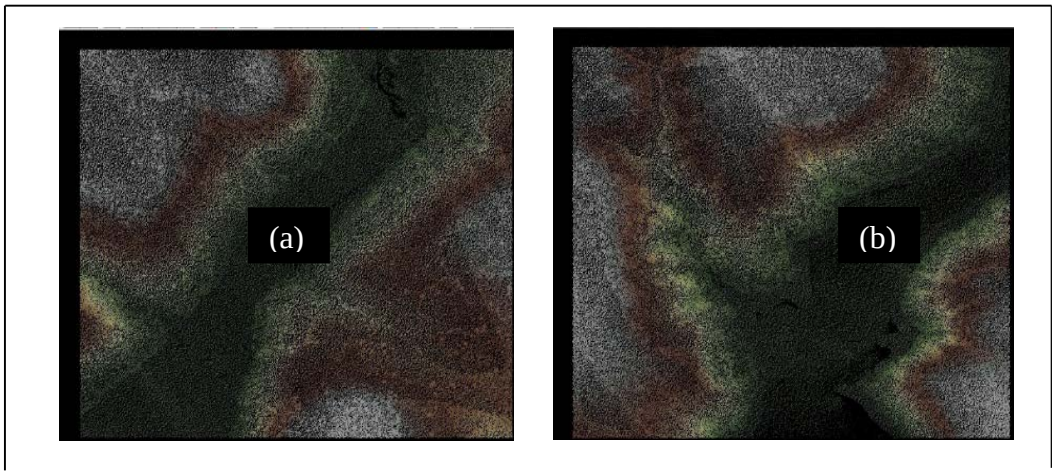


Şekil 12. Riegl_1200 verisine ait nokta bulutlarının parçalar halindeki alt çalışma verilerine ait görüntüleri (a,b,c,d,e,f ve g) aynı verilerin birleştirilmiş görüntüsü (h)

Şekil 12'nin devamı

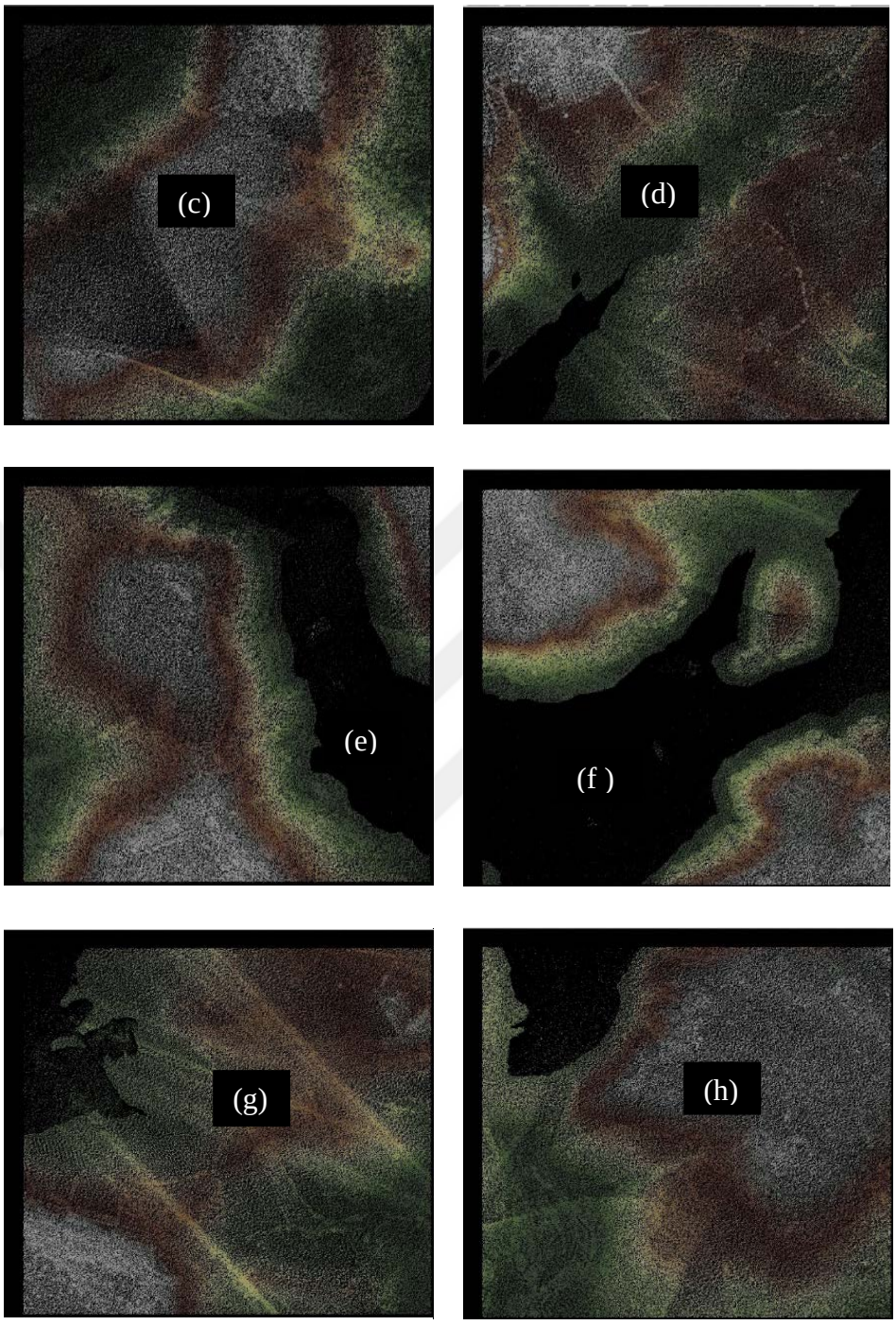


Riegl LMS-Q 1560 LiDAR sistemi ile 2600 m. yükseklikten gerçekleştirilen uçuş sonucu elde edilen 7 farklı *.*las* uzantılı dosya FugroViewer programı aracılığı ile görüntülenerek Kestel Barajı ve yakınındaki nokta bulutları tespit edilmiştir.

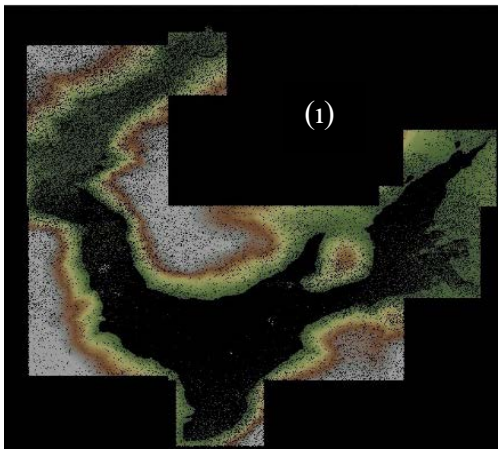


Şekil 13. Riegl_2600 verisine ait nokta bulutlarının parçalar halindeki alt çalışma verilerine ait görüntüleri (a,b,c,d,e,f,g ve h) ve aynıverilerin birleştirilmiş görüntüsü (ı)

Şekil 13’ün devamı



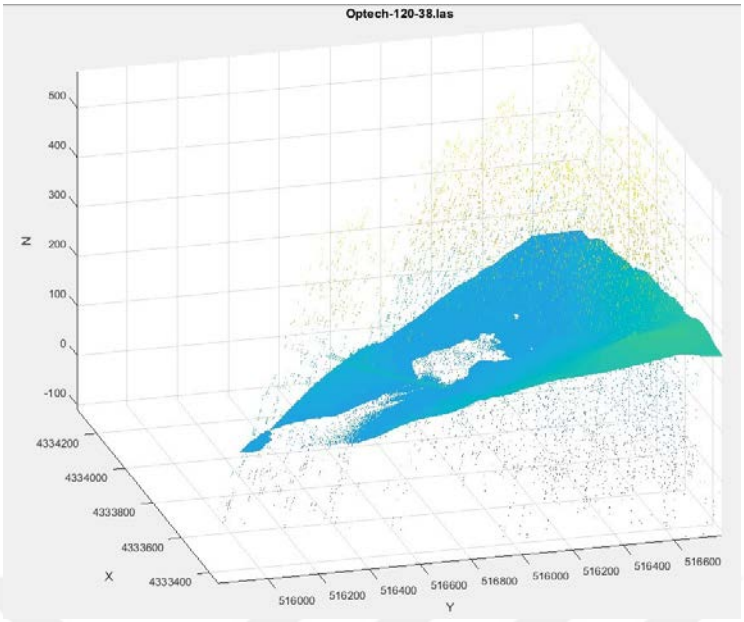
Şekil 13'ün devamı



2.2. 3B Nokta Bulutu Verilerinin Sinyal Yoğunluğu (İntensity) Değerlerine Göre Filtrelenmesi

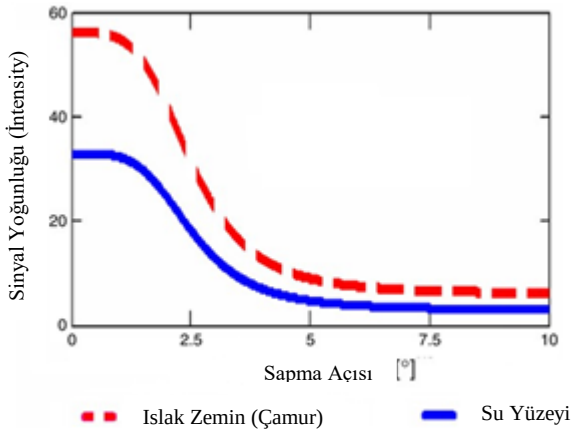
Nokta sayısı fazla olan LiDAR verileriyle MATLAB ortamında çalışmanın zorluğundan dolayı yapılan her işlem *.las uzantılı veriler için ayrı ayrı uygulanmıştır. Parça parça yapılan bu işlemler sonucunda elde edilen veriler birleştirilerek sınır çıkarma işlemine bu haliyle devam edilmiştir. Mevcut test bölgesine ait elimizde bulunan dört farklı LiDAR verisi (Optech_1200, Optech_2600, Riegl_1200 ve Riegl_2600) için yapılan uygulamalar temsil amaçlı Optech-1200 verisi görselleri kullanılarak anlatılmış ve sonuç kısmında ise bu dört veri için bulunan sonuçlar birlikte karşılaştırılmıştır.

Öncelikle MATLAB ortamında LiDAR verisi okutabilmek için *.*las* uzantılı veriler için yazılmış olan "*lasdata*" fonksiyonu (EK 1) kullanılmıştır. Optech_1200 nokta bulutunu oluşturan ilk kısım Optech_1200_38 verisi olmuştur. Bu fonksiyon sayesinde *.*las* verilerinin içinde bulunan bütün bilgilere (x,y,z koordinatları, intensity değerleri vb.) ulaşarak koordinat bilgisi ve intensity değerleri tespit edilmiş (Şekil 14). Nokta bulutu içerisinden kıyı hattını oluşturacak olan yani karadan yansıyan noktaları tespit edebilmek için noktaların sinyal yoğunluğu (intensity) değerleri dikkate alınmıştır. Baraj rezervuar alanını yansıtacak noktalar için belirlenen sinyal yoğunluğu değeri altında kalan noktaları eleyerek bir filtreleme uygulanmıştır.



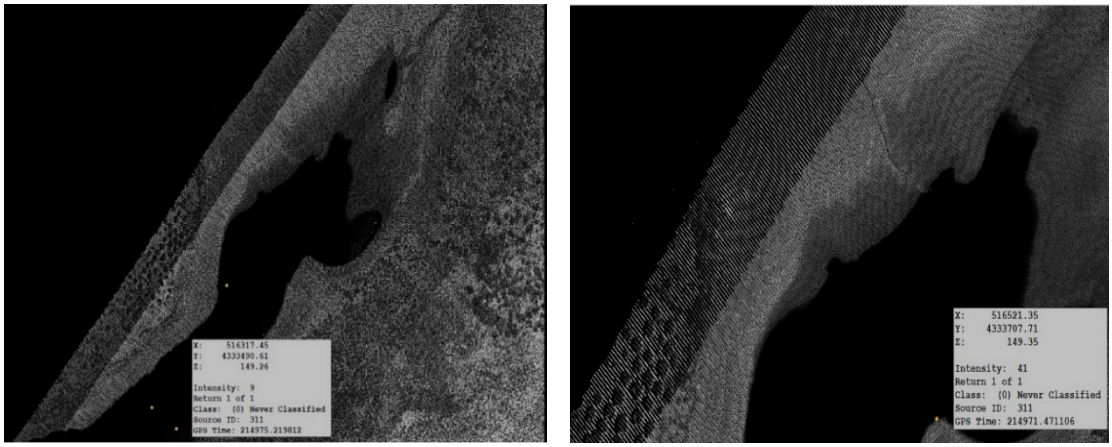
Şekil 14. Optech_1200_38 verisi nokta bulutu gösterimi

LiDAR sisteminde kullanılan lazer ışınının nereden dönüp geldiği ve nasıl bir yoğunlukla bu işlemi tamamladığı önemli bir bilgidir. Söz konusu baraj rezervuar alanının tespiti olduğundan sağlıklı bir kıyı hattının belirlenebilmesi için kara ve su ayırımının iyi bir şekilde yapılması gerekir. Bergama test bölgesinde iki farklı LiDAR sistemindeki lazer ışınları kızılötesine en yakın elektromanyetik ışık spektrumuna sahip oldukları için su yüzeyinde soğurma (yutma) durumuyla karşı karşıya kalırlar. Bundan dolayı su yüzeyine ait veri elde edilemezken su yüzeyine yakın veya su yüzeyindeki cisimlerden dönen sinyallerin yoğunluk (intensity) değerlerine göre filtrelenmesi gerekir. Bu uygulama için belirlenen sinyal yoğunluk (intensity) değeri daha önce yapılmış olan benzer bir çalışmadan (Höfle vd., 2009) ıslak zemin (çamur) ile su yüzeyini ayırt etmek için tespit edilen intensity değeri kullanılmıştır (Şekil 15). Bu değerin daha sağlıklı bir şekilde tespit edilebilmesi için bölgede yapılacak olan sinyal yoğunluğu (intensity) ölçümlerini değerlendirmek gereklidir. Ancak bu aşamada böyle bir imkân olmadığı için sözü edilen çalışmada tespit edilmiş olan intensity değeri kullanılmıştır.



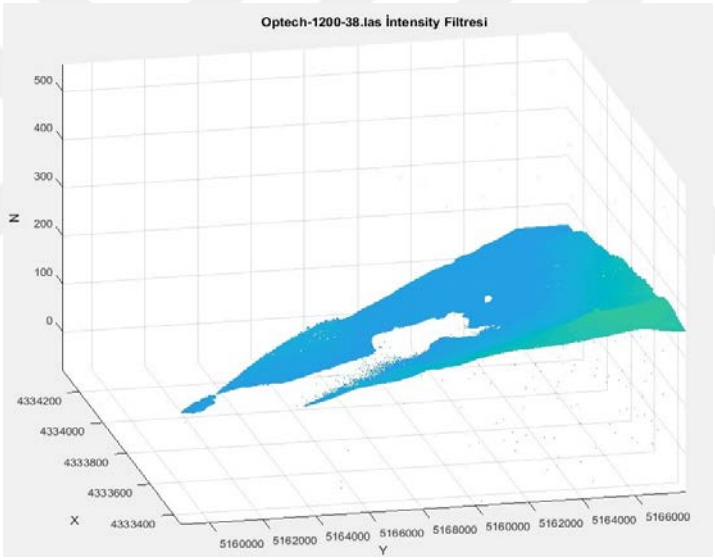
Şekil 15. Islak zemin (çamur) ve su yüzeyine ait sinyal yoğunluğu (intensity) değerlerinin gösterimi (Höfle vd., 2009)

Kara ve su ayırımını yaparak kıyı hattını belirlemek için kullanılacak sinyal yoğunluk (intensity) değeri 40 olarak belirlenmiştir. Bu değerin altında kalan verilerin kıyı hattını temsil etmediği düşünüldüğü için mevcut nokta bulutundan bu veriler ayıklanarak çıkarılmalı yani filtre edilmelidir. Böylelikle su yüzeyine yakın mesafeden (su altı 2-3m) gelen verileri ve su yüzeyinin üstündeki sazlık, balık çitleri için kurulan kafesler gibi cisimlerden gelen sinyaller karadan gelen sinyaller kadar yoğun olamayacağı için bu filtreleme sonucu bu veriler ayıklanmıştır (Şekil 16). Böylelikle Kestel Barajı rezervuar alanı tespiti için kullanılacak veri kümesini belirlemek için ilk filtreleme işlemi intensity değerlerine göre yapılmıştır.



Şekil 16. Noktalara ait sinyal yoğunluğu (intensity) gösterimi (Optech_1200_38 verisi)

Sinyal yoğunluğu (intensity) filtrelemesi ve sonraki uygulamalar için las dosyalarından okuttuğumuz değerler arasından 3B konum bilgisi (x,y,z) ve intensity değerleri ile filtreleme işlemi yapılmıştır. Ancak konum bilgisine ait verilerin double (çift duyarlılıklı) dizi formatında olup intensity değeri ise unit16 veri türünde olmasından kaynaklı birbirleri arasındaki dönüşüme gerek duyulmuştur. Bu dönüşüm "*im2double*" fonksiyonu ile gerçekleştirilmiş ve double veri türünde 40 değerine (filtrelemede baz alınan değer) sahip olan değerin unit16 veri türünde 0.0006103608758678569 değerine eşit olduğu tespit edilmiştir. Böylelikle matlab ortamında yazılan kodlarda (EK 2) da görüldüğü gibi baz alınan değere eşit ve ondan büyük olan değerler filtrelenmiştir. Kestel Barajı rezervuar alanını tespit etmek için kullanılacak noktalar için ilk filtreleme işlemi gerçekleştirilmiştir (Şekil 17).



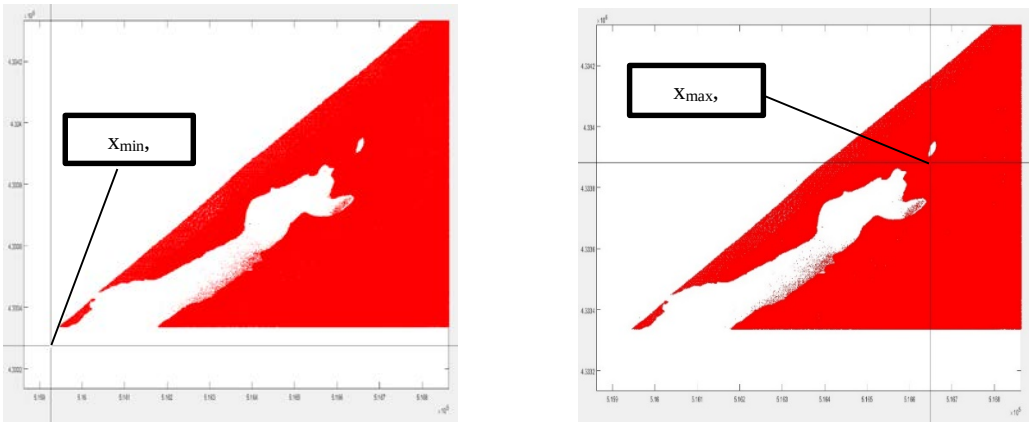
Şekil 17. Sinyal yoğunluğu (intensity) filtreleme sonucu
(Optech_1200_38 verisi)

Sinyal yoğunluğu (intensity) filtrelemesi sonucunda Optech_1200_38 verisine ait olan 6061830 nokta sayısı 4618134'e düşmüştür (Tablo 5). Bu filtreleme sonucunda %24 oranında veri ayıklanmıştır.

2.3. Çalışma Alanının Sınırlandırılması

Kestel Barajı rezervuar alanının tespitinde yardımcı olacak noktalar belirlenirken, ilgisiz alanlar silinerek nokta yoğunluğu azaltılmıştır. Bu sebepten dolayı MATLAB ortamında *"ginput"* fonksiyonu yardımıyla yazılan kodlar (EK 2) sayesinde bu veri kümesindeki nokta sayısı azaltılmıştır.

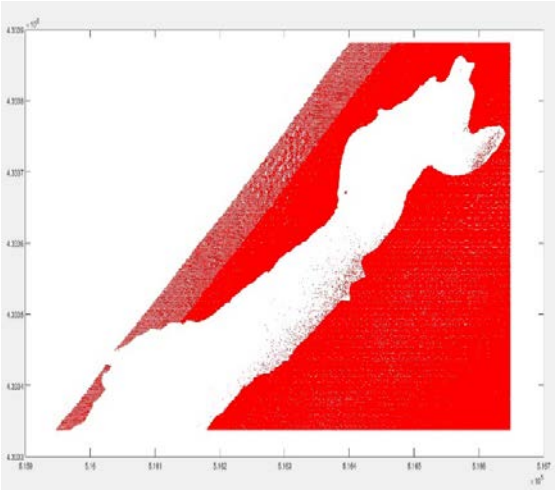
"*ginput*" fonksiyonuyla gerçekleştirilen uygulamada çalışma alanının belirlenmesinde ve aynı zamanda çalışma alanının dışında kalan alanın silinmesinde dikkat edilen husus, çakışacak olan eksenlerin iyi belirlenmesidir. Söz konusu uygulamada baraj rezervuar alanı tespiti olduğu için baraj suyuna yakın alanların tespiti için özen gösterilmiştir. Çalışma alanı belirlenirken seçilen noktalar içinde koordinat değerleri en az ve an fazla (x_{min}, y_{min}, x_{max}, y_{max}) değere sahip olan noktaların oluşturduğu sınırlar içerisinde kalan alan çalışma alanı olarak tespit edilir. Bu prensip gereği seçilen noktalar çalışma alanı olarak seçmek istediğimiz yerin çapraz olarak köşe noktaları olması gerekir. Sağdan sola veya soldan sağa çapraz köşeler olacak şekilde seçilen noktalar çalışılacak bölgeyi daraltarak gereksiz noktalardan arındırmaktadır (Şekil 18). Böylece nokta bulutu içerisinde ciddi boyutta bir veri azaltılmış olmaktadır.



(a) (b)

Şekil 18. "*ginput*" fonksiyonu ile çalışma alanının seçimi (Optech_1200_38 verisi);
a) ve b) Öncesi durum c) Sonrası durum

Şekil 18'in devamı



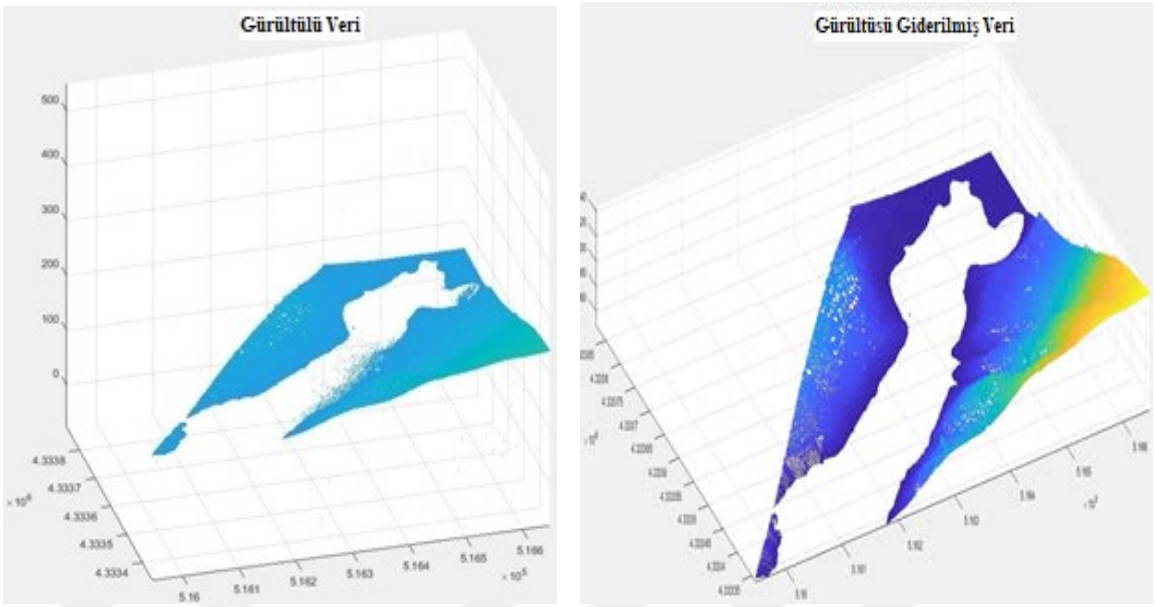
(c)

"*ginput*" fonksiyonuyla yapılan uygulama sonucunda veri kümesinde 4618134 olan nokta sayısı 1587194 noktaya düşürülmüştür (Tablo 5). Bu uygulama sonucunda %65 oranında veri ayıklanmış ve azaltılmıştır.

2.4. Nokta Bulutu Verisinin Ön İşlenmesi

LiDAR verileri için MATLAB ortamında gerçekleştirilen "*pcdenoise"* fonksiyonu uygulaması bir gürültü filtreleme işlemidir. Noktaların birbirine olan yakınlığını yani komşuluğunu baz alarak yapmış olduğu bir filtreleme tekniğidir.

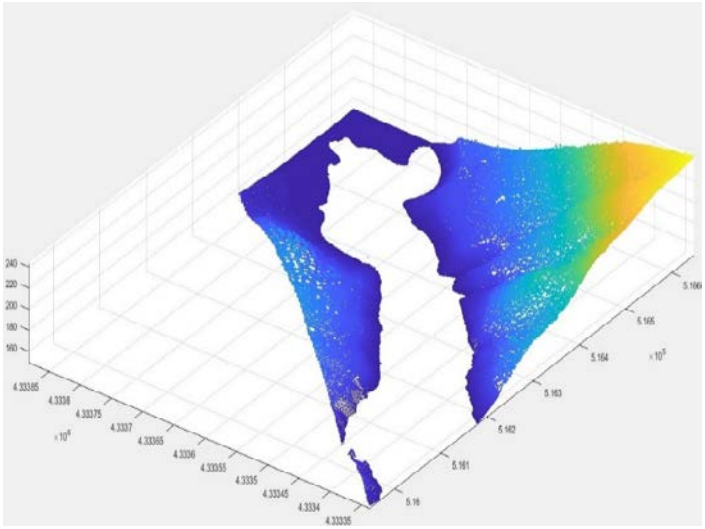
LiDAR tekniği uygulanırken kullanılan lazer ışınının yeryüzeyinden dönen bazı sinyalleri saçılarak düzensiz bir şekilde nokta bulutuna dâhil olabilmekte ve böylelikle bu veri kümesi içerisinde gürültü kümesi oluşmaktadır. Bu noktaları temizlemek için uygulanan "*pcdenoise*" fonksiyonu ile daha sağlıklı nokta bulutu elde edilmiştir. Akabinde "*downsample*" fonksiyonu ile mükerrir noktalarnı silinmesi için bir filtreleme işlemi gerçekleştirilir. Böylelikle aynı noktaları veri kümesinden silerek daha sade bir nokta bulutu elde edilmiştir (Şekil 19).



Şekil 19. Gürültülü ve gürültüsü giderilmiş veri gösterimi (Optech_1200_38 verisi)

Bu uygulama yapılırken dikkat edilmesi gereken husus, noktaların birbirine yakınlığına dayanan bir uygulama olduğu için uçuşların 1200 m ve 2600 m'den gerçekleştiği göz önünde bulundurulmalıdır. Söz konusu yüksekliklerden elde edilen nokta sıklığı 1200 m'de 8 nokta/m^2'ye düşerken, 2600 m'de 2 nokta/m^2'ye düşmektedir. Bu detay uygulamadaki eşik değerlerini değiştirdiği için dikkat edilmelidir. En yakın komşuluk ilişkisine dayanarak çözümleme yapan bu uygulama için noktaların birbirine olan yakınlığı için bir eşik değer belirlenir ('NumNeighbors',20 gibi) ve bu değere göre filtreleme işlemi uygulanır (Ek 2).

Uçuş yüksekliğine bağlı olarak nokta yoğunluğu da değiştiğinden noktaların birbirine olan yakınlıklarında farklılık olacaktır. Dolayısıyla "*pcdenoise*" uygulanırken bu kriter göz önünde bulundurulmalıdır. Bir diğer uygulama "*downsample*" ise mükerrir noktalar var ise bunları temizlemektedir. Tekrara giren nokta kalabalığının önüne geçerek veri hacminin büyümesini engellemektedir (Şekil 20).

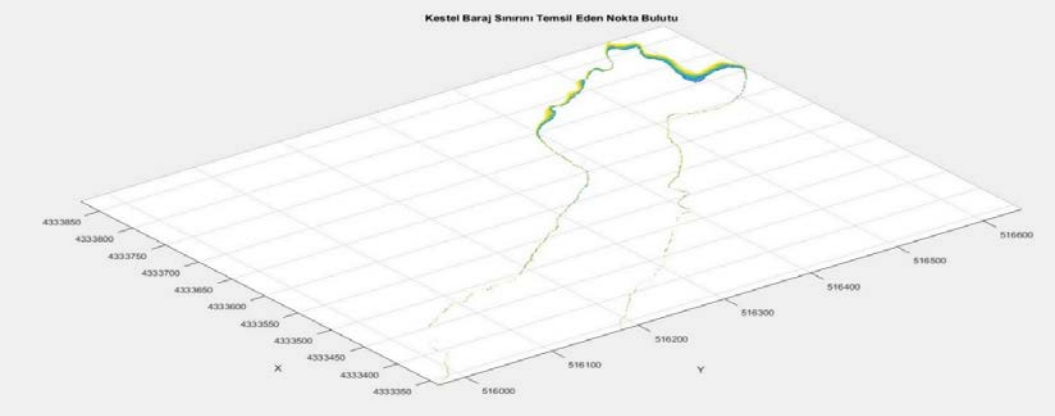


Şekil 20. "*pcdenoise*" ve "*downsample*" fonksiyonları uygulanmış veri

"*pcdenoise*" ve "*downsample*" fonksiyon uygulamalarını yapmadan önce nokta bulutunda bulunan veri sayısı 1587194 iken, bu ön işlem sonucu veri sayısı 1554658 noktaya düşmüştür (Tablo 5). Bu uygulamalar neticesinde %2 oranında veri ayıklanmış ve azaltılmıştır.

2.5. Çalışma Alanını Temsil Eden Noktaların Tespiti

Optech_1200_38 nokta bulutu verisi için yapılan filtreleme ve ön işlem uygulamaları sonucu elde edilen nokta kümesi içinden, Kestel Barajı rezervuar alanını temsil eden noktalar tespit edilmiştir. Sinyal yoğunluğu (intensity) filtrelemesi sonucu kara ve su ayırımını yaparak kıyı çizgisini oluşturan noktalarla çalışıldı. Akabinde ön işlem uygulamaları yapılarak gürültülü veri ve mükerrir noktalar ayıklandı. Kalan nokta bulutu verisinin yükseklik değerlerine bakıldığında en düşük yükseklik kıyı çizgisini meydana getirmektedir. Bu noktalara yaklaşık 20-30 cm düşey uzaklıkta yakın olan noktaların tespiti MATLAB ortamında kodlanarak (Ek 2) elde edilmiştir. Böylelikle Kestel Barajı rezervuar alanını temsil edecek noktalar belirlenmiş olup rezervuar alanın çizimi için bu noktalara yoğunluk sağlanmış olacaktır (Şekil 21).

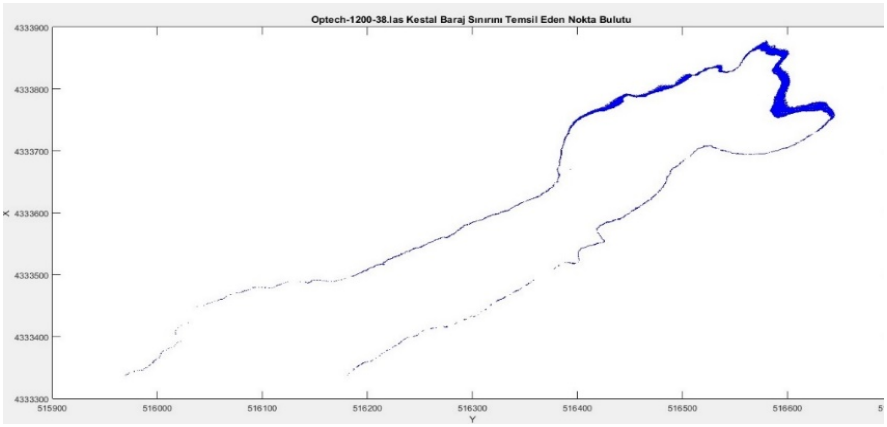


Şekil 21. Kestel Barajı rezervuar alanını temsil eden noktalar (Optech_1200_38 verisi)

Optech_1200_38 verisi için bulunan minimum su kotu ve 20-30 cm kot aralığına kadar oluşturulan Kestel Baraj sınırını temsil eden nokta bulutu 20560 nokta sayısına kadar düşürülmüştür (Tablo 5).

2.6. Sonuç Nokta Bulutunun Görüntülenmesi

Kestel_Optech_1200 verisini oluşturan 6 farklı *las* dosyası için yukarıda yapılan işlemler ayrı ayrı gerçekleştirildikten sonra tek bir matris halinde birleştirilmiştir. Bu işlem için bu aşamadan sonra bulunan sonuçlar ascii kodlu dosyaya txt uzantılı olarak kayıt edilmiştir (Ek 2). Bu veri MATLAB ortamında görsel hale getirilmiştir (Şekil 22).

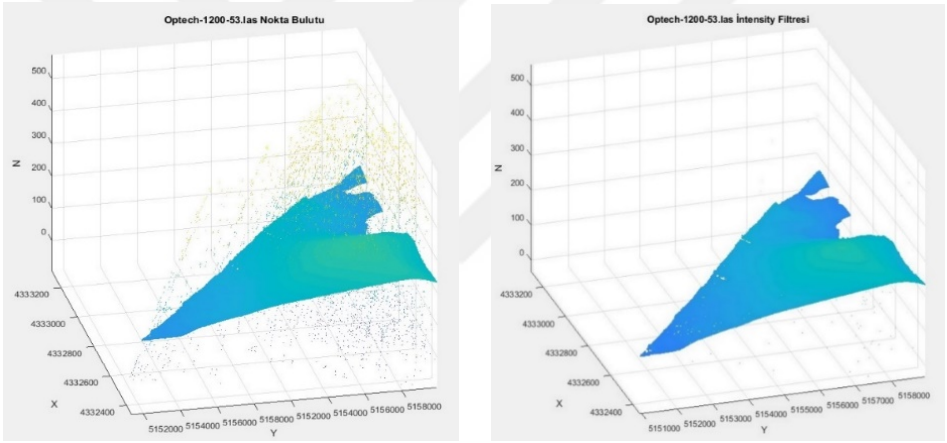


Şekil 22. Kestel Barajı rezervuar alanını temsil eden nokta bulutu (Optech_1200_38 verisi)

2.7. Optech_1200 Verisini Oluşturan Diğer Parçalara Uygulanan İşlemler

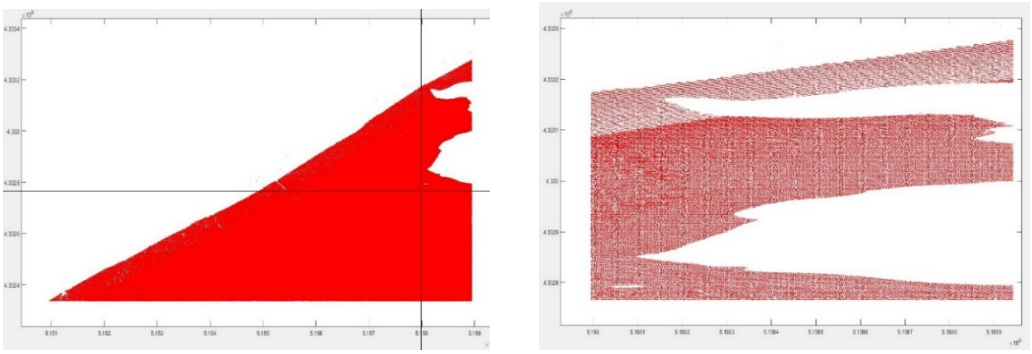
2.7.1. Optech_1200_53 Verisi

Optech_1200 verisini oluşturan ikinci kısım ise Optech_1200_53 nokta bulutu verisidir. Bu veri için de aynı filtrelemeler ve ön işlem uygulamaları gerçekleştirilerek Kestel Barajı rezervuar alanını temsil eden noktalar bulunmuştur. İlk işlem olarak sinyal yoğunluğu (intensity) filtrelemesi uygulanmıştır (Şekil 23). 3861122 nokta sayısına sahip olan Optech_1200_53 verisi intensity filtrelemesinden sonra 3160563 nokta sayısına inmiştir (Tablo 5). Bu filtreleme sonucunda %18 oranında veri ayıklanmıştır.



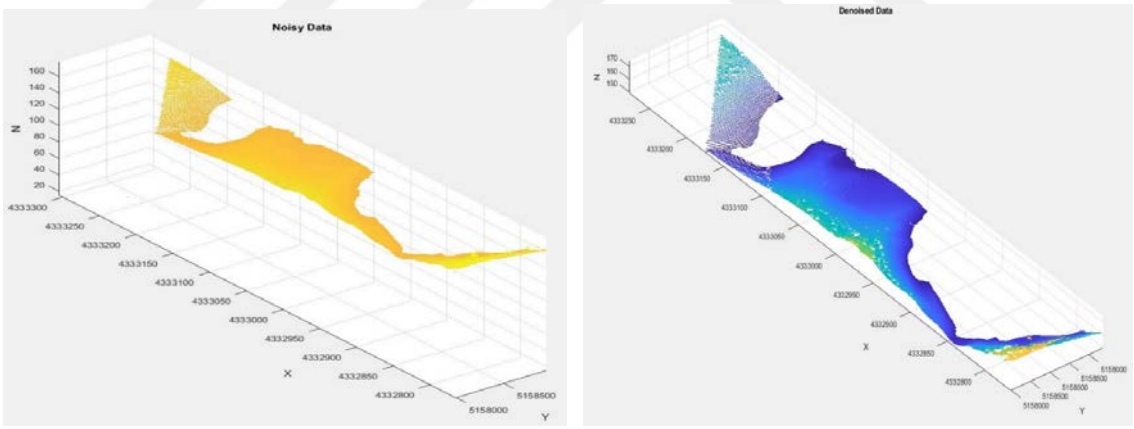
Şekil 23. Nokta bulutu ve intensity filtrelemesi (Optech_1200_53 verisi)

Optech_1200_53 verisi için uygulanan diğer bir işlem baraj rezervuar alanının dışında kalan noktaların silinmesi işlemidir. Bu işlem MATLAB ortamında "*ginput*" fonksiyonu ile yapılmaktadır (Şekil 24). İntensity filtrelemesi sonucu 3160563 noktaya düşen veri sayısı bu uygulama sonucunda 229656 olmuştur (Tablo 5). Bu uygulama ile %92 oranında veri azaltılmıştır.



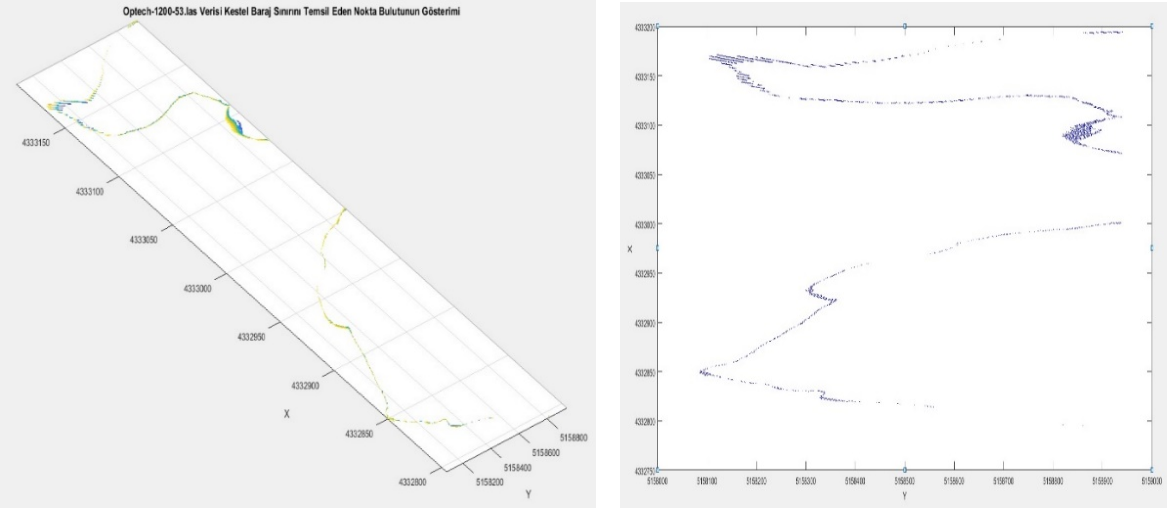
Şekil 24. “ginput” uygulama sonucu nokta bulutu (Optech_1200_53 verisi)

Nokta bulutlarını işlemeden önce yapılması gereken bir ön işlem uygulaması olan “*pcdenoise*” ve “*downsample*” uygulaması MATLAB ortamında gerçekleştirilmiştir (Şekil 25). Optech_1200_53 verisine ait nokta sayısı 229656 iken 209359 nokta sayısına gerilemiştir (Tablo 5). Bu uygulama neticesinde %8 oranında veri azaltılmıştır.



Şekil 25. “*pcdenoise*” ve “*downsample*” uygulaması öncesi ve sonrası (Optech_1200_53 verisi)

Optech_1200_53.las verisi için yapılan filtrelemeler ve ön işlem uygulamaları sonucunda baraj rezervuar alanını temsil edecek noktalar MATLAB ortamında uygun algoritma (Ek 3) ile kodlanarak tespit edilmiş ve kayıt altına alınmıştır (Şekil 26). Bu işlem ile 209359 olan nokta sayısı 3470 nokta sayısına ulaşmıştır (Tablo 5). Böylece %98 oranında veri ayıklanmıştır.

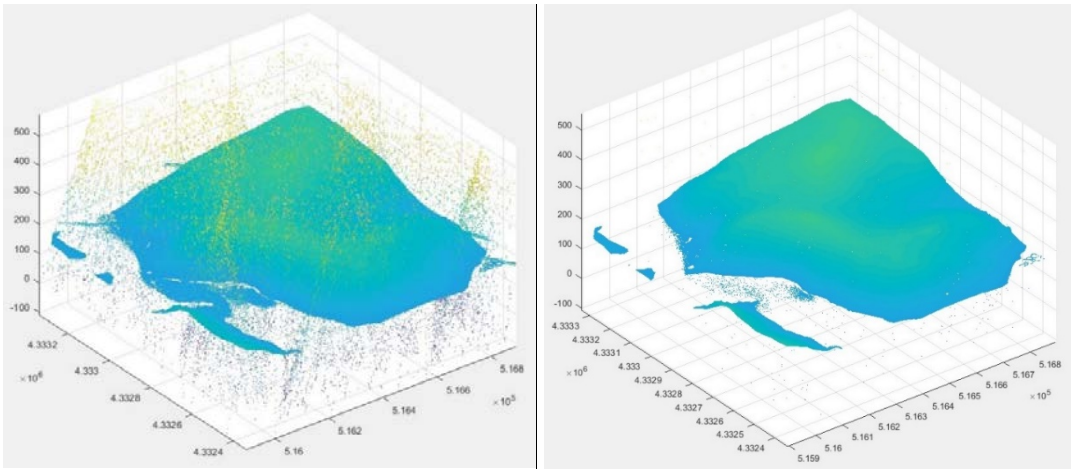


Şekil 26. Kestel Baraj sınırını temsil eden nokta bulutu görünümü (Optech_1200_53 verisi)

2.7.2. Optech_1200_54_1 Verisi

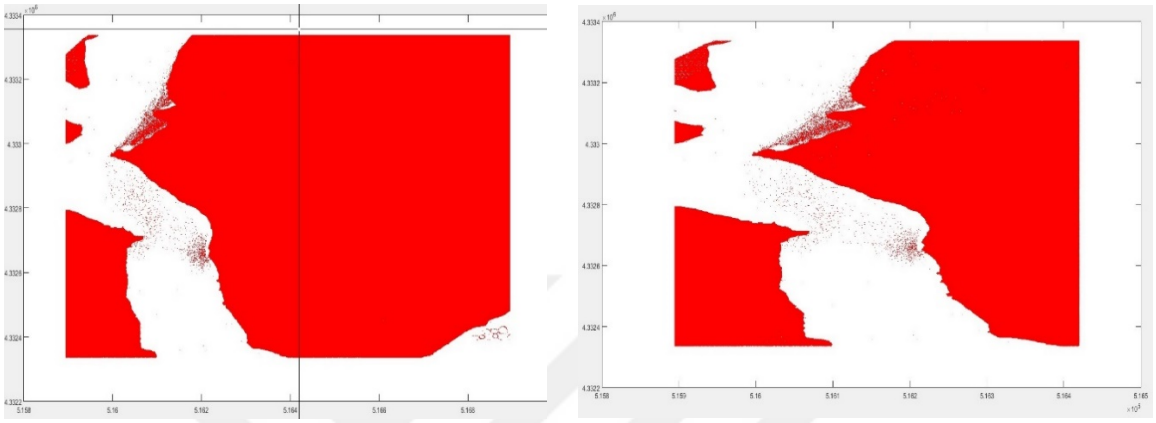
Optech_1200 nokta bulutu verisinin üçüncü kısmı olan Optech_1200_54 kendi içerisinde iki parçaya ayrılmıştır. Bunun sebebi baraj rezervuar alanı kendi içinde iki parça halinde göründüğü için bu kısmı bölerek değerlendirmek daha sağlıklı olacaktır.

Optech_1200_54 verisini iki parçaya bölmeden önce intensity filtrelemesi uygulanmıştır (Şekil 27). Bu işlem ilk başta 13706673 olan nokta sayısını 10905563 nokta sayısına düşürmüştür (Tablo 5). Bu uygulama ile %20 oranında veri ayıklanmıştır.



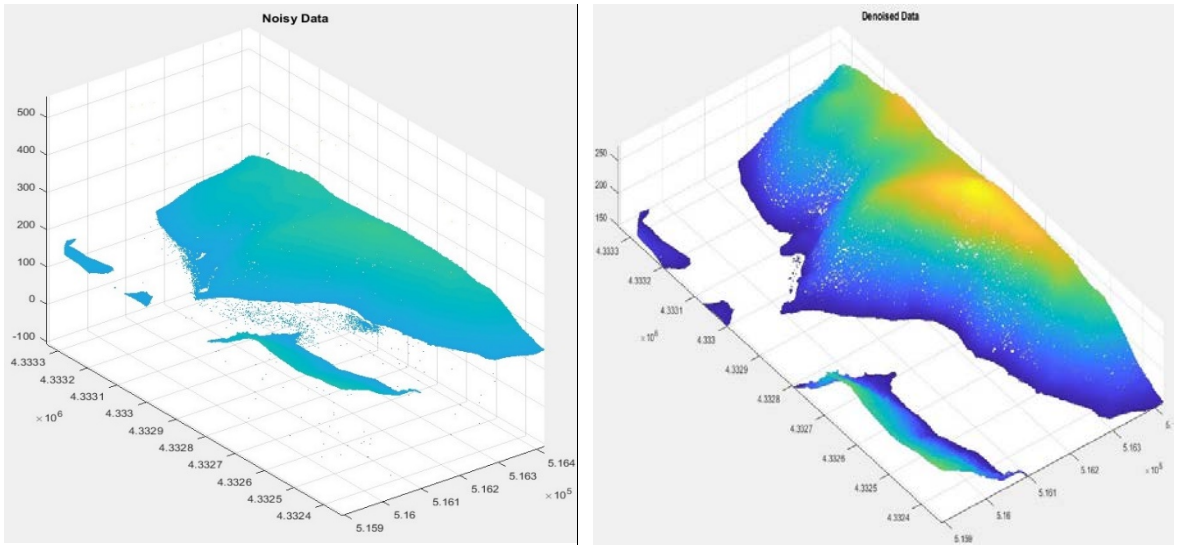
Şekil 27. Nokta bulutu ve intensity filtrelemesi (Optech_1200_54 verisi)

Optech_1200_54 verisi incelendiğinde Kestel Barajı rezervuar alanına ait iki farklı bölge olduğu görülmüştür. Bu alanları ayrı ayrı işlmelere tabi tutmak için "ginput" komutu kullanılarak Optech_1200_54_1 ve Optech_1200_54_2 şeklinde ikiye ayrılmıştır (Şekil 28). Bu uygulama ile 10905563 olan nokta sayısı 4185982 nokta sayısına gerilemiştir (Tablo 5). Yapılan işlem sonucunda %61 oranında veri ayıklanmıştır.



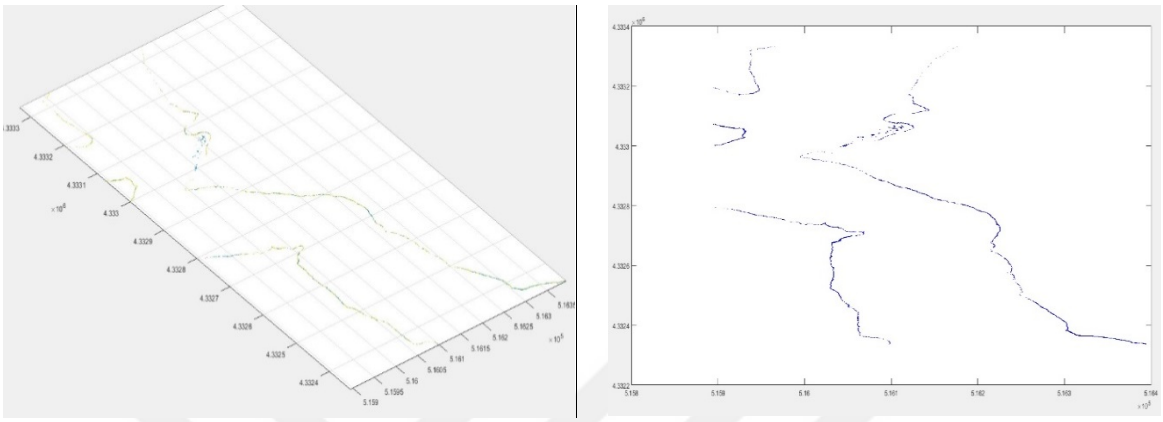
Şekil 28. "ginput" uygulama sonucu nokta bulutu (Optech_1200_54_1 verisi)

Optech_1200_54_1 verisi elde edildikten sonra "*pcdenoise*" ve "*downsample*" ön işlem uygulamaları yapılmıştır (Şekil 29). Bu işlemler neticesinde 4185982 olan nokta sayısı 4129353 nokta sayısına düşmüştür (Tablo 5). Böylece %2 oranında veri azaltılmıştır.



Şekil 29. "*pcdenoise*" ve "*downsample*" uygulaması öncesi ve sonrası (Optech_1200_54_1 verisi)

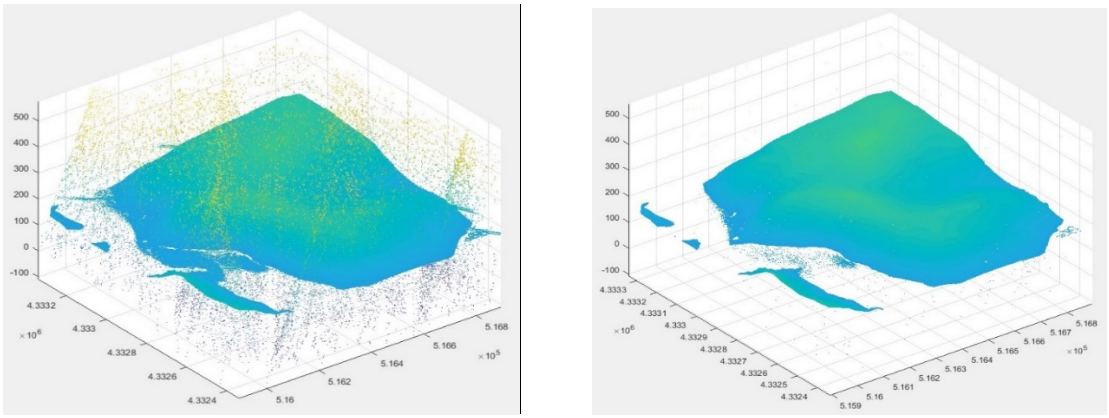
Optech_1200_54_1 verisine ait Kestel Barajı rezervuar alanını temsil eden noktalar MATLAB ortamında uygun algoritma (Ek 4) ile belirlenerek kayıt altına alınmıştır (Şekil 30). Bu işlem yapılırken 4129353 olan nokta sayısı 8321 olmuştur (Tablo 5). B uygulama neticesinde %99 oranında veri ayıklanmıştır.



Şekil 30. Kestel Baraj sınırını temsil eden nokta bulutu görünümü (Optech_1200_53_1 verisi)

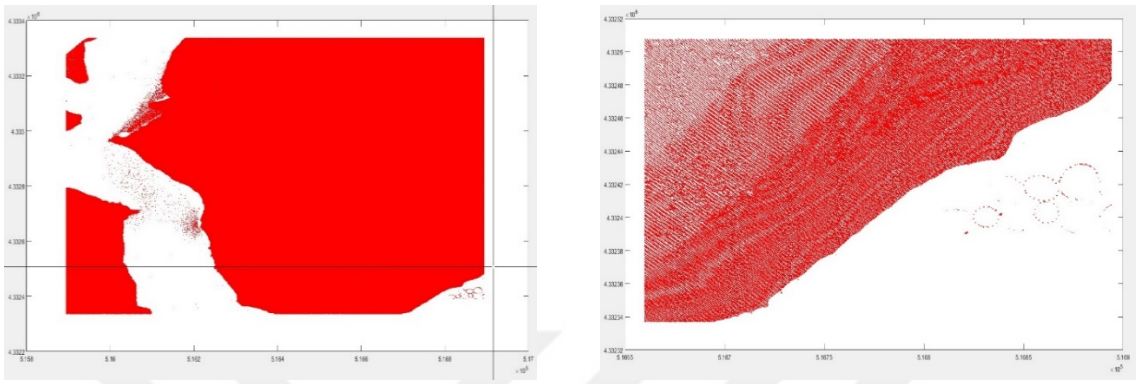
2.7.3. Optech_1200_54_2 Verisi

Optech_1200_54 verisine ait olan diğer kısım Optech_1200_54_2 verisi elde edilirken öncelikle Optech_1200_54 nokta bulutu verisi sinyal yoğunluğu (intensity) filtrelemesinden geçirilmiştir (Şekil 31). Optech_1200_54 verisine ait ilk kısma ait aynı işlem yapıldığı için sayısal değerleri aynıdır.



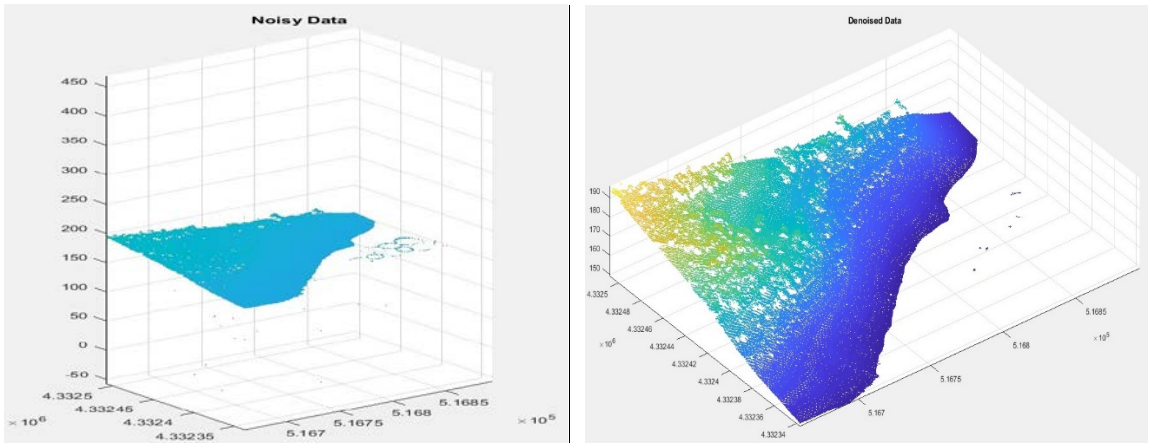
Şekil 31. Nokta bulutu ve intensity filtrelemesi (Optech_1200_54 verisi)

Optech_1200_54_2.las verisinin elde edildiği kısım ise MATLAB ortamında "ginput" komutu ile gerçekleştirilen uygulamadır (Şekil 32). Bu uygulama neticesinde 10905563 olan nokta sayısı 419755 olarak tespit edildi (Tablo 5). Bu sonuç itibariyle %96 oranında veri azaltılmıştır.



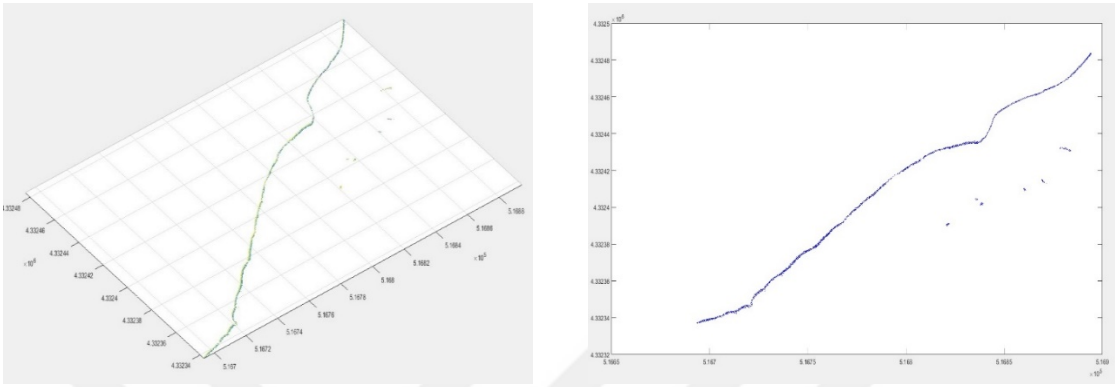
Şekil 32. "ginput" uygulama sonucu nokta bulutu (Optech_1200_54_2 verisi)

Optech_1200_54_2 verisi için yapılan ön işlem uygulaması MATLAB ortamında gerçekleştirilmiştir (Şekil 33). Bu işlem sonucunda 419755 olan nokta saysısı 416066 olarak azalmıştır (Tablo 5). Bu uygulama ile %1 oranında veri ayıklanmıştır.



Şekil 33. "*pcdenoise*" ve "*downsample*" uygulaması öncesi ve sonrası (Optech_1200_54_2 verisi)

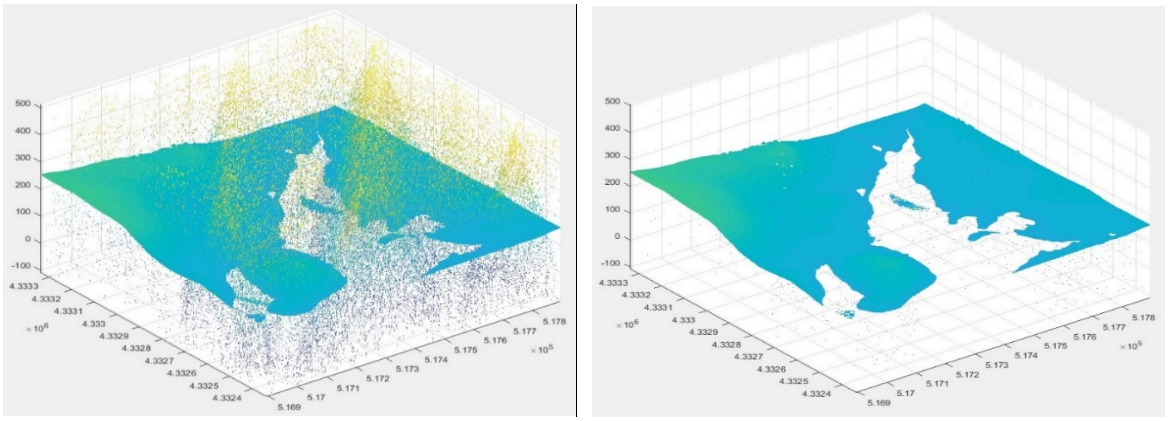
Optech_1200_54_2 verisine ait Kestel Barajı rezervuar alanını temsil eden noktalar MATLAB ortamında uygun algoritma (Ek 4) ile kodlanarak tespit edildi ve kayıt altında tutulmuştur (Şekil 34). Bu işlem sonucu 416066 olan nokta sayısı 2846 olarak tespit edilmiştir (Tablo 5). Böylece %99 oranında veri azaltılmıştır.



Şekil 34. Kestel Baraj sınırını temsil eden nokta bulutu görünümü (Optech_1200_53_2 verisi)

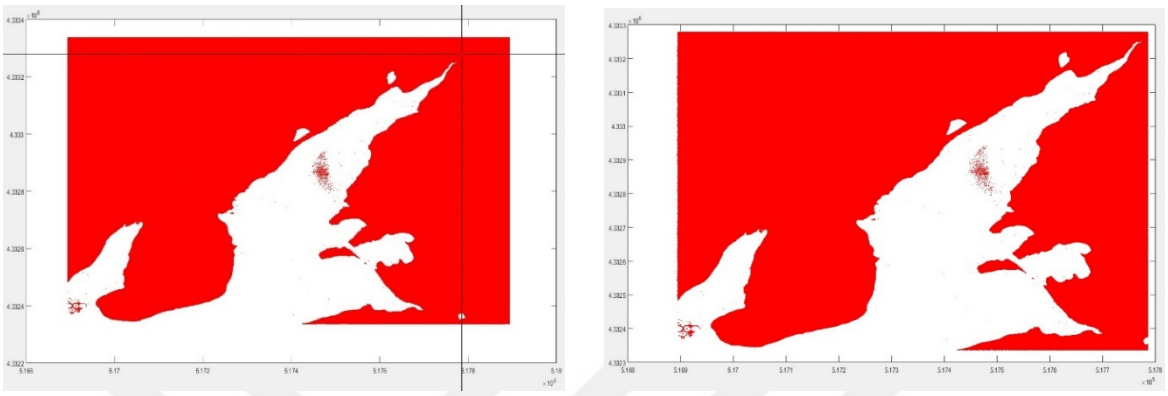
2.7.4. Optech_1200_55 Verisi

Optech_1200 verisine ait dördüncü kısım olan Optech_1200_5 verisine ilk olarak sinyal yoğunluğu (intensity) filtrelemesi uygulanmıştır (Şekil 35). Bu işlem ile 11775794 olan nokta sayısı 10635665 olarak gerilemiştir (Tablo 5). Bu uygulama sonucunda %10 oranında veri ayıklanmıştır.



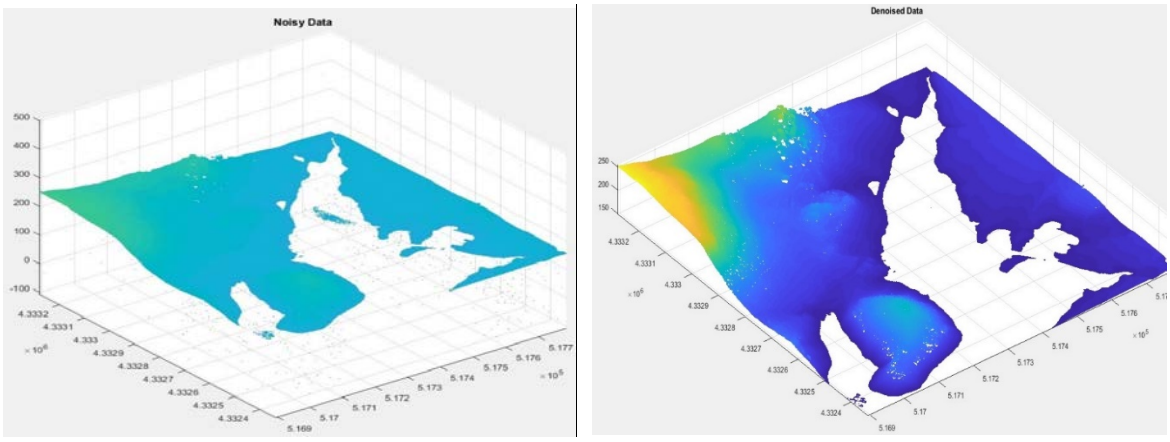
Şekil 35. Nokta bulutu ve intensity filtrelemesi (Optech_1200_55 verisi)

Optech_1200_55 verisine ait istenmeyen alanların silinmesi için MATLAB ortamında "ginput" komutu yardımıyla işlem yapılmıştır (Şekil 36). Bu uygulama ile 10635665 olan nokta sayısı 8431799 nokta sayısına gerilemiştir (Tablo 5). Böylece %20 oranında veri ayıklanmıştır.



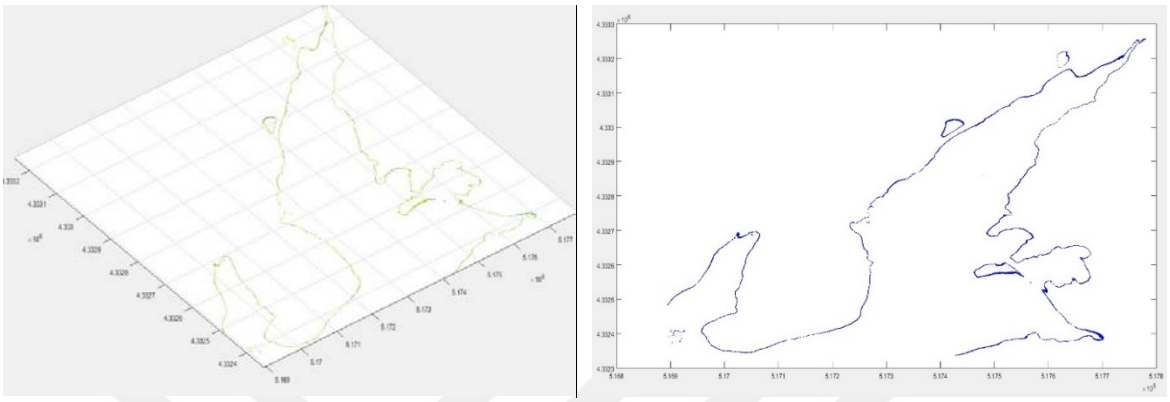
Şekil 36. "ginput" uygulama sonucu nokta bulutu (Optech_1200_55 verisi)

Optech_1200_55 verisine ait ön işlem süreci "pcdenoise" ve "downsample" uygulamarı yaparak devam etmiştir (Şekil 37). Bu işlem ile 8431799 olan nokta sayısı 8324078 nokta saysısına düşmüştür (Tablo 5). Bu uygulama ile %1 oranında veri azaltılmıştır.



Şekil 37. "*pcdenoise*" ve "*downsample*" uygulaması öncesi ve sonrası (Optech_1200_55 verisi)

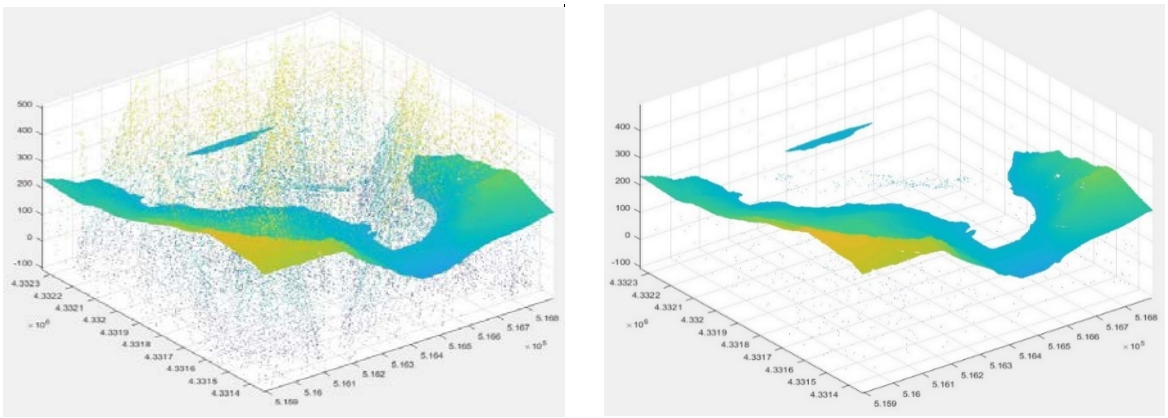
Optech_1200_55 verisi için Kestel Barajı rezervuar alanını temsil eden nokta bulutunun tespitinde MATLAB ortamında uygulanan işlem (Ek 5) kullanılmıştır ve kayıt altına alınmıştır (Şekil 38). Bu işlemle beraber 8324078 olan nokta sayısı 39732 olarak belirlenmiştir (Tablo 5). Böylelikle %99 oranında veri azaltılmıştır.



Şekil 38. Kestel Baraj sınırını temsil eden nokta bulutu görünümü (Optech_1200_55 verisi)

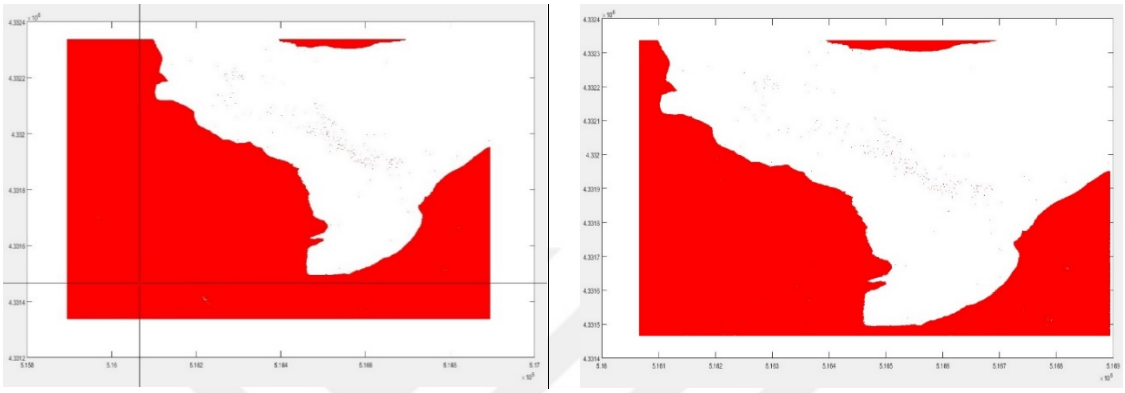
2.7.5. Optech_1200_70 Verisi

Optech_1200 verisine ait beşinci kısım olan Optech_1200_70 verisi için ilk olarak intensity filtrelemesi uygulanmıştır (Şekil 39). Bu filtreleme ile nokta sayısı 11002578 olan verinin 8595753 nokta saysısına gerilediği görülmüştür (Tablo 5). Bu uygulama ile %22 oranında veri ayıklanmıştır.



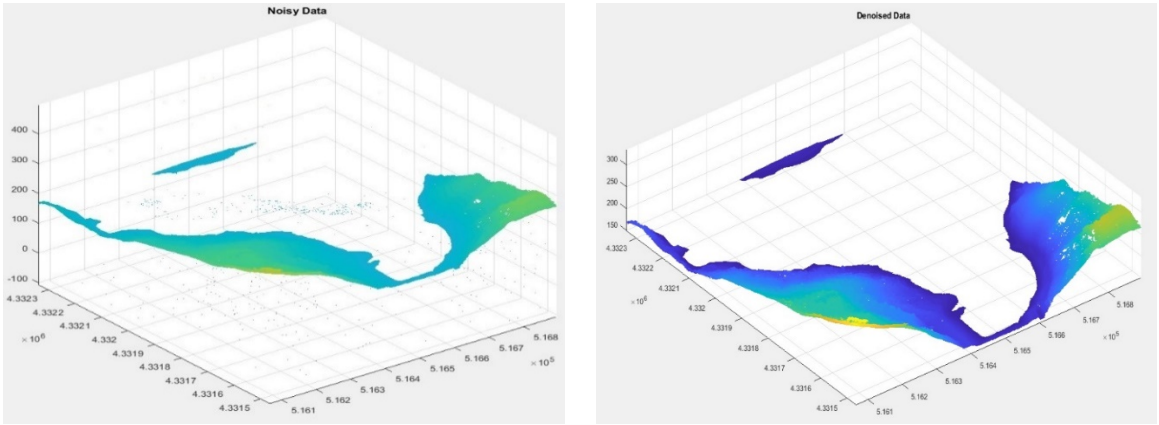
Şekil 39. Nokta bulutu ve intensity filtrelemesi (Optech_1200_70 verisi)

Optech_1200_70 verisine ait Kestel Barajı rezervuar alanını temsil etmeyen alanlar MATLAB ortamında "ginput" komutu yardımıyla silinmiştir (Şekil 40). Bu uygulama ile 8595753 olan nokta sayısı 4234644 nokta sayısına gerilemiştir (Tablo 5). Bu işlem itibariyle %50 oranında veri ayıklanmıştır.



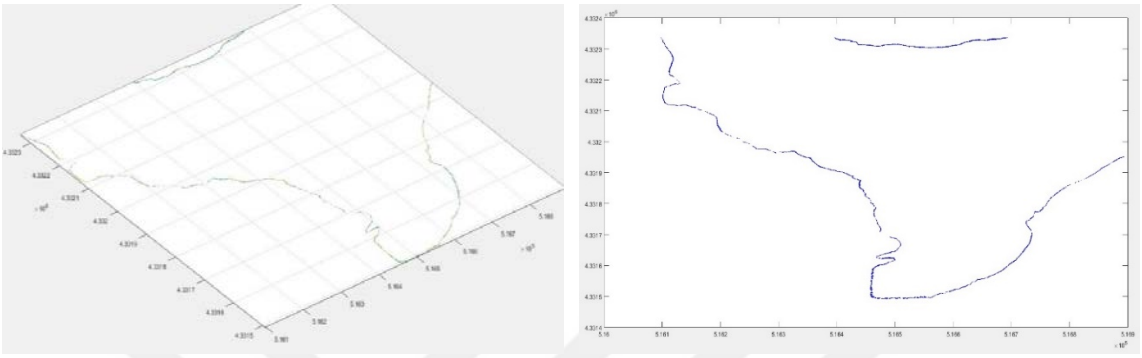
Şekil 40. "ginput" uygulama sonucu nokta bulutu (Optech_1200_70 verisi)

Optech_1200_70 nokta bulutu için yaplan ön işlem uygulaması "pcdenosie" ve "downsample" komutları ile MATLAB ortamında gerçekleştirildi (Şekil 41). Bu uygulama neticesinde 4234644 olan nokta sayısı 4182553 nokta sayısına düşmüştür (Tablo 5). Bu işlemle birlikte %12 oranında veri ayıklanmıştır.



Şekil 41. "*pcdenoise*" ve "*downsample*" uygulaması öncesi ve sonrası (Optech_1200_70 verisi)

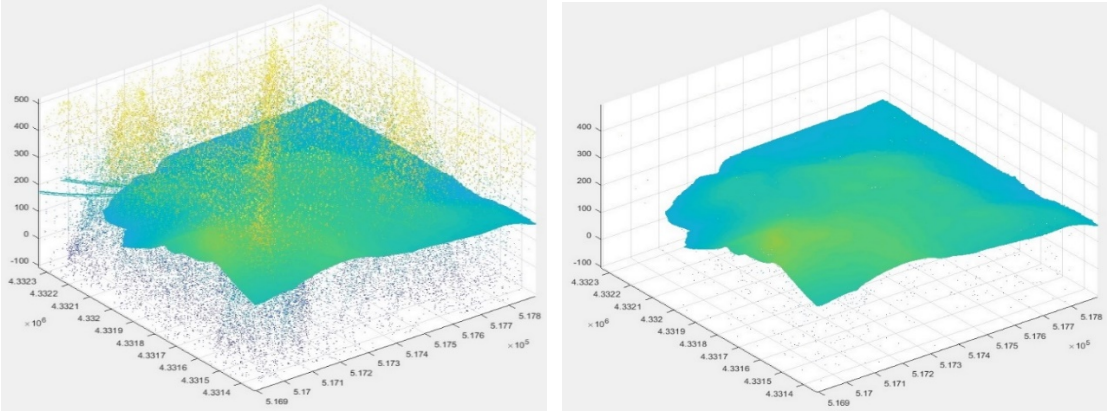
Optech_1200_70 verisine ait elde edilen nokta bulutu içerisinden Kestel Barjı rezervuar alanını temsil eden noktalar MATLAB ortamında uygulanan uygun algoritma ve kodlama (Ek 6) ile tespit edilmiştir (Şekil 42). Bu işlem 4182553 nokta sayısına sahip olan veriyi 16926 nokta sayısına düşürmüştür (Tablo 5). Bu uygulama ile %99 oranında veri azaltılmıştır.



Şekil 42. Kestel Baraj sınırını temsil eden nokta bulutu görünümü (Optech_1200_70 verisi)

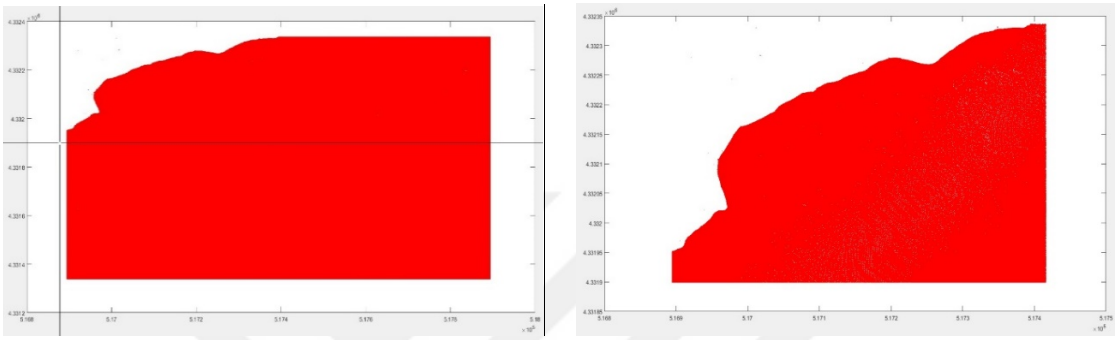
2.7.6. Optech_1200_71 Verisi

Optech_1200 verisine ait altıncı kısım olan Optech_1200_71 verisine öncelikle sinyal yoğunluğu (intensity) filtrelemesi uygulayarak, kara ve su ayrımının iyi şekilde yapılmasına dikkat edilmiştir (Şekil 43). Bu işlemle beraber 17546476 olan nokta sayısı 13659264 nokta sayısına düşmüştür (Tablo 5). Bu uygulama ile %22 oranında veri ayıklanmıştır.



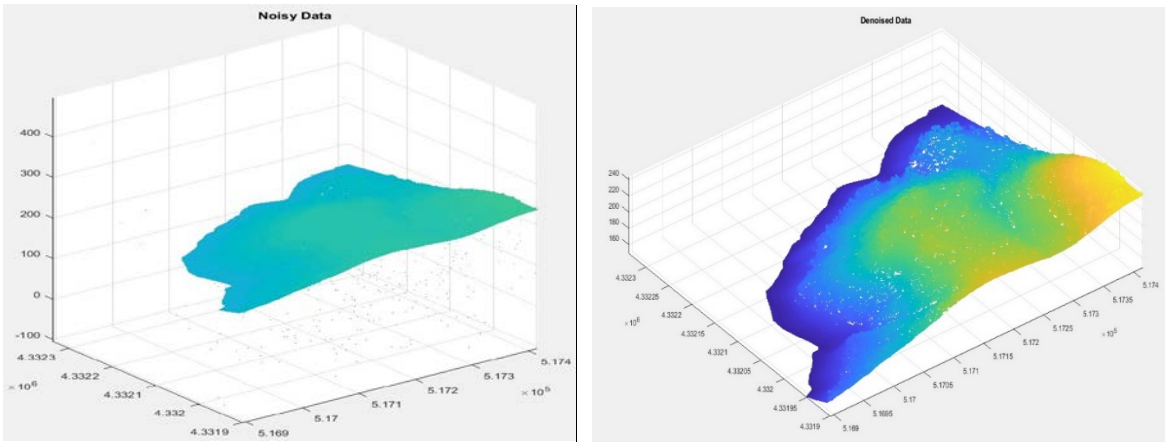
Şekil 43. Nokta bulutu ve intensity filtrelemesi (Optech_1200_71 verisi)

Optech_1200_71 verisine ait rezervuar alanının dışında kalan, işleme dahil edilmeyen alanların silinmesi için MATLAB ortamında "ginput" komutu ile uygulama gerçekleştirilmiştir (Şekil 44). Bu uygulama ile 13659264 olan nokta sayısı 2322591 nokta sayısına gerilemiştir (Tablo 5). Böylelikle uygulama sonucu %82 oranında veri ayıklanmıştır.



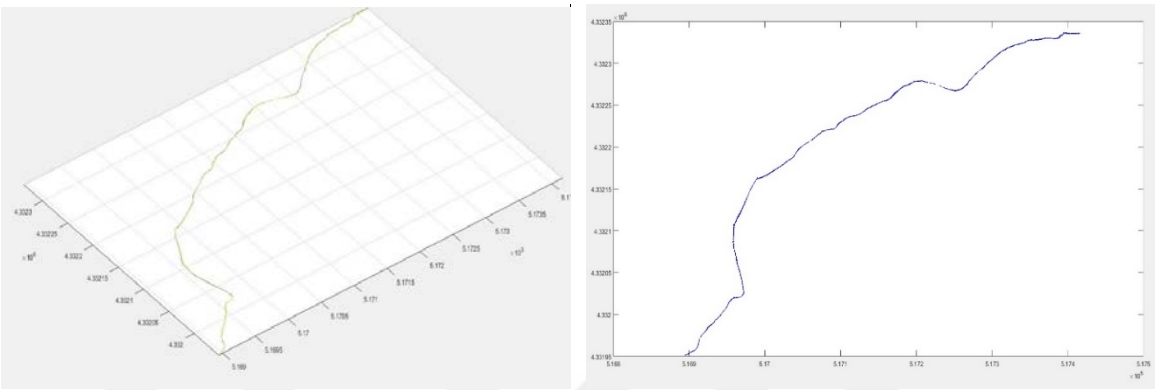
Şekil 44. "ginput" uygulama sonucu nokta bulutu (Optech_1200_71 verisi)

Optech_1200_71.las verisine ait ön işlem süreci "pcdenoise" ve "downsample" komutları ile MATLAB ortamında gerçekleştirilmiştir (Şekil 45). Bu işlem sonucunda 2322591olan nokta sayısı 2281512 nokta sayısına kadar azalmıştır (Tablo 5). Bu sonuç itibariyle %2 oranında veri ayıklanmıştır.



Şekil 45. "*pcdenoise*" ve "*downsample*" uygulaması öncesi ve sonrası (Optech_1200_71 verisi)

Optech_1200_71 verisine ait nokta bulutu içerisinden Kestel Barajı rezervuar alanını temsil eden verileri elde etmek için MATLAB ortamında uygun algoritma ve kodlama (Ek 7) ile tespit edilmiştir (Şekil 46). Bu uygulama ile 2281512 olan nokta sayısı 6224 nokta sayısına gerilemiştir (Tablo 5). Bu sonuç ile %99 oranında veri ayıklanmıştır.



Şekil 46. Kestel Baraj sınırını temsil eden nokta bulutu görünümü (Optech_1200_71 verisi)

Optech_1200 verisine ait 63954473 nokta sayısı ile çalışılmaya başlanmış olup, gerekli filtreleme işlemleri ve nokta bulutu verileri için gerekli ön işlemler sonucu Kestel Barajına ait su kotunu temsil eden 98074 adet nokta tespit edilmiştir (Tablo 5).

Optech_1200 verisi için yapılan işlemlerin aynısı diğer LiDAR verileri olan Optech_2600, Riegl_1200 ve Riegl_2600 için de yapılmış ve diğer tablolarda (Tablo 6, 7, 8) ifade edilmiştir.

Tablo 5. Optech_1200 verisine ait nokta sayısının değişim tablosu

Optech_1200	Nokta Bulutu	İntensity Filtrelemesi	ginput Uyg.	pcdenoise ve downsample Uyg.	Su Kotu Aralığı
Optech_1200_38	6061830	4618134	1585142	1553636	20555
Optech_1200_53	3861122	3160563	229656	209359	3470
Optech_1200_54	13706673	10905563	4605737	4545419	11167
Optech_1200_55	11775794	10635665	8431799	8324078	39732
Optech_1200_70	11002578	8595753	4234644	4182553	16926
Optech_1200_71	17546476	13659264	2322591	2281512	6224
	Σ=63954473	**Σ= 51574942**	**Σ=21409569**	**Σ=21096557**	**Σ= 98074**

Optech_2600 verisine ait 40738128 nokta sayısı ile çalışılmaya başlanmış olup, gerekli filtreleme işlemleri ve nokta bulutu verileri için gerekli ön işlemler sonucu Kestel Barajına ait su kotunu temsil eden 100722 adet nokta tespit edilmiştir (Tablo 6)

Tablo 6. Optech_2600 verisine ait nokta sayısının değişim tablosu

Optech_2600	Nokta Bulutu	İntensity Filtresi	ginput Uyg.	pcdenoise ve downsample Uyg.	Su Kotu Aralığı
Optech_2600_178	6819761	4363326	1244449	1028871	10033
Optech_2600_179	5493875	3818468	1965441	1822971	22598
Optech_2600_185	7151405	4727590	483686	230877	5604
Optech_2600_186	4709983	2509608	2509608	2082780	23795
Optech_2600_199	6182102	3873793	64311	57287	1295
Optech_2600_200	6594430	3957975	2012748	1362654	12018
Optech_2600_208	3786572	2232177	162561	134472	5379
	Σ=40738128	**Σ=25482937**	**Σ=8442804**	**Σ=6719912**	**Σ= 80722**

Riegl_1200 verisine ait 101512926 nokta sayısı ile çalışılmaya başlanmış olup, gerekli filtreleme işlemleri ve nokta bulutu verileri için gerekli ön işlemler sonucu Kestel Barajına ait su kotunu temsil eden 119385 adet nokta tespit edilmiştir (Tablo 7).

Tablo 7. Riegl_1200 verisine ait nokta sayısının değişim tablosu

Riegl_1200	Nokta Bulutu	İntensity Filtresi	ginput Uyg.	pcdenoise ve downsample Uyg.	Su Kotu Aralığı
Riegl_1200_83	8363206	4181828	1868083	1636798	7341
Riegl_1200_84	18261733	10737472	251500	233620	8553
Riegl_1200_102	13177844	8000057	5060847	4428273	17959
Riegl_1200_103	14929772	10122486	6640002	5910312	29621
Riegl_1200_104	16210243	10959664	2957246	2768500	12527
Riegl_1200_121	17440233	10904489	2278402	2089976	10674
Riegl_1200_122	13129895	8771166	5136301	4701695	13710
	Σ=101512926	**Σ=63677162**	**Σ=24192381**	**Σ=21769174**	**Σ=100385**

Riegl_2600 verisine ait 31853705 nokta sayısı ile çalışılmaya başlanmış olup, gerekli filtreleme işlemleri ve nokta bulutu verileri için gerekli ön işlemler sonucu Kestel Barajına ait su kotunu temsil eden 84404 adet nokta tespit edilmiştir (Tablo 8).

Tablo 8. Riegl_2600 verisine ait nokta sayısının değişim tablosu

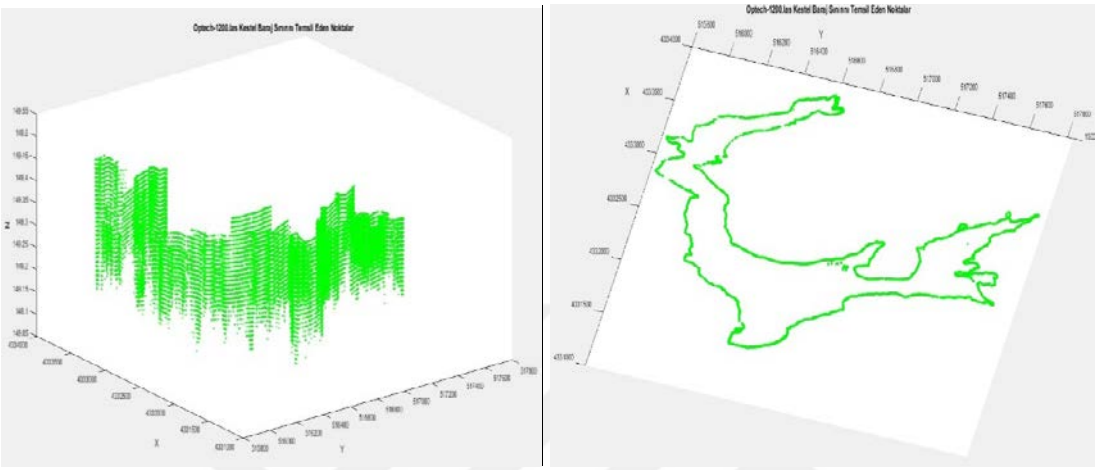
Riegl_2600	Nokta Bulutu	İntensity Filtresi	ginput Uyg.	pcdenoise ve downsampleUyg.	Su Kotu Aralığı
Riegl_1200_89	3070747	2361702	1382	1057	566
Riegl_1200_108	3244678	2071728	1025129	885795	8568
Riegl_1200_109	4994230	3423931	166731	144313	9554
Riegl_1200_110	4167762	3411293	551038	326804	5270
Riegl_1200_128	4920017	3129592	1252388	1020735	9940
Riegl_1200_129	3291306	2245981	2245981	1815499	35271
Riegl_1200_130	3869563	2909597	258613	252910	10880
Riegl_1200_149	4295402	3224593	175265	165652	4355
Σ=31853705		**Σ=22778417**	**Σ=5676527**	**Σ=4612765**	**Σ= 84404**

2.8. Sonuçların "*.las" Formatına Dönüştürülmesi ve Birleştirilmesi

Kestel Barajı rezervuar alanını tespit etmek için elde edilen sonuçlar **.txt* uzantılıdır. Bu veri türlerini, Martin İsenburg tarafından geliştirilen Lastools programına ait "*txt2las*" fonksiyonu kullanılarak **.las* formatına dönüşümü sağlanmıştır. İstenilen format elde edildikten sonra sonraki süreç MATLAB ortamında devam etmiştir.

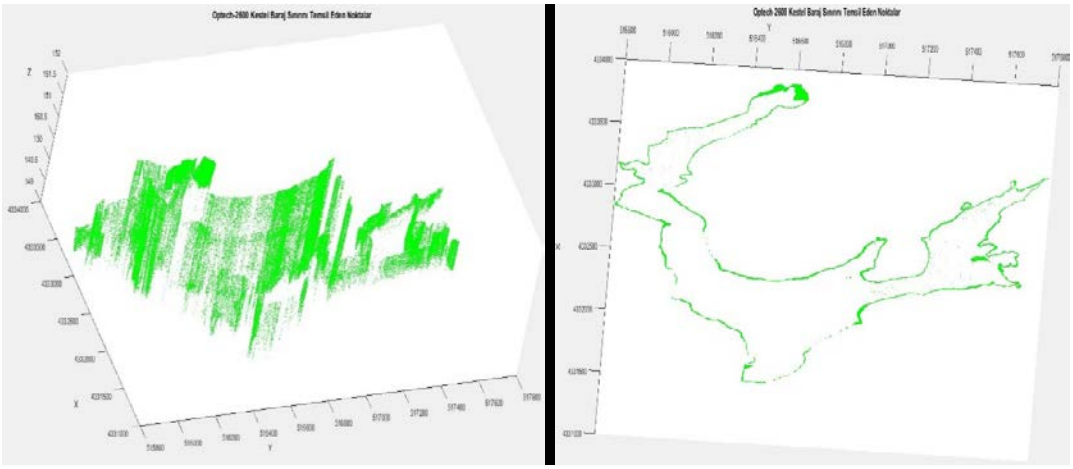
Kestel Barajı rezervuar alanı tespiti için şimdiye kadar yapılan uygulamaların sonuç kısımlarını MATLAB ortamında birleştirerek sonrasında sınır çıkarımı için gerekli olan işlemlere geçilmiştir. Kestel barajına ait elde edilen dört farklı nokta bulutu ile birleştirme işlemi gerçekleştirilerek rezervuar alanını tespit etmek için her veri grubuna aynı işlemler uygulanmış ve sınır çıkarımı sağlanmıştır (Ek 8).

Optech_1200 nokta bulutu verisine ait Kestel Barajı rezervuar alanını temsil eden nokta bulutu kısım kısım elde edildikten sonra MATLAB ortamında birleştirilerek üç boyutlu ve iki boyutlu gösterimi yapılmıştır (Şekil 47).



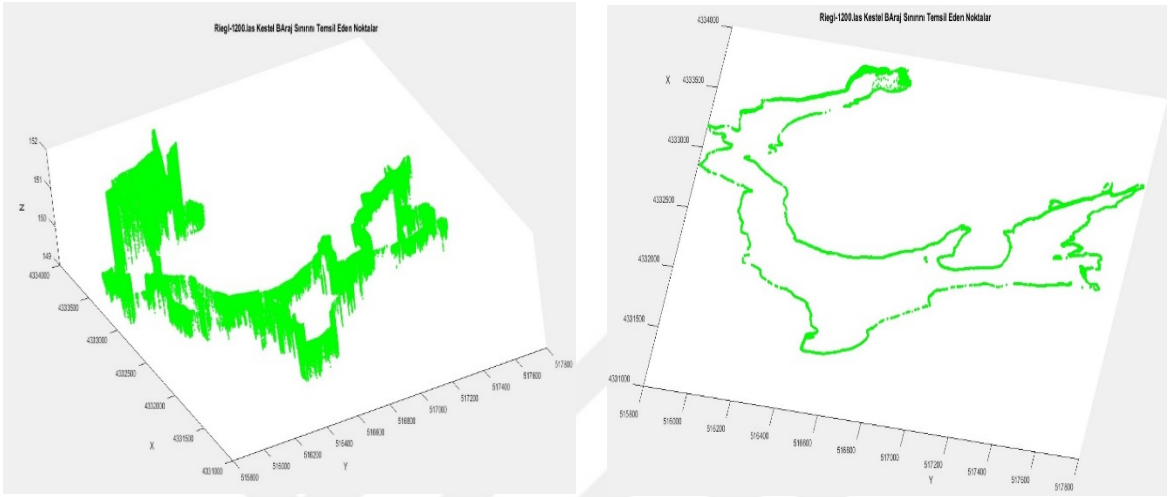
Şekil 47. Optech_1200 verisine ait Kestel Baraj sınırını temsil eden noktalar

Optech_26000 verisini oluşturan kısımlar MATLAB ortamında birleştirilerek üç boyutlu ve iki boyutlu görünümü sağlanmıştır (Şekil 48).



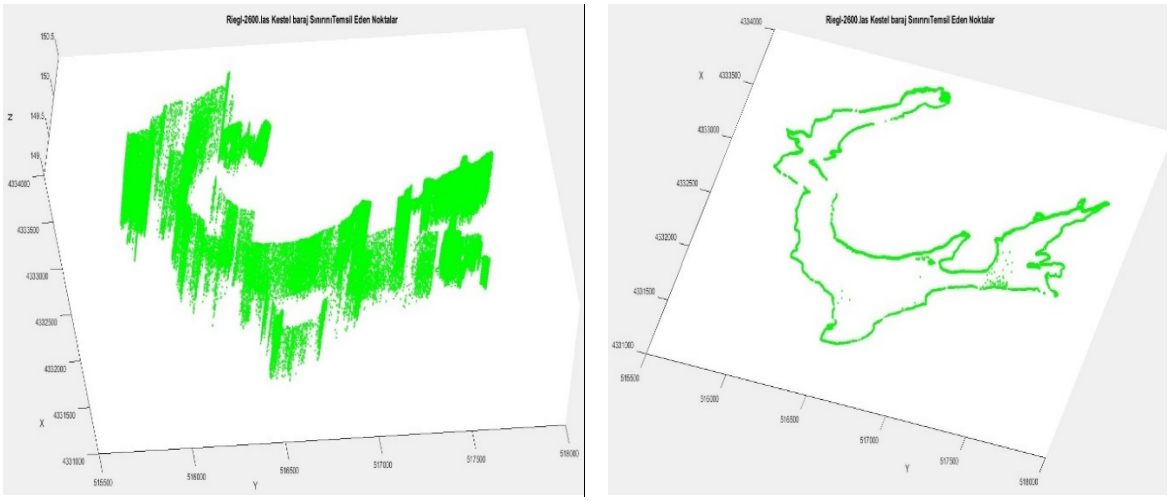
Şekil 48. Optech_2600 verisine ait Kestel Baraj sınırını temsil eden noktalar

Riegl_1200 verisini oluşturan kısımlar için de Optech verileri için yapılan işlemler uyulandıktan sonra MATLAB ortamında birleştirilerek üç boyutlu ve iki boyutlu gösterimi yapılmıştır (Şekil 49).



Şekil 49. Riegl_1200 verisine ait Kestel Baraj sınırını temsil eden noktalar

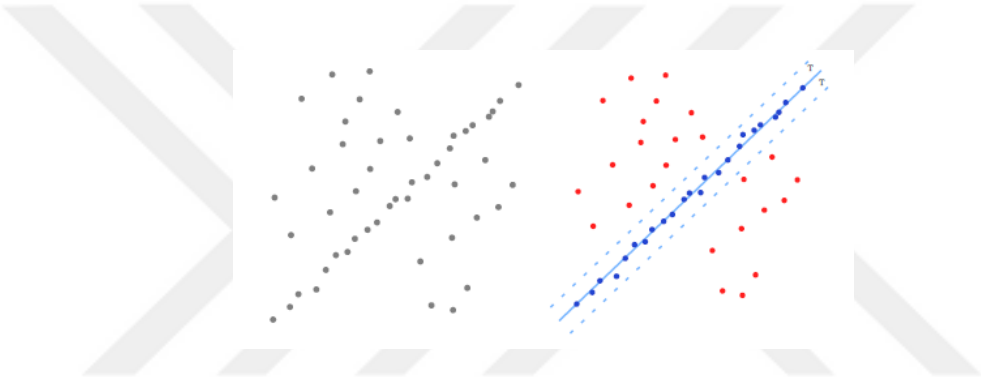
Riegl_2600 verisini meydana getiren parçalar MATALB ortamında bir araya getirilerek üç boyutlu ve iki boyutlu görünümü sağlanmıştır (Şekil 50).



Şekil 50. Riegl_2600 verisine ait Kestel Baraj sınırını temsil eden noktala

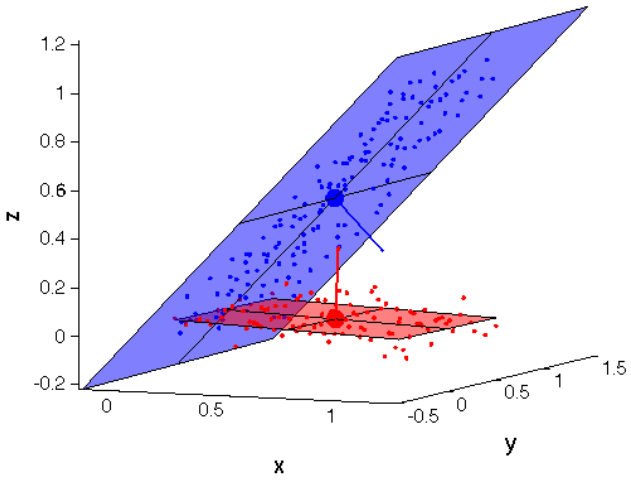
2.9. Kestel Barajı Rezervuar Alanının Tespit Edilmesi

Kestel Barajı rezervuar alanını temsil eden noktalar tespit edildikten sonra bu nokta bulutu içerisinde RANSAC algoritması prensiplerine dayanarak MATLAB ortamında "*pcfitplane*" uygulaması yapılmıştır. Kestel Baraj sınırını temsil eden nokta bulutu içerisinde yükseklik verisi referans alınarak düzgün bir dağılım olacak şekilde bir yüzey belirlenmiştir. Buradaki amaç Kestel Baraj sınırını temsil eden nokta bulutu içerisinden su kotuna en yakın bir düzlem oluşturularak, bu düzlemi oluşturan nokta bulutu içerisinden kıyı çizgisine ait noktaların tespiti amaçlanmıştır(Şekil 51).



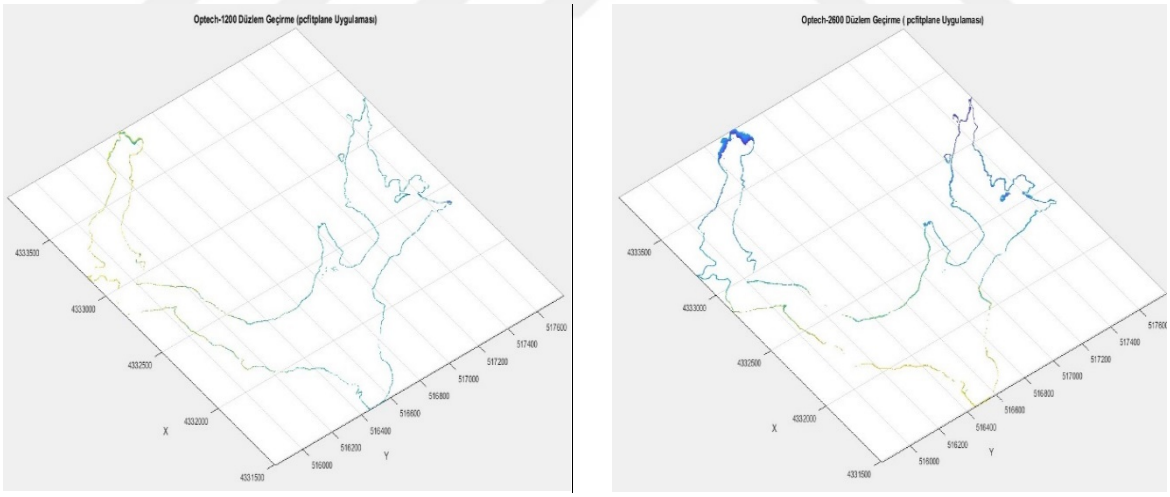
Şekil 51. RANSAC algoritmasının çalışma prensibi

MATLAB ortamında gerçekleştirilen "*pcfitplane*" uygulamasında referans vektörü olarak belirlediğimiz yükseklik verisi önemli bir etkendir. Yükseklik verilerini referans alarak oluşruracağı yüzeyde iki de önemli eşik değeri vardır. Bunlar yüzeyi meydana getirecek noktaların bu referans yüzeyine olan öklid uzaklığı ve açısal mesafesidir (Şekil 52). Bu iki eşik değerini belirleyerek oluşturmuş olduğumuz modelde söz konusu nokta bulutu içerisinde bir düzlem oluşturulmuştur (Ek 8). Bu düzlem sayesinde elde edilen noktalar ile baraj rezervuar alanının tespiti daha sağlıklı olmuştur.



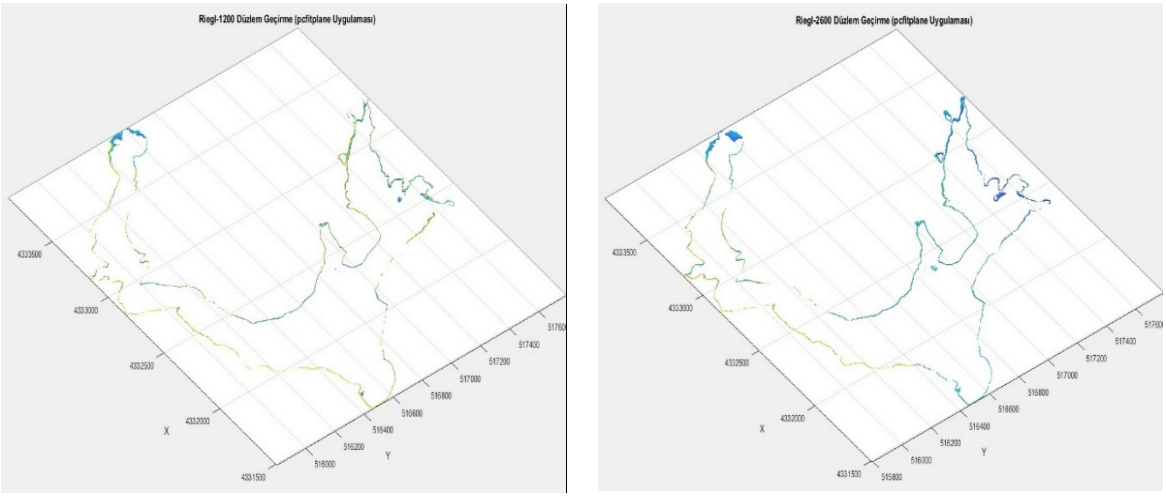
Şekil 52. "*pcfitplane*" uygulamasına ait temsili gösterim

Optech_1200 ve Optech_2600 verileri için uygulanan "*pcfitplane*" modeli ile elde edilen düzlem ve bu düzlemi oluşturan noktalar MATLAB ortamında elde edilerek, gösterimi sağlanmıştır (Şekil 53)



Şekil 53. Optech_1200 ve Optech_2600 verilerine pcfitplane uygulanması

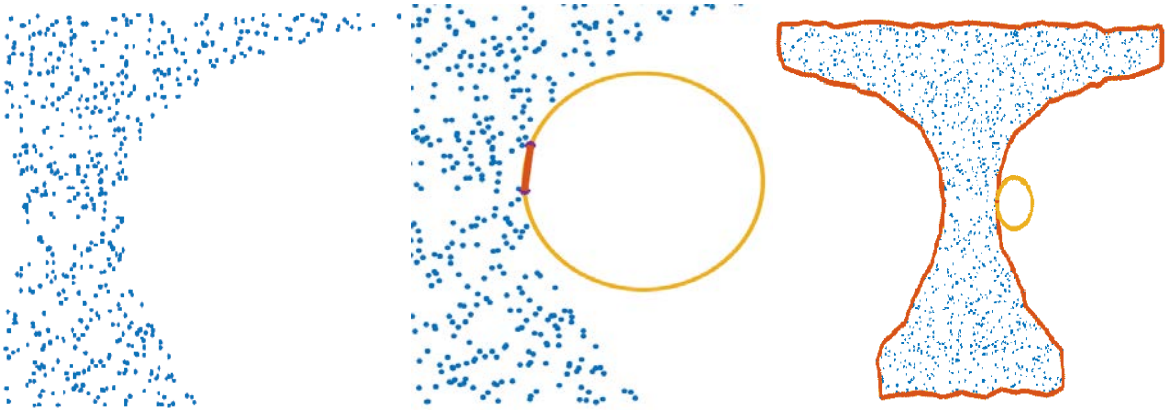
Riegl_1200 ve Riegl_2600 verileri için baraj rezevuar alanını tespit etmek için kullanılan düzlem MATLAB ortamında "pcfitplane" modeli ile sağlanmıştır. Elde edilen düzlem ve bu düzlemi oluşturan noktaların gösterimi MATLAB ortamında gerçekleştirilmiştir (Şekil 54).



Şekil 54. Riegl_1200 ve Riegl_2600 verilerine pcfitplane uygulanması

2.10. Kestel Barajı Rezervuar Alanına Ait Sınır Çıkarımı

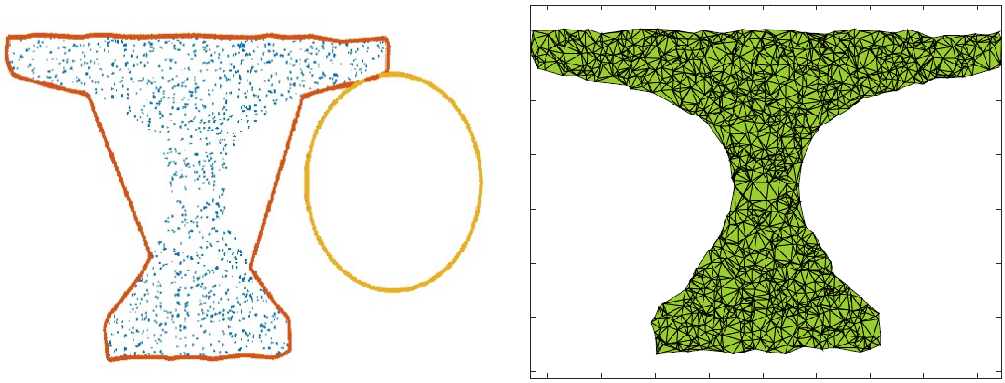
MATLAB ortamında "*alpha shapes*" uygulaması, nokta bulutu içindeki noktaların belli bir çap (radius) değeri ile tespit edilerek belirlenen üçgenleme işlemidir. Bu işlemin ardından çizilen üçgenlerin en dışından sınır geçirilerek söz konusu alana ait sınır verisini tayin eden bir yöntemdir (Şekil 55).



(a) (b) (c)

Şekil 55. "alpha shapes" uygulaması; a) Nokta bulutu verisi b) Nokta komşuluğuna göre belirlenen çap değeri c) Belirlenen çap değeriyle elde edilen şekil d) Farklı çap değeri ile elde edilen şekil e) "alpha shapes" uygulaması sonucu elde edilen üçgenleme

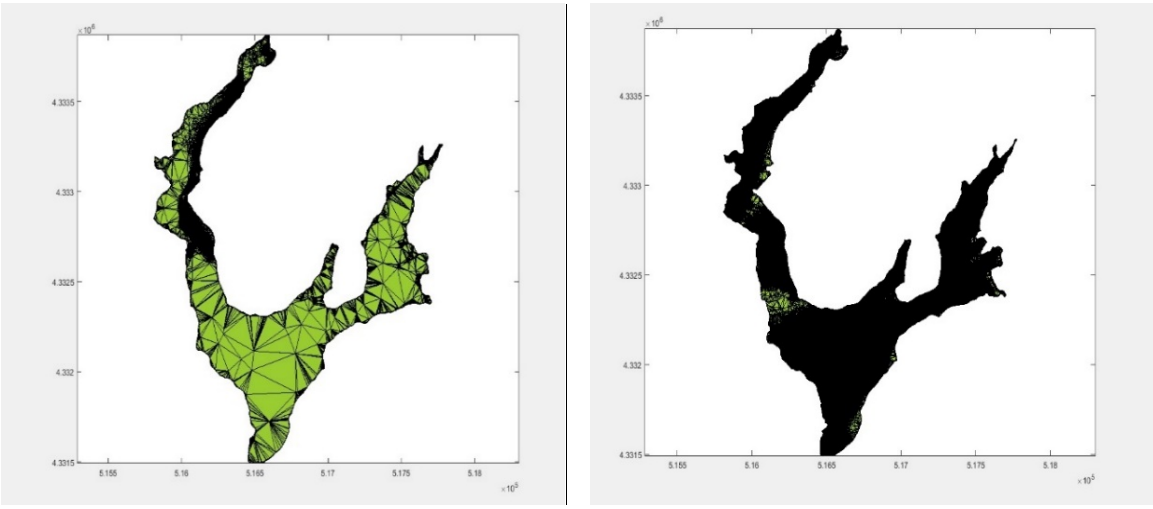
Şekil 55'in devamı



(d) (e)

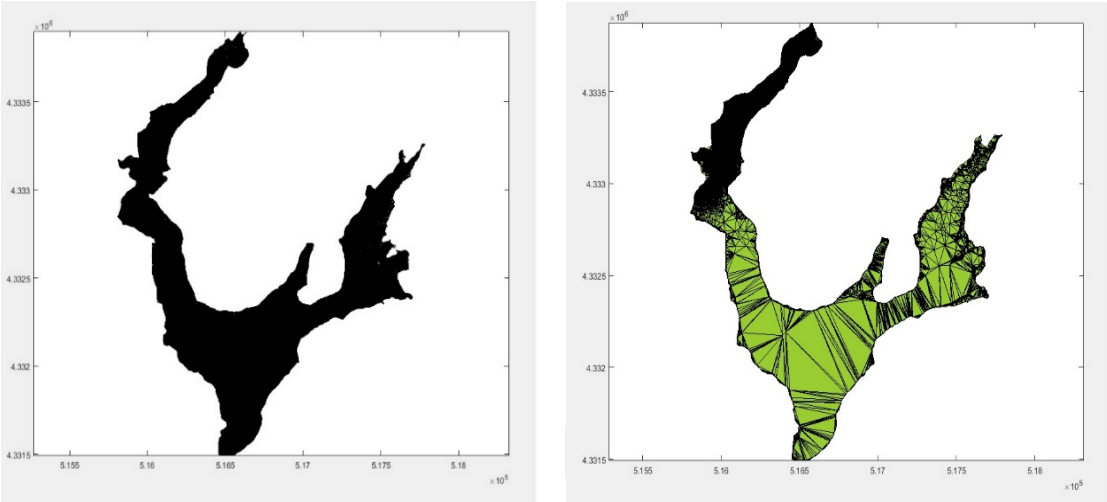
Bu ugulama gerçekleştirilirken dikkat edilen hususlar, üçgenleme yapılacak noktaları belirleyen çap değerini belirlemek, üçgenleme yapılacak alanın büyüklüğü ve tek parça halinde üçgenlemeyi bitirmektir (Ek 8).

Bu kriterler ile gerçekleştirilen uygulamada her nokta bulutu için farklı değerler girilerek en iyi sonuç aranmıştır. Optech_1200 ve Optech_2600 nokta bulutu verileri için uygulanan "*alpha shapes*" işlemi ile MATLAB ortamında sonuçlar elde edilmiştir (Şekil 56).



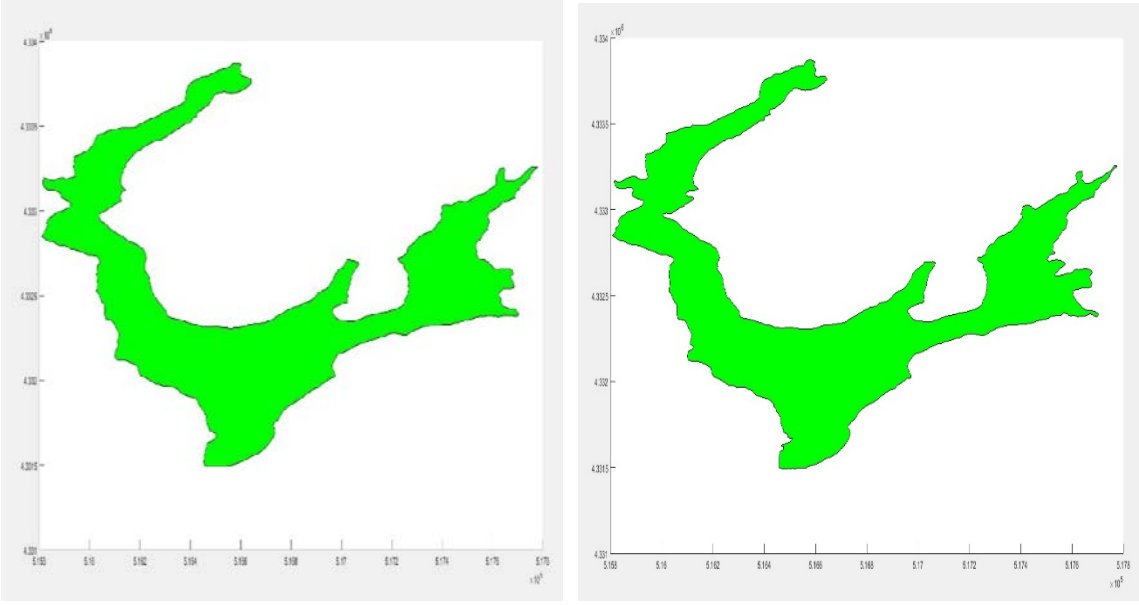
Şekil 56. Optech_1200 ve Optech_2600 verilerine ait "*alpha shapes*" üçgenleme görüntüsü

Riegl_1200.las ve Riegl_2600.las nokta bulutlatlarına ait MATLAB ortamında gerçekleştirilen "*alpha shapes*" uygulaması sonucu veriler elde edilmiştir (Şekil 57).



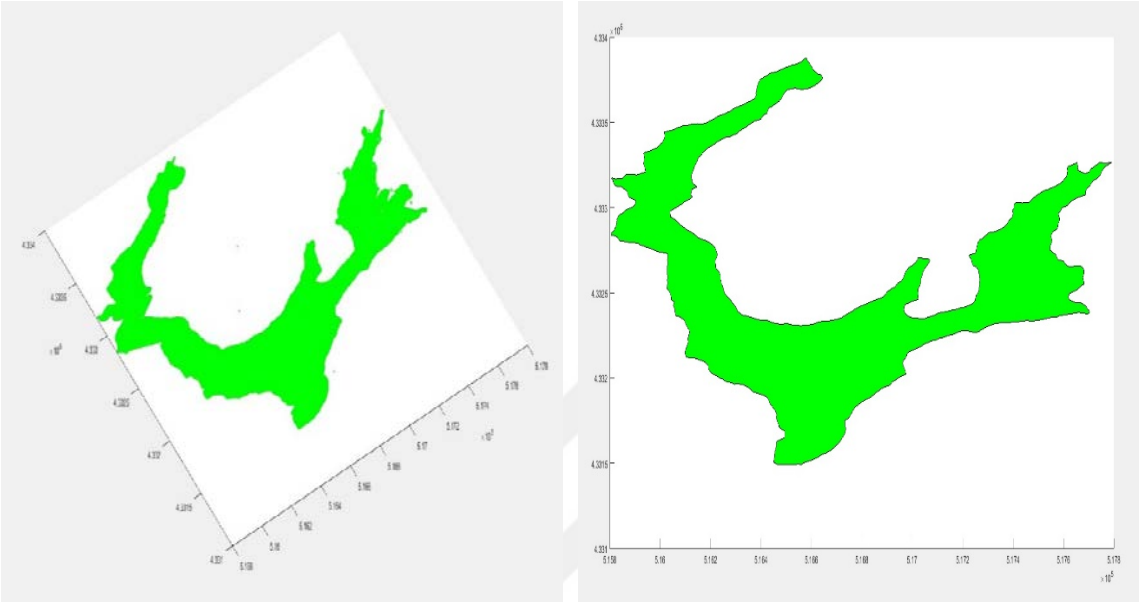
Şekil 57. Riegl_1200 ve Riegl_2600 verilerine ait "*alpha shapes*" üçgenleme görüntüsü

Optech_1200.las ve Optech_200.las verilerine uygulanan "alpha shapes" işlemi ile elde edilen üçgenleme sonucundaki oluşturulan alanın gösterimi sağlanmıştır (Şekil 58).



Şekil 58. Optech_1200.las ve Optech_2600.las verilerine ait "alpha shapes" alan görüntüsü

Riegl_1200.las ve Riegl_2600.las verilerine ait MATLAB ortamında gerçekleştirilen "*alpha shapes*" komutuyla elde edilen üçgenleme sonrası rezervuar alanı tespit edilmiştir (Şekil 59).

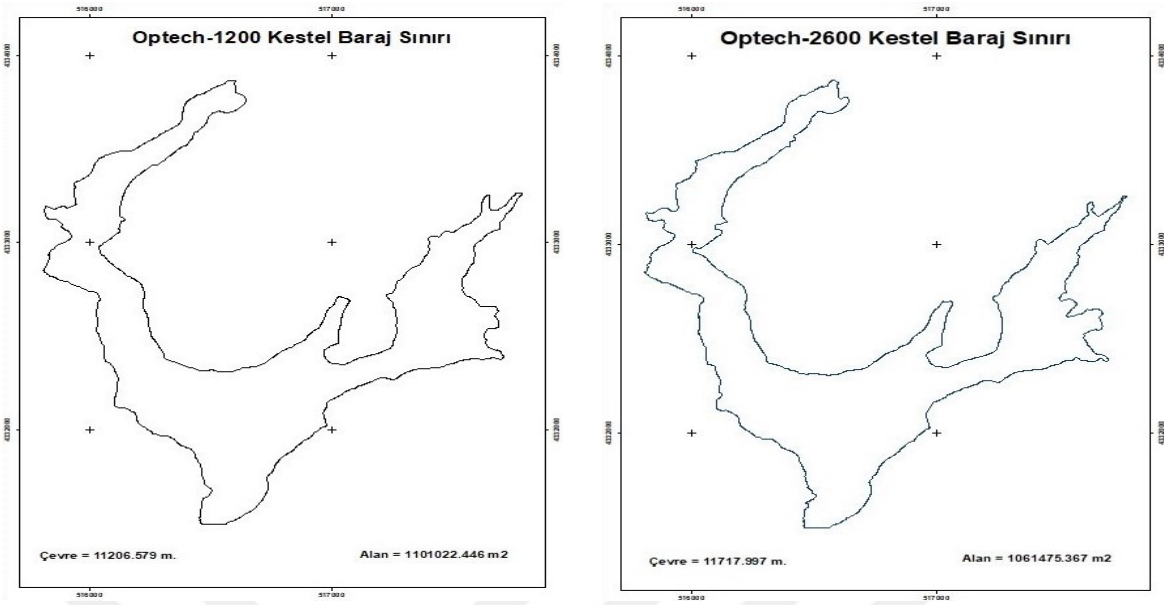


Şekil 59. Riegl_1200 ve Riegl_2600 verilerine ait "alpha shapes" alan görüntüsü

2.11. Sonuç Verilerine Ait Bilgiler ve Verilerin "*.shp" Formatına Dönüştürülmesi

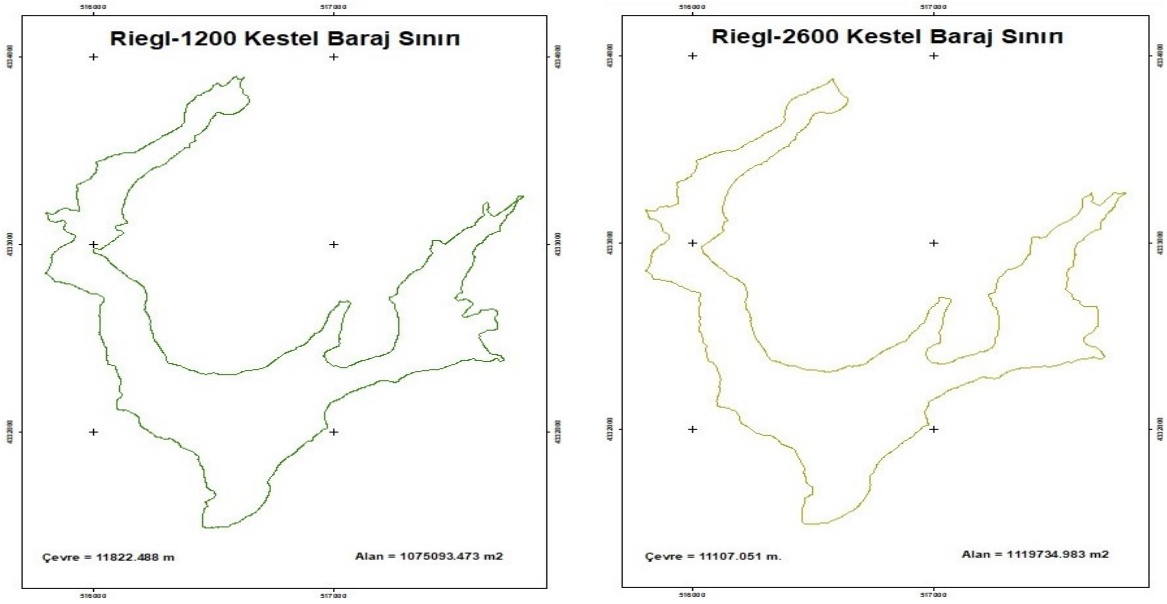
Sonuç olarak elde ettiğimiz sınıra dair çevre ve alan bilgilerini tespit edip, bu sınır verisini *"shp"* uzantılı şekilde export etme imkânı MATLAB ortamında gerçekleştirilmiştir (Ek 8).

ArcGIS ortamında "*shp*" uzantılı elde dilen Optech_1200 ve Optech_2600 sınırlarına dair harita bilgileri işlenerek bulunan çevre ve alan bilgileri yazılmıştır (Şekil 60).



Şekil 60. Optech-1200 ve Optech-2600 Kestel Baraj sınır tespiti

Kestel Barajı rezervuar alanına ait dört farklı lidar nokta bulutu ile yapılan sınır çıakrımı işlemi "*shp*" uzantılı verileri ArcGIS ortamında açarak harita bilgileri işlenmiştir. Riegl_1200 ve Riegl_2600 verileride aynı ortamda işlenerek çevre ve alan bilgileri yazılmıştır (Şekil 61).

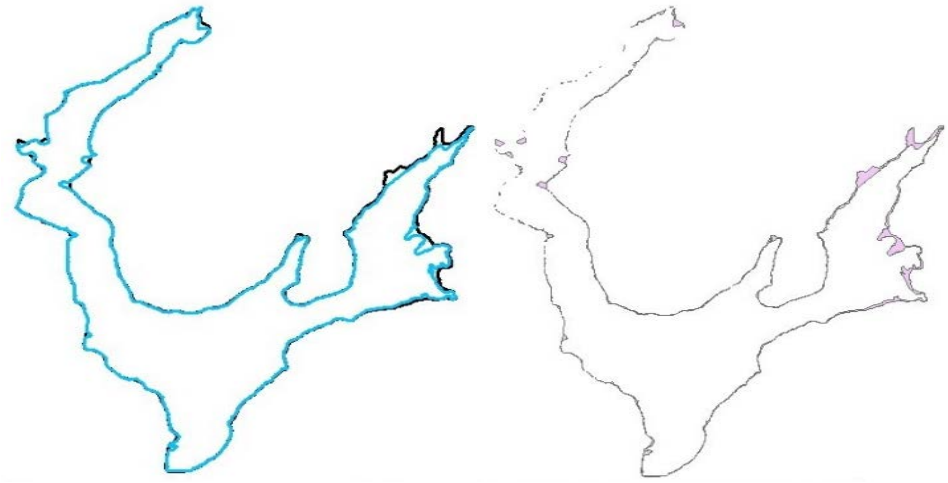


Şekil 61. Riegl-1200 ve Riegl-2600 Kestel Baraj sınır tespiti

2.12. Bulunan Sonuçların Referans Baraj Sınırıyla Çakıştırılması ve Değerlendirilmesi

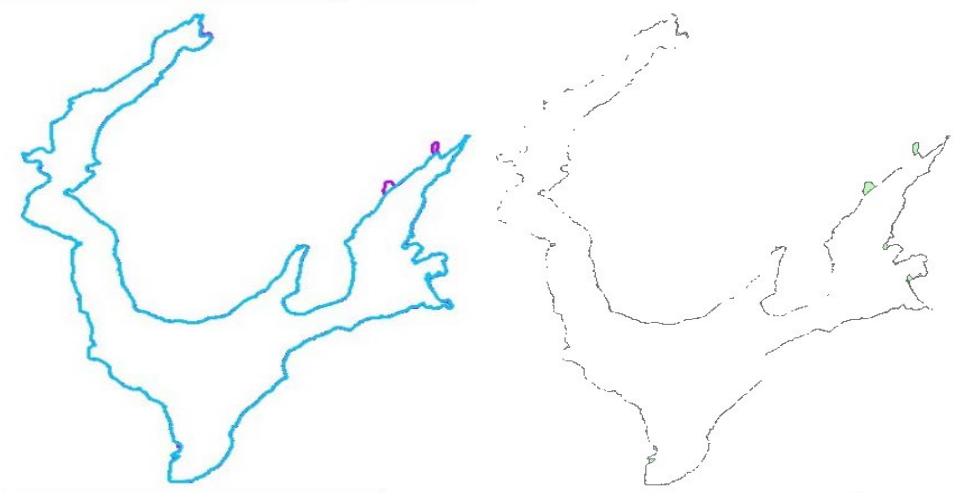
2013 yılına ait hava fotoğraflarından elde edilip 2016 yılında onaylanan İzmir ili Bergama ilçesine ait hâlihazır verisinden sağlanan Kestel Barajına ait sınır bilgisi, bulunan sonuçlarla çakıştırılarak elde dilen sonuçlar tablo halinde gösterilmiştir (Tablo 9). Referans olarak kullanacak olduğumuz kestel baraj sınırına ait detaylı bilgiler ise, çevre uzunluğu 12228.045 m olup alanı ise 1053263.960 m^2'dir.

İlk olarak tespit edilen Optech_1200 sınırıyla referans baraj sınır çakıştırılarak aralaraındaki benzerlikler ve farklılıklar görülmüştür (Şekil 62). Referans baraj sınırı oalrak kullanmış olduğumuz veri Kestel_Hâlihazır ismiyle turkuaz rengindedir. Tespit edilen sınır ise Kestel_Optech_1200 ismiyle siyah renktedir. Alansal olarak benzerliklere ve farklılıklara bakıldığı zaman %96 oranında bir benzerlik tespit edilmiştir (Tablo 9).



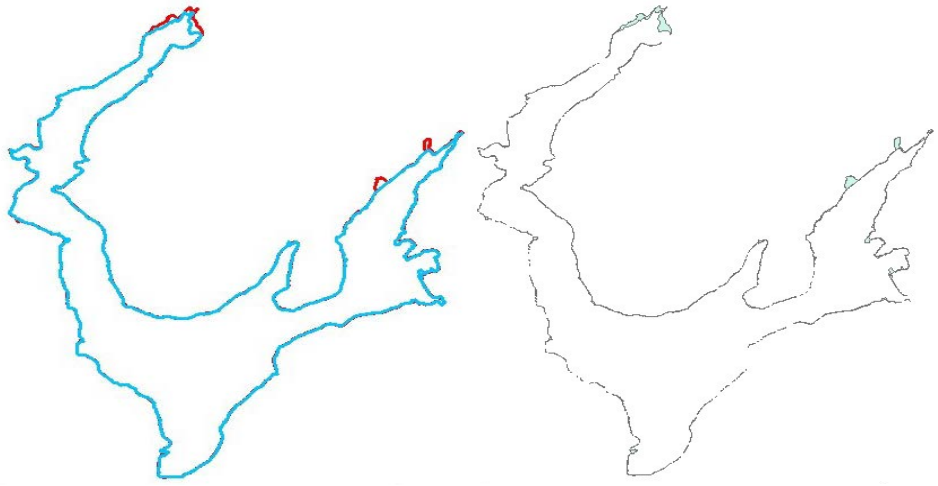
Şekil 62. Optech-1200 Kestel Baraj sınırı ile Hâlihazır baraj sınırının çakıştırılması ve farkları

Optech_2600 verisine ait tespit edilen sınır ile referans olarak kullandığımız sınır arasındaki ilişkiyi görmek için ArcGIS ortamında çakıştırma işlemi gerçekleştirilmiştir (Şekil 63). Kestel_Hâlihazır ismi ile kullanılan referans sınırı turkuaz renginde olup, Kestel_Optech_2600 ismiyle tespit edilen sınır pembe renktedir. İki sınır arasındaki alansal ilişkiye bakıldığında %99 oranında bir benzerlik göze çarpmaktadır (Tablo 9).



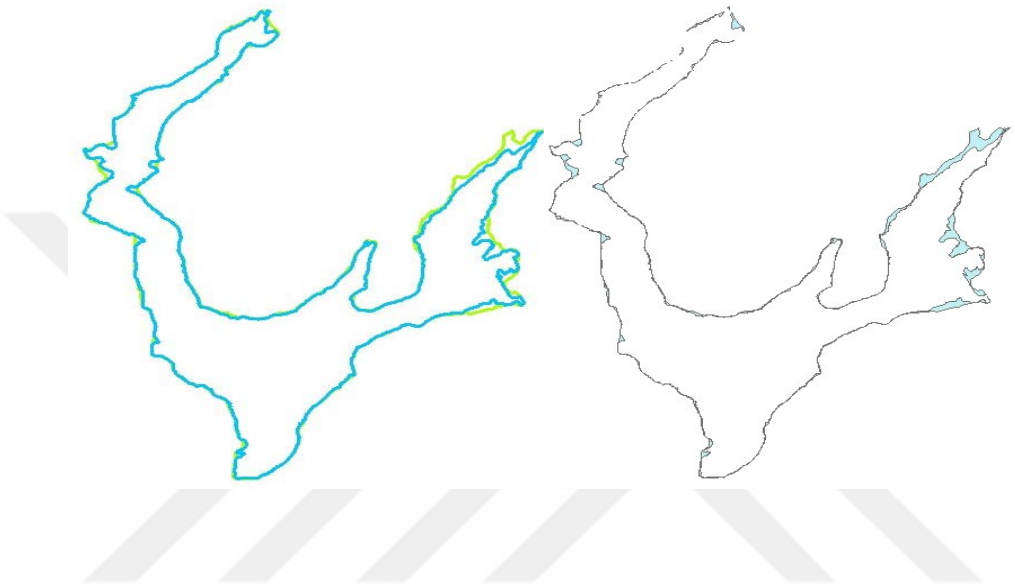
Şekil 63. Optech-2600 Kestel Baraj sınırı ile Hâlihazır baraj sınırının çakıştırılması ve farkları

Riegl_1200 verisine ait tespit edilen sınır ile hava fotoğraflarından elde edilen hâlihazır verisinden temin edilen referans sınırı ArcGIS ortamında çakıştırılmıştır (Şekil 64). Kestel_Hâlihazır ismiyle kullanılan referans sınırı turkuaz renginde olup, tespit edilen sınır Kestel_Riegl_1200 isminde ve kırmızı renktedir. Bu işlem sonucunda alansal olarak benzerlik ve farklılığa bakıldığında %98 oranında bir benzerlik tespit edilmiştir (Tablo 9).



Şekil 64. Riegl-1200 Kestel Baraj sınırı ile Hâlihazır baraj sınırının çakıştırılması ve farkları

ArcGIS ortamında, tespit edilen Riegl_2600 verisine ait sınır ile Kestel Barajına ait referans sınırı çakıştırma işlemi gerçekleştirildi (Şekil 65). Kestel_hâlihazır ismiyle kullanılan referans sınırı turkuaz renginde olup, Kestel_Riegl_2600 ismiyle kullanılan tespit edilen sınır yeşil renktedir. İki sınıra ait çakıştırma işlemi gerçekleştirildiktan sonra aralarındaki alansal ilişkiye dikkat edildiğinde %95 oranında bir benzerlik tespit edilmiştir (Tablo 9).



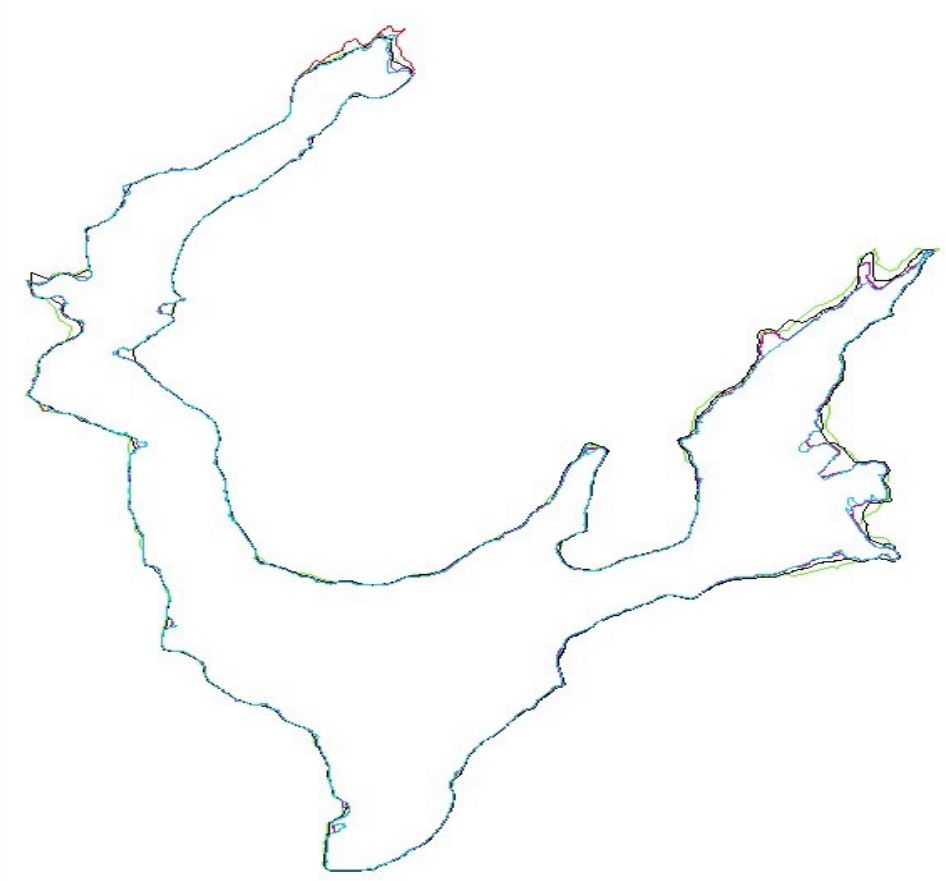
Şekil 65. Riegl-2600 Kestel Baraj sınırı ile Hâlihazır baraj sınırının çakıştırılması ve farkları

Tespit edilen sınırlar ile referans sınırının ArcGIS ortamında karşılaştırılması sonucu ortaya çıkan alansal benzerlikler ve farklılıklar tablo haline getirilmiştir (Tablo 9). Benzerlik oranlarının %95-%99 arasında değişiklik gösterdiği tespit edilmiştir.

Tablo 9. Tespit edilen sınırlar ile referans sınırının alansal karşılaştırılması

	Alan (m^2)	**Örtüşen Alan (m^2)**	**Örtüşmeyen Alan (m^2)**	**Benzerlik Oranı**
Kestel_Hâlihazır	1053263.960			
Optech_1200	1101022.447	1051605.805	49416.642	%96
Optech_2600	1061475.366	1050289.596	11185.770	%99
Riegl_1200	1075093.475	1052355.035	22738.440	%98
Riegl_2600	1119630.010	1052652.230	66977.780	%95

Referans sınırı olarak kullanılan Kestel_Hâlihazır verisi ile elde edilen sınırlar (Kestel_Optech_1200, Kestel_Optech_2600, Kestel_Riegl_1200 ve Kestel_Riegl_2600) ArcGIS ortamında çakıştırılarak aralarındaki benzerlikler ve farklılıklar görülmüştür (Şekil 66).



Şekil 66. Optech-1200, 2600 ve Riegl-1200,2600 Kestel Baraj sınırları ve hâlihazır baraj sınırının çakıştırılmış halde gösterimi

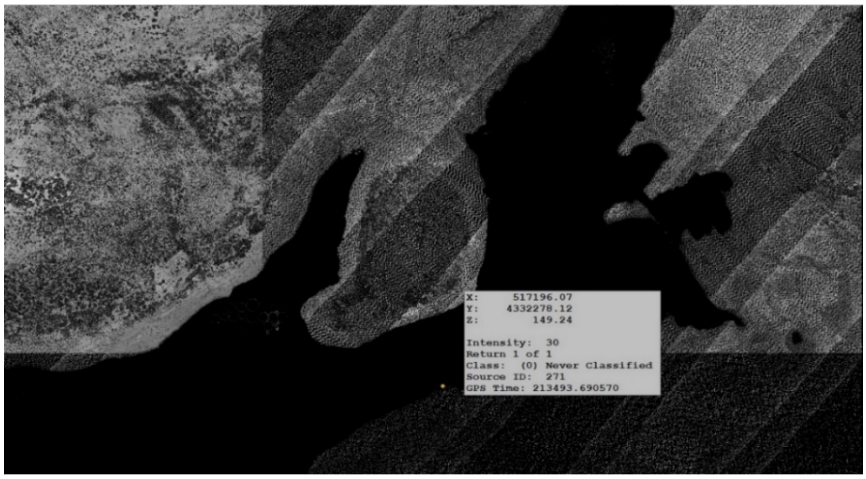
3. BULGULAR VE TARTIŞMA

Bu tez çalışmasında, LiDAR sistemi ile elde edilen nokta bulutu verisi içerisinden suyla çevrili bir yerin karayla kesiştiği yeri yani kıyı çizgisini (rezervuar alanı) otomatik bir biçimde nasıl çıkarılabileceği ele alınmıştır. Bu çalışma nokta bazlı bir çalışma olduğu için yapılan işlemlerdeki öncelik en sağlıklı nokta bulutuna erişip ilgili noktalar ile rezervuar alanını elde etmek amaçlanmıştır. Bu amaç doğrultusunda MATLAB programlama dili kullanılarak sonuçlar elde edilmiştir.

Kestel Barajı rezervuar alanına dair çalışma Harita Genel Komutanlığı (HGK) tarafından Bilimsel Araştırma ve Koordinasyon Komisyonu (BARKOK) tarafından gerçekleştirilen LiDAR uçuşları sayesinde temin edilmiş ve ölçüm sırasında 51 adet kontrol noktasının 26 adedi kontrol noktası, 25 adedi ise denetleme noktası olarak kullanılmıştır. Optech marka LiDAR sistemi ile elde edilen denetleme noktalaraındaki karesel ortalama hata ± 0.07 m, Riegl marka LiDAR sistemi ile elde edilen denetleme noktalarındaki karesel ortalama hata ± 0.03 m olarak hesaplanmıştır. Temin edilen bu noktalara güvenerek yürütülen bu çalışmada harici bir ölçme işlemi gerçekleştirilememiş ve elde edilen nokta koordinatı ölçme doğruluğu kabul edilmiştir.

Bu tez kapsamında yapılan bazı uygulamalar için çok daha doğru sonuçlar alabilmek adına bölgede ölçüm yapılması çok daha sağlıklı olacaktır. Örneğin, noktalara dair tespit edilen sinyal yoğunluğu (intensity) değerini karayla suyu ayırt etmek açısından emin olunan noktalar üzerinden ölçüm yapılıp yorumlamak çok daha sağlıklı olacaktır fakat imkânlar doğrultusunda bu gerçekleşememiştir. Yapılan benzer çalışmalardan elde edilen değerler dikkate alınarak bu değere karar verilmiştir.

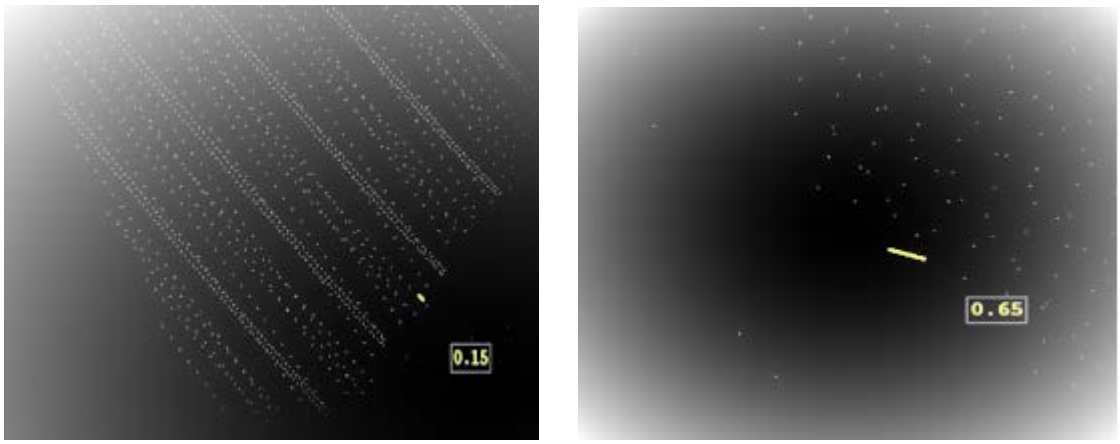
Suyun belli bir mesafe altından veya ıslak bir araziden yansıyan parlaklık değerlerini arazide görerek emin olunan noktalar üzerinden değerlendirmek uygulamada kullanılacak eşik değerini belirleme açısından çok daha uygundur (Şekil 67). Böylelikle kara ve su ayırımı için ölçüm sonucu tespit edilen bir eşik değeri kullanılarak daha hassas sonuçlar elde edilebilir.



Şekil 67. LiDAR nokta bulutunda bulunan bir noktaya ait sinyal yoğunluğu (intensity) değeri

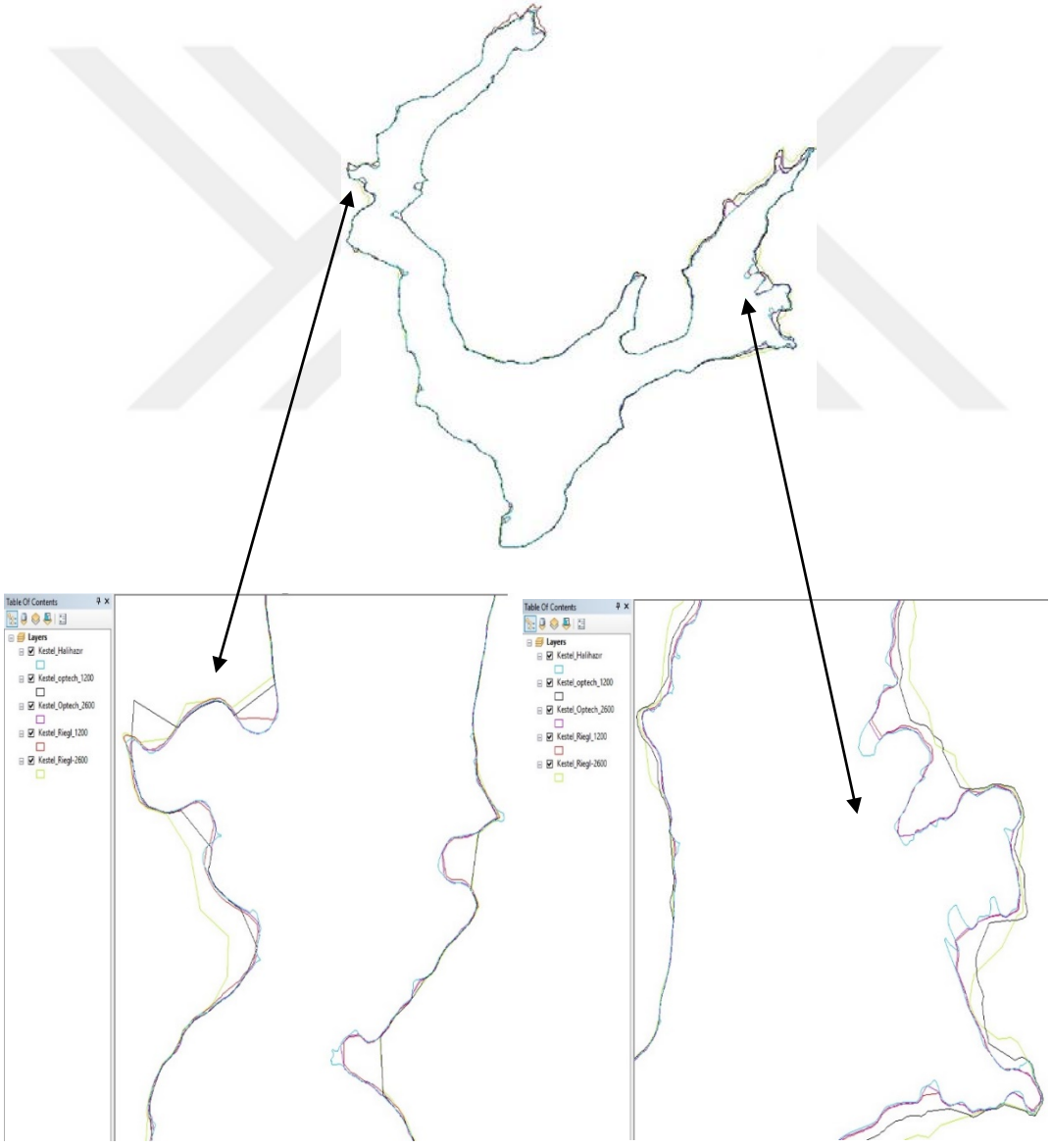
İki farklı firma ile iki farklı yükseklikten elde edilen değerlerde dikkat edilmesi gereken bir diğer husus ise, yükseklik farkından kaynaklı metre kareye düşen nokta sayısına dikkat edilmelidir. 1200 m yüksekten yapılan uçuşlarda metre kareye 8 nokta düşerken, 2600 m yükseklikten gerçekleştirilen ölçümlerde ise metre kareye 2 nokta düşmektedir. Bu bilgi noktalar arasında komşuluk ilişkisine yani noktalar arasındaki mesafe bilgisine dayanan uygulamalar için önemlidir.

LiDAR nokta bulutundan gürültüyü ve çakışan noktaları gidermek için uygulanan "*pcdenoise*" ve "*downsample*" fonksiyonlarında kritik eşik değerini belirlerken göz ardı edilmemesi gereken husustur. Belirlenecek eşik değeri noktaların birbirine olan yakınlığını ilgilendirdiği için farklı yüksekliklerden elde edilen nokta bulutları için bu değer farklılık gösterecektir (Şekil 68).



Şekil 68. Yükseklikleri 1200 m ve 2600 m olan uçuşlardaki nokta yakınlıkları

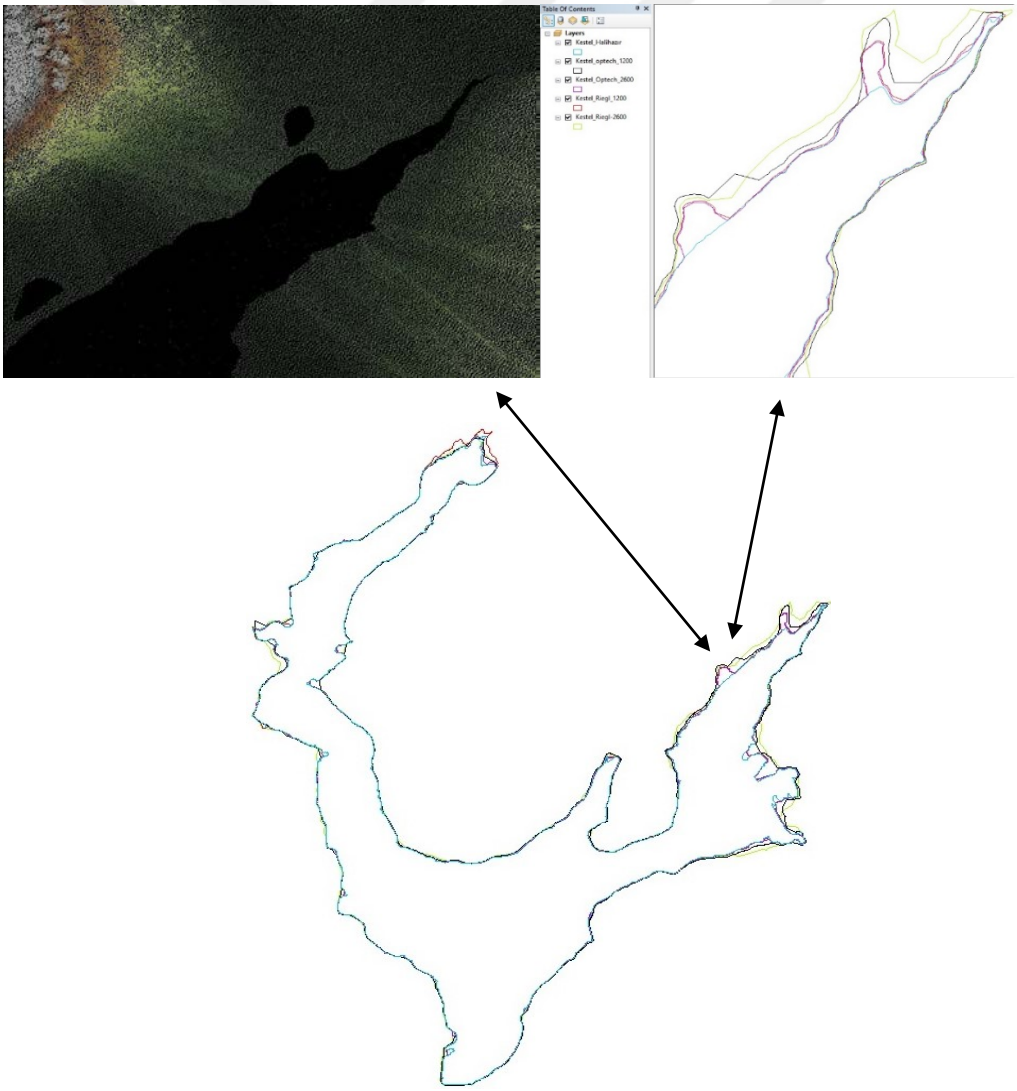
LiDAR nokta bulutu verileri için yapılan ön işlem uygulamalarının ardından, Kestel Barajı rezervuar alanının tespiti için gerekli noktalar sağlıklı bir şekilde bulunmuştur. Sınır çizimi için uygulanan "*alpha shapes*" komutunda dikkat edilmesi gereken husus yarıçap değerinin doğru tespit edilmesidir. Belirlenen yarıçap değerine uygun üçgenleme işlemi bu doğrultuda yapılmaktadır. Hangi noktaların bu üçgenleri oluşturacağı bu değer sayesinde belirlenmektedir. Değer küçük tutuldukça şeklin konkav (iç bükey) olması sağlanırken değer büyük tutuldukça konveks (dış bükey) olması sağlanmaktadır. Söz konusu Kestel Barajı gibi sınır belirleme işlemleri için girinti çıkıntının fazla olduğu yerlerde bu değeri iyi belirlemek sağlıklı sınıra ulaşmak için büyük önem arz etmektedir (Şekil 69).



Şekil 69. "alpha shapes" uygulamasındaki radius değerinin şekil üzerindeki konkav veya konveks oluşumdaki etkisinin görünümü

Sonuçlara bakıldığında dikkat çeken ve sınır değişimini en fazla etkileyen bir diğer husus ise baraj alanı içerisinde oluşan küçük su göletleridir. LiDAR nokta bulutlarını ön işleme tabi tuttuktan sonra baraj sınırını bulmamıza yardımcı olacak nokta kümesini elde ederken en az kota sahip olan nokta ve o noktanın 20 ila 30 cm aralığındaki noktaları tespit edilmiştir. Bu işlemi yaparken aynı bölge içinde aynı kota sahip farklı daha küçük göletler var ise bunları da işleme dahil ettiği için baraj sınırında bir değişim kaçınılmazdır (Şekil 70).

Bu tür yerleri ya en başta "*ginput*" uygulaması dahilinde seçilen alanın dışında bırakılması gerekecek yada "*alpha shapes*" uygulamasında üçgenleme yapılacak noktaların dışında tutulması sağlanmalıdır. Bu husus, üzerinde çalışılması gereken ve iyileştirilmesi gereken bir problem olarak çıkmıştır. İlerleyen çalışmalarda bu problemin çözümü için uğraş verilecektir.



Şekil 70. Baraj sınırını belirlerken sınır değişimine etki eden küçük göletler

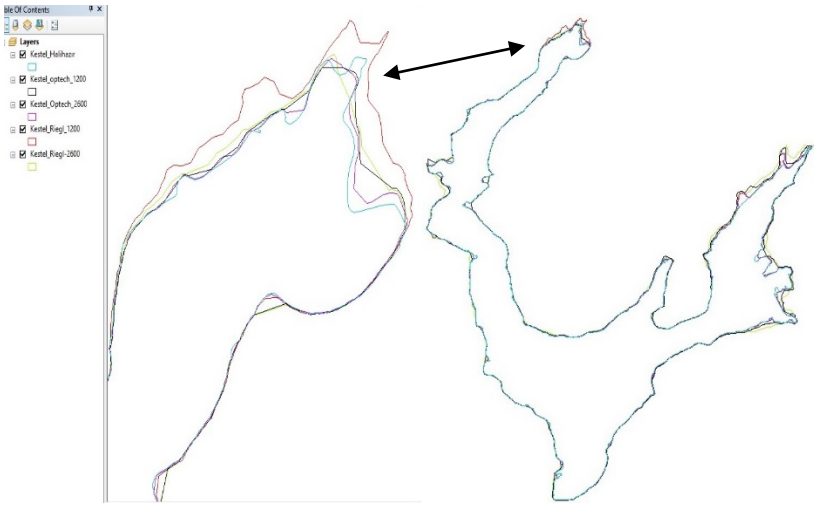
Kestel barajı gibi sulama barajlarındaki su seviyelerinin çok fazla değişkenlik gösterdiği bilinmektedir. Bu değişkenliği göz önünde bulundurarak elde edilen sonuçları değerlendirmek ve referans sınırı olarak kabul edilen değerin yapılan LiDAR ölçümlerine yakın bir tarihte elde edildiğinden emin olmak gerekmektedir. Söz konusu baraj bir sulama barajı olduğundan dolayı su seviyesindeki değişimin fazla olması kıyas yapılacak referans sınırını da iyi belirlemeyi gerektirecektir. Bu hususta değişimin ne kadar farklılık gösterdiğini Google Earth aracılığı ile farklı tarihlerde çekilen görüntüler ortaya koymaktadır (Şekil 71, 72).

Şekil 71. 12 Haziran 2010 tarihinde Goole Earth aracılığıyla gösterilen Kestel Barajı

Şekil 72. 19 Kasım 2016 tarihinde Goole Earth aracılığıyla gösterilen Kestel Barajı

Kestel Barajı rezervuar alanına ait elde edilen sonuçları kıyaslamak için referans olarak seçilen veri 2013 yılında çekilmiş hava fotoğraflarından elde edilen Hâlihazır Haritadan elde edilmiştir. Söz konusu yer su değişiminin çok fazla yaşandığı bir sulama barajı olması sebebiyle sağlıklı bir kıyas ancak LiDAR uçuşunun gerçekleştiği gün elde edilmiş olan baraj sınırı ile yapılabilir. LiDAR uçuşunun yapıldığı tarihlerdeki baraj suyu seviyelerine bakıldığı zaman, 20-21 Ekim 2014 tarihleri arasında Optech Pegasus HA500 lidar sistemi ile yapılan test uçuşunda barajdaki su seviyesi 93.70 m. olup, 03-05 Kasım 2014 tarihinde ise Riegl LMS-Q 1560 lidar sistemi ile yapılan test uçuşunda barajdaki su seviyesi 110.47 m'dir. İki uçuş arasındaki gün farkı çok fazla olmamasına rağmen (15 gün) baraj suyu seviyesindeki değişim çok bariz kendini belli etmektedir (16.77m). Bu yüzden elde edilen sonuçların daha sağlıklı değerlendirilmesi için değişimin daha az yaşandığı dönemlerden elde edilen verilerle kıyas yapmak daha sağlıklı olacaktır.

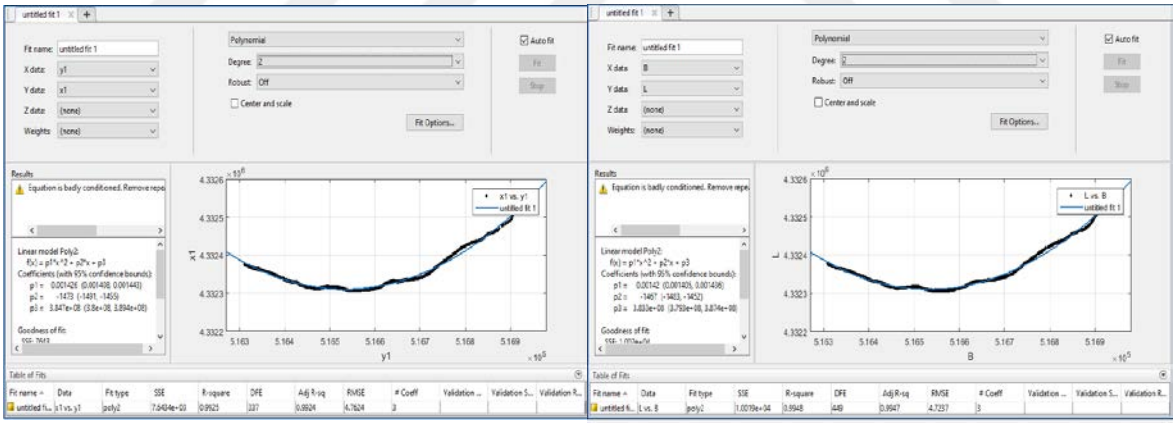
Sonuç olarak elde ettiğimiz Kestel baraj sınırlarına dikkatli bakıldığında değişimin en fazla görüldüğü yerler baraj suyunun genişleme sağladığı uç noktalarda olduğu görülecektir (Şekil 73), bunun sebebi iki uçuş arasındaki baraj suyu seviyesindeki farklılıkdan dolayı olduğu göze çarpmaktadır. Bir diğer farklılığın görüldüğü yerler ise baraj sınırını oluşturan girinti ve çıkıntıların fazla olduğu bölgeler olmaktadır. Bunun sebebi ise LiDAR nokta bulutu için yapılan uygulamalardaki seçilen eşik değerleri büyük rol oynamaktadır. Çok daha sağlıklı bilgiler eşliğinde seçieln eşik değerleri ile daha iyi sonuçlar alınacağı görülmektedir.



Şekil 73. İki uçuş süresi arasındaki baraj su seviyesindeki değişimin sınırlardaki farklılığa etkisinin görünümü

Optech_1200 verisi ile tespit edilen rezervuar alan sınırı Google Earth programı yardımıyla görüntülendikten sonra belirli bir hat boyunca (Şekil 74-a) TCX Converter programı ile konum bilgisi tespit edildi. Tespit edilen sınıra ait noktalar (Optech_1200) ve Google Earth üzerinden aynı sınır hattı üzerinden elde edilen noktalar ile MATLAB ortamında polinom eğrileri elde edildi (Şekil 74-b ve Şekil 74-c). Polinom eğrilerine ait kat sayı karşılaştırmaları (6) numaralı forlmülde ve elde edilen noktalara ait güven aralığı (r-square) tespit edilerek benzerlikleri ve farklılıkları hakkında yorum yapma imkânı oluşmuştur (Tablo 10).

(a)



(b) (c)

Şekil 74. a) Karşılaştırma için belirlenen hat b) Google Earth üzerinden alınan noktalara ait polinom eğrisi c) Tespit edilen Optech_1200 verisine ait polinom eğrisi

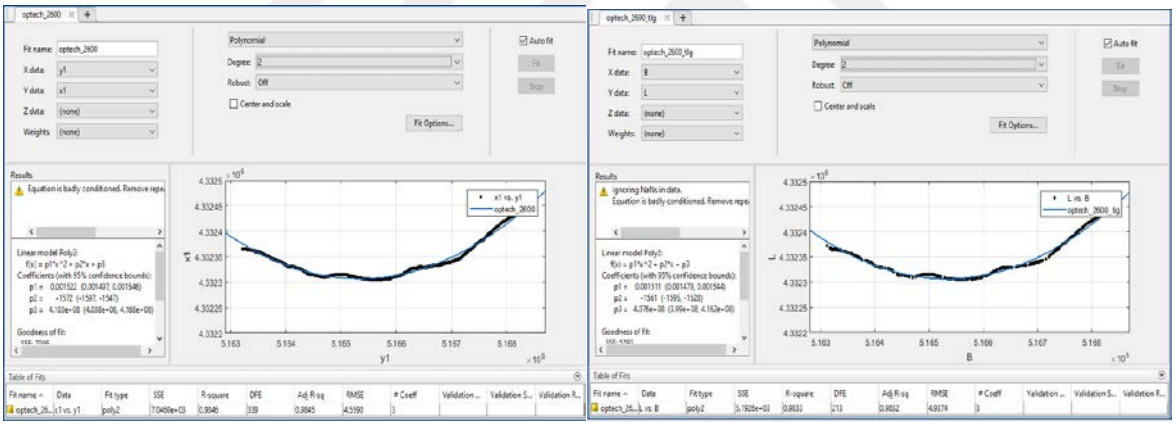
$$f(x)=p1x^2 + p2x + p3 \quad (6)$$

p1 = 0.00143 p2 = -1473 p3 = $3.487x10^8$ p1 = 0.00142 p2 = -1467 p3 = $3.833x10^8$

R-square =0.99 (Google Earth) R-square = 0.99 (Optech_1200)

Optech_2600 verisine ait elde edilen rezervuar alan sınırı içinde Optech_1200 verisi için yapılan işlemler tekrarlanmıştır (Şekil 75). Polinom eğrilerine ait kat sayı karşılaştırmaları (7) numaralı forlmülde ve elde edilen noktalara ait güven aralığı (r-square) tespit edilmiştir. Aynı zamanda farklı yükseklikten elde edilen veriler olduğu için benzer bir hat üzerinden işlemi gerçekleştirerek yükseklik farkının sonuçlar nasıl bir etkisinin olduğu yorumlanabilmiştir.

(a)



(b) (c)

Şekil 75. a) Karşılaştırma için belirlenen hat b) Google Earth üzerinden alınan noktalara ait polinom eğrisi c) Tespit edilen Optech_2600 verisine ait polinom eğrisi

$$f(x)=p1x^2 + p2x + p3 \quad (7)$$

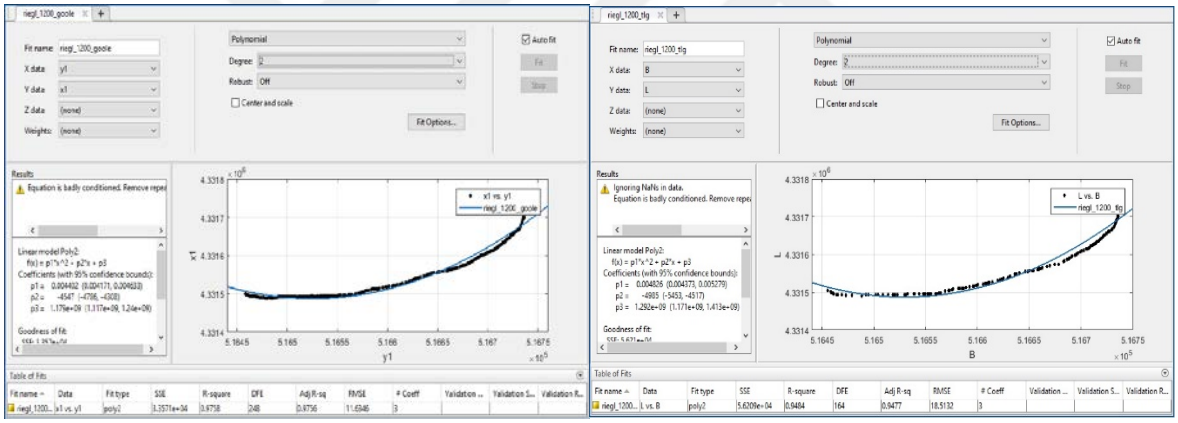
p1 = 0.00152 p2 = -1572 p3 = $4.103x10^8$ p1 = 0.00151 p2 = -1561 p3 = $4.076x10^8$

R-square =0.98 R-square = 0.98

(Google Earth) (Optech_2600)

Riegl_1200 verisine ait elde edilen rezervuar alan sınırı için de diğer verilere yapılan işlemler tekrarlanmıştır (Şekil 76). Polinom eğrilerine ait kat sayı karşılaştırmaları (8) numaralı forlmülde ve elde edilen noktalara ait güven aralığı (r-square) tespit edilmiştir. Optech_1200 ve Optech_2600 verisi için belirlenen hattan farklı bir yer seçilerek elde edilen sonuçların doğruluğunun homojenliği tespit edilmiştir.

(a)



(b) (c)

Şekil 76. a) Karşılaştırma için belirlenen hat b) Google Earth üzerinden alınan noktalara ait polinom eğrisi c) Tespit edilen Riegl_1200 verisine ait polinom eğrisi

$$f(x)=p1x^2 + p2x + p3 \quad (8)$$

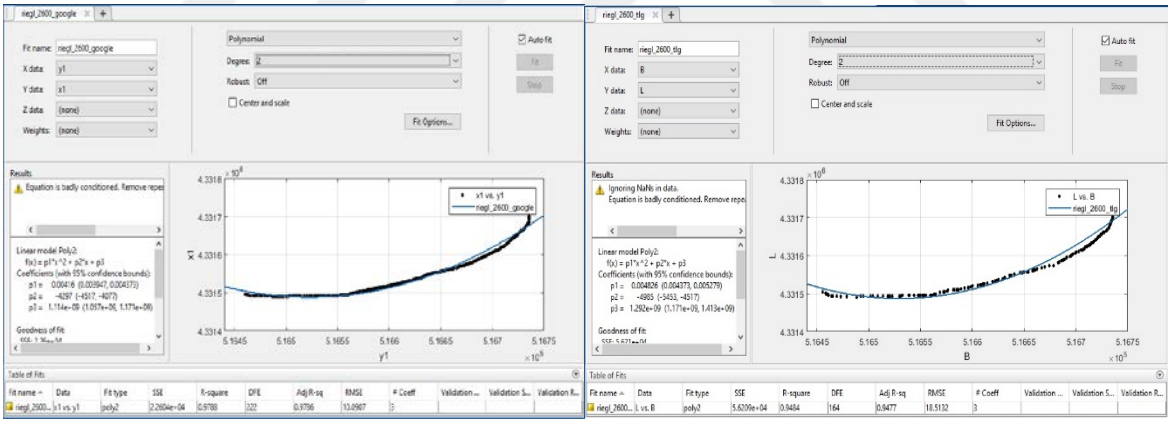
p1 = 0.00440 p2 = -4547 p3 = $1.179x10^9$ p1 = 0.00483 p2 = -4985 p3 = $1.292x10^9$

R-square =0.98 R-square = 0.95

(Google Earth) (Riegl_1200)

Riegl_2600 verisi kullanılarak elde edilen rezervuar alan sınırının doğruluğu hakkında yorum yapılabilmesi için yukarıda uygulanan işlemler tekrarlanarak polinom eğrileri elde edilmiştir (Şekil 77). Polinom eğrilerine ait kat sayı karşılaştırmaları (9) numaralı forlmülde ve elde edilen noktalara ait güven aralığı (r-square) tespit edilmiştir. Burada kullanılan veriler Riegl_1200 verisindeki gibi benzer bir hat üzerinden alınarak yükseklik farkına dair yorumlar yapılabilmiştir.

(a)



(b) (c)

Şekil 77. a) Karşılaştırma için belirlenen hat b) Google Earth üzerinden alınan noktalara ait polinom eğrisi c) Tespit edilen Riegl_2600 verisine ait polinom eğrisi

$$f(x)=p1x^2 + p2x + p3 \quad (9)$$

p1 = 0.00416 p2 = -4297 p3 = $1.114x10^9$ p1 = 0.00483 p2 = -4985 p3 = $1.292x10^9$

R-square =0.98 (Google Earth) R-square = 0.95 (Riegl_260)

Tespit edilen rezervuar alan sınırlarına dair doğrulukları hakkında Kestel bölgesindeki hâlihazır çalışmasından elde edilen referans kıyı çizgisi ile alansal benzerlikler ve farklılıklar tespit edilmiştir. Hâlihazır haritanın üretim tarihi LiDAR uçuşunun yapıldığı tarihte olmadığı için kıyas yaparken göz ardı edilmemesi gereken hususlara değinilmiştir. Söz konusu kıyasın Google Earth üzerinden nokta alımı gerçekleştirerek aynı hat üzerinden elde edilen noktaların oluşturmuş olduğu polinom eğrisinin katsayılarına bakarak yapılmıştır. Elde edilen polinom eğrilerine ait katsayılara bakıldığında benzerlik gösterdiğini, yükseklik farkının elde edilen sınırlara dair direkt olarak etki etmediği görülmektedir. Yükseklik farkının seçilen eşik değerlerini etkilediğinden bahsedilmiştir, uygun eşik değerleri ile farklı yükseklikten elde edilen verilerin yakın bir sınır değeri elde edildiği görülmektedir. Farklılığın ise, iki farklı markaya ait uçuşlar iki farklı tarihte gerçekleştiği için uçuşun yapıldığı tarihlerdeki su seviyelerin eşit olmadığından anlaşılmaktadır.

Tablo 10. Elde edilen sınır verileri ve Google Earth verilerinin polinom eğri katsayı karşılaştırması

Veriler	Polinom Katsayıları			R-square
	p1	**p2**	**p3**	
Google Earth	0.00143	-1473	$3.847x10^8$	0.99
Optech_1200	0.00142	-1467	$3.833x10^8$	0.99
Google Earth	0.00152	-1572	$4.103x10^8$	0.98
Optech_2600	0.00151	-1561	$4.076x10^8$	0.98
Google Earth	0.00440	-4547	$1.179x10^9$	0.98
Riegl_1200	0.00483	-4985	$1.292x10^9$	0.95
Google Earth	0.00416	-4297	$1.114x10^9$	0.98
Riegl_2600	0.00483	-4985	$1.292x10^9$	0.95

4. SONUÇLAR VE ÖNERİLER

Haritacılık uygulamaların çoğunda yer almaya başlayan LiDAR sisteminin baraj, göl gibi yerlerin sınırlarını tespit edebilmek için ne kadar uygun ve doğru olabileceği Kestel Barajı ölçeğinde gösterilmeye çalışılmıştır. Sadece LiDAR verisi değil aynı zamanda bu verinin işlendiği MATLAB programın da yazılan kodların da ihtiyacı ne kadar karşıladığı görülmüştür.

Yapılan çalışmada kullanılan veri olarak sadece LiDAR sistemi ile elde edilen nokta bulutları kullanılmıştır. Bu nokta bulutları haricinde herhangi bir veri (hava fotoğrafı, sayısal yükseklik modeli, vb.) kullanılmayıp salt bir nokta verisi üzerinden MATLAB progamlama dili aracılığıyla otomatik bir şekilde baraj rezervuar alanının çıkarımı yapılmıştır.

LiDAR verisi elde edilirken kullanılan sinyalinde etkisi büyük ölçüdedir. Bu çalışmada su yüzeyinden ve su altı topoğrafyasından herhangi bir veri alınamadığından model ve yöntemler ona göre belirlenmiştir. Sonuç olarak dört farklı veri setinden elde edilen rezervuar alanına ait sınırlar hem kendi aralarında hem de referans bir sınır verisi ile karşılaştırılmıştır. Aynı zamanda iki farklı yükseklikten elde edilen verilerlede kıyas yapma şansı olmuştur. Yükseklik farkının sınır değişimine direkt olarak bir etkisinin olmadığı sadece nokta yoğunluğunu değiştirdiği görülmüştür. Nokta yoğunluğuna göre uygulanan matematik modellerdeki eşik değerleri değiştirilerek birbirine yakın sonuçlar elde edilmiştir.

Uygulanan birçok matematik modelde dikkat edilmesi gereken, noktaların birbirine olan yakınlığı (komşuluğu) olduğu için veri setine ait nokta yoğunluğuda her model için kullanılacak eşik değerini ona göre değiştirmektedir. Sinyal yoğunluğu (intensity) filtrelemesi, nokta bulutundaki gürültüyü gidermek için kullanılan "*pcdenoise*" ve mükerrir noktaları ayıklayan "*downsample*" fonksiyonları ve "*alphashapes*" uygulamaları gibi matematik modellerde kritik olan eşik değerin iyi belirlenmesidir.

Kestel sulama barajı gibi söz konusu yerlerde yapılacak olan sınır tespitleri sayesinde zamana bağlı olarak değişen su kütlesine dair hesaplar yapılabilir. Bu hesaplar neticesinde su tasarrufu ile ilgili bilgilere ulaşma imkânı oluşturulabilir. Böylece su değişiminin yaşandığı böyle yerlerden edinilen bilgiler devlet kamu kurumlarından Devlet Su İşlerinin pek çok konuda işine yarayacağı düşünülebilir.

Bu tez çalışmasına ilişkin güncel bir bilgi olarak, Harita Genel Komutanlığı tarafından Bergama test bölgesinde LiDAR uçuşlarının yapıldığı ve bölgede hiperspektral görüntülerin elde edildiği ve paylaşıma açıldığı bilgisi verilmiştir (URL 3). Böylelikle 2100 m. den elde edilen çok bantlı (hiperspektral) görüntüler sayesinde su yüzeyinden de veri temin ederek yapılan çalışma geliştirilebilecek ve daha iyi sonuçlar elde edilebilecektir.

Tespit edilen sonuçlar Google Earth aracılığı ile elde edilen görüntülerle çakıştırılarak yapılan çalışmaya görsellik katmıştır (Şekil 78, 79). Elde edilen rezervuar alanları ile mevcut görüntülerdeki suların çakışmama sebebi farklı zaman dilimlerinde elde edilmesinden kaynaklıdır.

Şekil 78. Sonuç olarak bulunan Optech-1200 ve Optech-2600 Kestel Baraj sınırının Google Earth görüntüsüyle çakıştırılmış hali

Şekil 79. Sonuç olarak bulunan Riegl-1200 ve Riegl-2600 Kestel Baraj sınırının Google Earth görüntüsüyle çakıştırılmış hali

Tespit edilen sonuçların doğruluğuna dair sağlıklı bir kıyas yapabilmek için LiDAR uçuşunun yapıldığı tarihlerde elde edilmiş hava fotoğrafının olması gereklidir. Ancak böyle bir çalışma yapılamadığından, 2013 yılında hava fotoğraflarından elde edilen hâlihazır haritadan Kestel Barajına ait referans sınırı elde edilmiş ve karşılaştrmalar bu veri üzerinden yapılmıştır. Elde edilen sonuçların doğruluğu hakkında yorum yapabilmek için mevcut imkânlar doğrultusunda Google Earth üzerinden konum bilgisi elde ederek tespit edilen sınırlara ait belli aralıklar karşılaştırılmıştır.

Google Earth ve TCX Converter programları yardımıyla elde edilen sınır hattına denk gelen yerlerden konum bilgisi (X,Y,Z) elde edilerek MATLAB ortamında 2. dereceden polinom eğrisi uydurulmuştur. Aynı sınır hattını temsil edecek olan ve tespit edilen noktalar ile de 2. dereceden polinom eğrileri elde edilerek polinom kat sayıları ile karşılaştırma yapma imkânı olmuştur

5. KAYNAKLAR

Canaz S., Karslı F., Guneroglu A. ve Dihkan M., 2015. Automatic boundary extraction of inland water bodies using LİDAR data, Ocean & Coastal Management, Trabzon, Turkey, vol.118, no.B, pp.158-166.

Canaz S., Karsli F., Guneroglu A. ve Dihkan M., 2015. Lidar Verileri Kullanılarak Göl Sınırlarının Otomatik Olarak Belirlenmesi, TUFUAB III. Teknik Kurultay, Konya, ss.1-1.

Ekercin, S. ve Üstün B., 2004. Uzaktan Algılamada Yeni bir Teknoloji: Lidar, HKMO Jeodezi, Jeoinformasyon ve Arazi Yönetimi Dergisi, İTU, İstanbul, 91, 34-38.

Fischler, M. A. and Bolles R. C., 1981. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." Communications of the ACM, 24,6,381-395.

Hartley, R. and Zisserman A., 2000. Multiple view geometry in computer vision, Cambridge Univ Press, 671 p., ISBN: 0521540518.

HGK, 2014. LiDAR Test Uçuş Programı, http://www.hgk.gov.tr. Erişim: 14 Aralık 2014

Höfle B., Vetter M., Pfeifer N., Mandlburger G. ve Stötter J., 2009. Water surface mapping from airborne laser scanning using signal intensity and elevetion data, Earth Surface Processes and Landforms, Vienna, Austria, 34, 1635–1649.

Jiang H., Member, IEEE, Zhang S., Tan Z. and Wang C., 2014. Connectivity-Based Boundary Extraction of Large-Scale 3D Sensor Networks: Algorithm and Applications, IEE Transactions on Parallel and Distributed Systems, China, vol. 25, no. 4.

Kayı A., Erdoğan M. ve Eker O., 2015. Optech HA_500 ve Riegl LMS-Q1560 ile Gerçekleştirilen LiDAR Test Sonuçları, Harita Dergisi, Ankara.

Kalkan, Y. ve Alkan R.M., 2005. Sularla Kaplı Alanlarımız ve Hidrografik Ölçmeler, TMMOB HKMO 10.Harita Bilimsel ve Teknik Kurultayı, Ankara.

Kalkan Y., 2005. Barajlarımızdaki Hidrografik Ölçmeler ve Sediment Hareketleri, TMMOB HKMO 12. Harita Bilimsel ve Teknik Kurultayı, Mayıs, Ankara.

Liadsky J., 2007. Recent advancements in commercial LiDAR mapping and imaging systems. Informally published manuscript, Optech Incorporate, Available from NPS LiDAR Workshop. Retrieved from http://www.nps.edu/academics/Centers/RemoteSensing/ Presentations/LiDAR/ Presentations/ RecentAdvancements.pdf.

Morsy S., Shaker A. and El-Rabbany A., 2018. Using Multispectral Airbone LiDAR Data for Land/Water Discrimination: A Case Study at Lake Ontario, Canada, 8, 3, 349.

Mostafa, M. M., ve Soussa, H. K.,2006.Monitoring of LakeNasser using remote sensing and GIS techniques. (ISPRSMid-term Symposium Proceeding. May 2006, Enschede), ISSN 0976 – 4380.

NOAA (National Oceanic and Atmospheric Administration) Coastal Services Center, (2012). "LiDAR 101: An Introduction to LiDAR Technology, Data, and Applications." Revised. Charleston, SC: NOAA Coastal Services Center.

Optech, 2014-a. LiDAR Test Report of TURKEY

Petzold, B., B., Reiss, P., and Stossel, W., 1999. Laser scanning-surveying and mapping agencies are using a new technique for the derivation of digital terrain models. ISPRS Journal of Photogrammetry & Remote Sensing, 54, 95–104.

Polat N. ve Uysal M. 2016. Hava Lazer Tarama Sistemi, Uygulama Alanları ve Kullanılan Yazılımlara Genel Bir Bakış, Afyon Kocatepe Fen ve Mühendislik Bilim Dergisi, Afyon.

RIEGL, 2014-a. LiDAR Test Report of TURKEY

Şehsuvaroğlu M.S., Eker O., Erdoğan M., Kıyı A. ve Yıldız F., 2015. Farklı Veri Yoğunluğuna Sahip Lidar Nokta Bulutlarından Elde Edilen Sonuç Ürünlerinin Karşılaştırılması, TUFUSB VIII. Teknik Sempozyum, Konya.

Turfan, M., 2002. Baraj Nedir? Devlet Su İşleri Genel Müdürlüğü Yayınları.

URL 1: http://www.dsi.gov.tr, DSİ Web Sayfası, Toprak ve Su Kaynakları, 30.01.2018.

URL 2: https://lidarnews.com, 18.10.2017

URL 3: https://www.hgk.msb.gov.tr/haber-558-bergama-test-bölgesine-ait-hiperspektral-goruntulerin-temini.html

Wehr A. and Lohr U., 1999, Airborne Laser Scanning, ISPRS Journal, Vol. 54, Issue 2/3 .

Zhang Hong, 2006. GIS Algorithm Elements. Science Press, Beijing, pp.41-43.

ÖZGEÇMİŞ

1989 yılında Artvin ili Borçka ilçesinde doğdu. İlköğretim eğitimini G.S.D. Eğitim Vakfı İlköğretim Okulu'nda ortaöğretim eğitimini ise Bahçelievler Bağlar Lisesi'nde tamamladı. 2007 yılında Karadeniz Teknik Üniversitesi, Gümüşhane Mühendislik Fakültesi, Jeodezi ve Fotogrametri Mühendisliği Bölümü'nü kazandı. 2011 yılı Haziran ayında bölümden mezun oldu. 2011-2012 eğitim-öğretim yılında Karadeniz Teknik Üniversitesi, Yabancı Diller Yüksek Okulu'nda eğitime başladı ve 2012 yılı Haziran ayında dil eğitimini tamamladı. 2012-2013 eğitim-öğretim yılında Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Harita Mühendisliği Anabilim dalında Yüksek Lisans programına başladı. Ders dönemini bitirdikten sonra tez aşamasında özel sektörde çalışmaya başladı. 2 yıl özel sektörde çalıştıktan sonra Artvin Çoruh Üniversitesi, Artvin Meslek Yüksekokulu'nda Öğretim Görevlisi olarak göreve başladı ve halen bu görevine devam etmektedir. İngilizce bilmektedir.

6. EKLER

Ek.1 (lasdata.m dosyası)

```
classdef lasdata < handle
    %lasdata LAS data reader / writer / converter
    %   Reads LAS files in format 1.0,1.1,1.2,1.3 and 1.4, supports point
    %   formats 0-10. This tool is intended for quick manual LAS file open,
    %   edit, repair and saving.
    %
    %   Load LAS file into class with c = lasdata('file.las');
    %   Default load only loads xyz coordinates. Additional variables can
    %   be loaded with the accessor functions get_(variable name).
    %
    %   Not much checks are made for the modified data. Feel free to modify
    %   class member variables, but don't break your data. ;)
    %
    %   Data is written in the format set in the class header, except
    %   that the target format can be given with the write_las function.
    %
    %   Copyright (C) Teemu Kumpumki / Tampere University of Technology 2014
        %       Licence: see the included BSD licence.
    properties
        x; %X coordinate
        y; %Y coordinate
        z; %Z coordinate
        intensity; %Return intensity
```

Ek 1’in devamı

```
bits; %Data in bitfields
bits2;%Data in bitfields from 1.4+
classification; %Classification
user_data; %User data
scan_angle; %Scan angle
point_source_id; %Point source id
gps_time; %GPS timestamp
red; %Red color channel
green; %Green color channel
blue; %Blue color channel
nir; %Near infrared color channel
extradata; %extra data found at the end of the point record
Xt; %Waveform line x(t) parameter
Yt; %Waveform line y(t) parameter
Zt; %Waveform line z(t) parameter
wave_return_point; %Time in picoseconds from wave recording start
wave_packet_descriptor; %Waveform packet descriptor index
wave_byte_offset; %Byte offset to waveform data (in/ext file)
wave_packet_size; %Size of single waveform packet
header; %Contain LAS header
variablerecords; %Contain LAS variable records
extendedvariables; %Contains LAS extended variable records
selection; %Filtering logical index
wavedescriptors; %Waveform descriptor structures
filename; %Name of the datafile loaded
```

Ek 1'in devamı

```
    end
    properties (Access=private)
        originalname;
        waveformfile; %name of the waveform file
        waveformfilefid;
        isLAZ; %laz compression check (to delete temporary file in destructor)
    end
methods (Access=public)
    function obj = lasdata(filename,command)
        obj.waveformfilefid = -1;
        obj.filename = filename;
        obj.originalname = filename;
        if exist('command','var') && ischar(command)
            if strcmp(command,'createemptyobj')
                obj.createemptyheader();
                disp('Empty object created, please set header & data values and then write las
file. You need to read the LAS specification for header values in each format.')
            elseif strcmp(command,'loadall')
                obj.las_read(filename);
                %try reading all variables
                obj.read_intensity();
                obj.read_classification();
                obj.read_user_data();
                obj.read_scan_angle();
                obj.read_point_source_id();
                obj.read_point_wave_info();
```

Ek 1’in devamı

```
            obj.read_gps_time();
            obj.read_bits();
            obj.read_extradata();
            obj.read_color();
        else
            error(['Invalid command: ' command]);
        end
    else
        obj.las_read(filename);
    end
end
function delete(obj)
    if obj.waveformfilefid > 2
        fclose(obj.waveformfilefid);
    end
    if obj.isLAZ
        delete([obj.filename]);
end
function obj = normalize_xyz( obj )
    if isempty(obj.x)
        warning('No data loaded.')
        return;
    end
    obj.x = double(obj.x) * obj.header.scale_factor_x + obj.header.x_offset;
    obj.y = double(obj.y) * obj.header.scale_factor_y + obj.header.y_offset;
```

Ek 1'in devamı

```
        obj.z = double(obj.z) * obj.header.scale_factor_z + obj.header.z_offset;
    end
    function data = toint32_xyz( obj )
        if isempty(obj.x)
            warning('No data loaded.')
            return;
        end
        if isa(obj.x,'int32')
            return;
        end
        data(:,1) = obj.x - obj.header.x_offset;
        data(:,2) = obj.y - obj.header.y_offset;
        data(:,3) = obj.z - obj.header.z_offset;
        data(:,1) = round(data(:,1) / obj.header.scale_factor_x);
        data(:,2) = round(data(:,2) / obj.header.scale_factor_y);
        data(:,3) = round(data(:,3) / obj.header.scale_factor_z);
        data = int32(data);
    end
    function x = get_x(obj)
        x = obj.x;
    end
    function y = get_y(obj)
        y = obj.y;
    end
    function z = get_z(obj)
```

Ek 1'in devamı

```
    z = obj.z;
  end
  function xy = get_xy(obj)
    xy = [obj.x obj.y];
  end
  function xyz = get_xyz(obj)
    xyz = [obj.x obj.y obj.z];
  end
  function intensity = get_intensity(obj)
    if isempty(obj.intensity)
      obj.read_intensity();
    end
    intensity = obj.intensity;
  end
  function classification = get_classification(obj)
    if isempty(obj.classification)
      obj.read_classification();
    end
    classification = obj.classification;
  end
  function user_data = get_user_data(obj)
    if isempty(obj.user_data)
      obj.read_user_data();
    end
    user_data = obj.user_data;
```

Ek 1'in devamı

```
    end
    function scan_angle = get_scan_angle(obj)
        if isempty(obj.scan_angle)
            obj.read_scan_angle();
        end
        scan_angle = obj.scan_angle;
    end
    function point_source_id = get_point_source_id(obj)
        if isempty(obj.point_source_id)
            obj.read_point_source_id();
        end
        point_source_id = obj.point_source_id;
    end
    function gps_time = get_gps_time(obj)
        if isempty(obj.gps_time)
            obj.read_gps_time();
        end
        gps_time = obj.gps_time;
    end
    function color = get_color(obj)
        if isempty(obj.red)
            obj.read_color();
        end
        if any(obj.header.point_data_format == [8 10])
            color = [obj.red obj.green obj.blue obj.nir];
```

Ek 1'in devamı

```
        else
            color = [obj.red obj.green obj.blue];
        end
    end
    function waveXYZ = get_waveXYZ(obj)
        if isempty(obj.Xt)
            obj.read_point_wave_info();
        end
        waveXYZ = [obj.Xt obj.Yt obj.Zt];
    end
    function return_point = get_wave_return_point(obj)
        if isempty(obj.Xt)
            obj.read_point_wave_info();
        end
        return_point = obj.wave_return_point;
    end
    function wavedesc = get_wave_descriptor(obj)
        if isempty(obj.Xt)
            obj.read_point_wave_info();
        end
        wavedesc = obj.wavedescriptors;
    end
    function returns = get_return_number(obj)
        if isempty(obj.bits)
            obj.read_bits();
```

Ek 1'in devamı

```
        end
        if obj.header.point_data_format < 6
            returns = bitand(obj.bits,7);
        else
            returns = bitand(obj.bits,15);
        end
    end
    function returns = get_number_of_returns(obj)
        if isempty(obj.bits)
            obj.read_bits();
        end
        if obj.header.point_data_format < 6
            returns = bitshift(bitand(obj.bits,56),-3);
        else
            returns = bitshift(bitand(obj.bits,240),-4);
        end
    end
    function returns = get_classification_flags(obj)
        if isempty(obj.bits)
            obj.read_bits();
        end
        if obj.header.point_data_format < 6
            returns = [];
        else
            returns = bitand(obj.bits2,15);
```

Ek 1'in devamı

```
        end
    end
    function returns = get_scan_direction_flag(obj)
        if isempty(obj.bits)
            obj.read_bits();
        end
        if obj.header.point_data_format < 6
            returns = bitand(obj.bits,64);
        else
            returns = bitand(obj.bits2,64);
        end
    end
    function returns = get_edge_of_flight_line(obj)
        if isempty(obj.bits)
            obj.read_bits();
        end
        if obj.header.point_data_format < 6
            returns = bitand(obj.bits,128);
        else
            returns = bitand(obj.bits2,128);
        end
    end
    function returns = get_scanner_channel(obj)
        if isempty(obj.bits)
            obj.read_bits();
```

Ek 1'in devamı

```
        end
        if obj.header.point_data_format < 6
            returns = [];
        else
            returns = bitshift(bitand(obj.bits2,48),-4);
        end
    end
    function waves = getwaveforms(obj,points)
        if isempty(obj.wave_packet_descriptor)
            obj = read_point_wave_info(obj);
        end
        bitstrs = cell(length(obj.wavedescriptors),1);
        for k=1:length(obj.wavedescriptors)
            bitstrs{k} = ['*ubit'
num2str(obj.wavedescriptors(obj.wave_packet_descriptor(k)).bits)];
        end
        fid = fopen(obj.waveformfile,'r'); %multithreading fix
        DATASTART = obj.header.start_of_waveform_data;
        waves = cell(length(points),1);
        for k=1:length(points)
            p = points(k);
            if fseek(fid,double(DATASTART+obj.wave_byte_offset(p)),-1)<0
                error('fseek failed');
            end
            if obj.wave_packet_descriptor(p) %if wavepacket id not zero then waveform exists
                DATAPOINTS = floor(double(obj.wave_packet_size(p))*8 / ...
```

Ek 1'in devamı

```
                double(obj.wavedescriptors(obj.wave_packet_descriptor(p)).bits));
            waves{k} = fread(fid,DATAPOINTS,...
                bitstrs{obj.wave_packet_descriptor(p)});
        else
            waves{k} = [];
        end
    end
    fclose(fid);
end
function obj = setfilter(obj,varargin)
% setfilter adds given filter command to the filter stack.
% To reset filtering, reload the class from the original data.
% Specify filter with setfilter('filter',value,'filter',value...)
% or give filters sequentially obj.setfilter(...); obj.setfilter(...),
% however this method is slower as it causes reloading of the data.
    if nargin < 3 %+1 comes from the class obj
        error('Incorrect parameter count.');
    end
    %clear filter
    if isempty(obj.selection)
        obj.selection = true(obj.header.number_of_point_records,1);
    end
    k=1;
    while k<=length(varargin)
        switch( varargin{k} )
```

Ek 1’in devamı

```
            case 'area'
                area = varargin{k+1};
                if numel(area) ~=4
                    error('Area definition requires rectangle [x,y,width,height].')
                end
                %reload if data is not in the original condition
                if isempty(obj.x) || ( length(obj.x) ~= length(obj.selection) )
                    obj.read_xyz(1);
                end
                obj.selection = obj.selection & ...
                    (obj.x >= area(1) & obj.x<= area(1)+area(3) & ...
                    obj.y >= area(2) & obj.y<= area(2)+area(4));
                k = k+2;
            case 'classification'
                val = varargin{k+1};
                if ~isnumeric(val)
                    error('Classification comparison requires a vector input of classification
codes.')
                end
                %reload if data is not in the original condition
                if isempty(obj.classification) || ( length(obj.classification) ~=
length(obj.selection) )
                    obj.read_classification(1);
                end
                obj.selection = obj.selection & ( ismember(obj.classification, val) );
                k = k+2;
```

Ek 1’in devamı

```
        case 'user_data'
            val = varargin{k+1};
            if numel(val) ~=1 || ~isnumeric(val)
                error('User data comparison requires a single number.')
            end
            %reload if data is not in the original condition
            if isempty(obj.user_data) || ( length(obj.user_data) ~= length(obj.selection) )
                obj.read_user_data(1);
            end
            obj.selection = obj.selection & (obj.user_data == val);
            k = k+2;
        case 'scan_angle'
            operator = varargin{k+1};
            if ~ischar(operator) && any(strncmp(operator,{'==','~=','<','>','<=','>='},2))
                error('Comparison operator must be one of [==,~=,<,>,<=,>=].')
            end
            val = varargin{k+2};
            if numel(val) ~=1 || ~isnumeric(val)
                error('Scan angle comparison requires a single number.')
            end
            %reload if data is not in the original condition
            if isempty(obj.scan_angle) || ( length(obj.scan_angle) ~= length(obj.selection)
                obj.read_scan_angle(1);
            end
            eval(['obj.selection = obj.selection & (obj.scan_angle ' operator ' val);']);
```

Ek 1’in devamı

```
                k = k+3;
            case 'custom'
                fil = varargin{k+1};
                if length(fil) == length(obj.x)
                    idx = find(obj.selection);
                    idx = fil(:).*idx;
                    fil = false(size(obj.selection));
                    idx(idx==0) = [];
                    fil(idx) = true;
                end
                if numel(fil) ~=length(obj.selection) || ~islogical(fil)
                    error('Custom filter must be as long as is the length of the original or current
data')
                end
                obj.selection = obj.selection & fil;
                k = k+2;
            otherwise
                error('Unknown filtering command');
        end
    end
    %reload existing datas and set filter
    if length(obj.x) ~= length(obj.selection)
        obj.read_xyz(1);
    end
    obj.x = obj.x(obj.selection);
    obj.y = obj.y(obj.selection);
```

Ek 1'in devamı

```
obj.z = obj.z(obj.selection);
if ~isempty(obj.intensity) && ( length(obj.intensity) ~= length(obj.selection) )
   obj.read_intensity(1);
end
if ~isempty(obj.intensity)
   obj.intensity = obj.intensity(obj.selection);
end
if ~isempty(obj.classification) && ( length(obj.classification) ~= length(obj.selection)
   obj.read_classification(1);
end
if ~isempty(obj.classification)
   obj.classification = obj.classification(obj.selection);
end
if ~isempty(obj.bits) && ( length(obj.bits) ~= length(obj.selection) )
   obj.read_bits(1);
end
if ~isempty(obj.bits)
   obj.bits = obj.bits(obj.selection);
   if ~isempty(obj.bits2)
      obj.bits2 = obj.bits2(obj.selection);
   end
end
if ~isempty(obj.user_data) && ( length(obj.user_data) ~= length(obj.selection) )
   obj.read_user_data(1);
end
```

Ek 1'in devamı

```
    if ~isempty(obj.user_data)
       obj.user_data = obj.user_data(obj.selection);
    end
    if ~isempty(obj.scan_angle) && ( length(obj.scan_angle) ~= length(obj.selection) )
       obj.read_scan_angle(1);
    end
    if ~isempty(obj.scan_angle)
       obj.scan_angle = obj.scan_angle(obj.selection);
    end
    if ~isempty(obj.point_source_id) && ( length(obj.point_source_id) ~=
length(obj.selection) )
       obj.read_point_source_id(1);
    end
    if ~isempty(obj.point_source_id)
       obj.point_source_id = obj.point_source_id(obj.selection);
    end
    if ~isempty(obj.gps_time) && ( length(obj.gps_time) ~= length(obj.selection) )
       obj.read_gps_time(1);
    end
    if ~isempty(obj.gps_time)
       obj.gps_time = obj.gps_time(obj.selection);
    end
    if ~isempty(obj.red) && ( length(obj.red) ~= length(obj.selection) )
       obj.read_color(1);
    end
    if ~isempty(obj.red)
```

Ek 1'in devamı

```
        obj.red = obj.red(obj.selection);
        obj.green = obj.green(obj.selection);
        obj.blue = obj.blue(obj.selection);
        if ~isempty(obj.nir)
            obj.nir = obj.nir(obj.selection);
        end
    end
    if ~isempty(obj.extradata) && ( length(obj.extradata) ~= length(obj.selection) )
        obj.read_extradata(1);
    end
    if ~isempty(obj.extradata)
        obj.extradata = obj.extradata(obj.selection,:);
    end
    if ~isempty(obj.Xt) && ( length(obj.Xt) ~= length(obj.selection) )
        obj.read_point_wave_info(1);
    end
    if ~isempty(obj.Xt)
        obj.Xt = obj.Xt(obj.selection);
        obj.Yt = obj.Yt(obj.selection);
        obj.Zt = obj.Zt(obj.selection);
        obj.wave_return_point = obj.wave_return_point(obj.selection);
        obj.wave_packet_descriptor = obj.wave_packet_descriptor(obj.selection);
        obj.wave_byte_offset = obj.wave_byte_offset(obj.selection);
        obj.wave_packet_size = obj.wave_packet_size(obj.selection);
    end
```

Ek 1'in devamı

```
    end
    function obj = read_xyz(obj,donotfilter)
        fid = fopen(obj.filename);
        fseek(fid,double(obj.header.offset_to_point_data),-1);
        LEN = obj.header.point_data_record_length;
        POINTS = obj.header.number_of_point_records;
        OFFSET = obj.header.offset_to_point_data;
        fseek(fid,OFFSET,-1);
        obj.x = fread(fid,POINTS,'*int32',LEN-4);  %x
        fseek(fid,OFFSET+4,-1);
        obj.y = fread(fid,POINTS,'*int32',LEN-4);  %y
        fseek(fid,OFFSET+8,-1);
        obj.z = fread(fid,POINTS,'*int32',LEN-4);  %z
        fclose(fid);
        %apply filter
        if ~exist('donotfilter','var')
            obj.x = obj.x(obj.selection);
            obj.y = obj.y(obj.selection);
            obj.z = obj.z(obj.selection);
        end
        obj.normalize_xyz();
    end
    function obj = write_xyz(obj)
        fid = fopen(obj.header.filename,'r+');
        fseek(fid,double(obj.header.offset_to_point_data),-1);
```

Ek 1'in devamı

```
        LEN = obj.header.point_data_record_length;
        OFFSET = 0;
        fseek(fid,double(obj.header.offset_to_point_data),-1);
        obj.columndatafwrite(fid,obj.toint32_xyz(),OFFSET,LEN);
        fclose(fid);
    end
    function obj = read_intensity(obj,donotfilter)
        fid = fopen(obj.filename);
        LEN = obj.header.point_data_record_length;
        POINTS = obj.header.number_of_point_records;
        OFFSET = obj.header.offset_to_point_data + 12;
        fseek(fid,OFFSET,-1);
        obj.intensity = fread(fid,POINTS,'*uint16',LEN-2);%intensity
        fclose(fid);
        %apply filter
        if ~exist('donotfilter','var')
            obj.intensity = obj.intensity(obj.selection);
        end
    end
    function obj = write_intensity(obj)
        fid = fopen(obj.header.filename,'r+');
        LEN = obj.header.point_data_record_length;
        OFFSET = 12;
        fseek(fid,double(obj.header.offset_to_point_data),-1);
        obj.columndatafwrite(fid,obj.intensity,OFFSET,LEN);
```

Ek 1'in devamı

```
        fclose(fid);
    end
    function obj = read_bits(obj,donotfilter)
        fid = fopen(obj.filename);
        LEN = obj.header.point_data_record_length;
        POINTS = obj.header.number_of_point_records;
        OFFSET = obj.header.offset_to_point_data + 14;
        fseek(fid,OFFSET,-1);
        obj.bits = fread(fid,POINTS,'*uint8',LEN-1);  %bits
        if obj.header.version_minor > 3 && obj.header.point_data_format > 5 %1.4 &&
pointformat >=6
            fseek(fid,OFFSET+1,-1);
            obj.bits2 = fread(fid,POINTS,'*uint8',LEN-1);  %bits2
        end
        fclose(fid);
        %apply filter
        if ~exist('donotfilter','var')
            obj.bits = obj.bits(obj.selection);
            if ~isempty(obj.bits2)
                obj.bits2 = obj.bits2(obj.selection);
            end
        end
    end
    function obj = write_bits(obj)
        fid = fopen(obj.header.filename,'r+');
        LEN = obj.header.point_data_record_length;
```

Ek 1’in devamı

```
        OFFSET = 14;
        fseek(fid,double(obj.header.offset_to_point_data),-1);
        obj.columndatafwrite(fid,obj.bits,OFFSET,LEN);
        if obj.header.version_minor > 3 && obj.header.point_data_format > 5 %1.4 &
pointformat >=6
            fseek(fid,double(obj.header.offset_to_point_data),-1);
            obj.columndatafwrite(fid,obj.bits2,OFFSET+1,LEN);
        end
        fclose(fid);
    end
    function obj = read_classification(obj,donotfilter)
        fid = fopen(obj.filename);
        fseek(fid,double(obj.header.offset_to_point_data),-1);
        LEN = obj.header.point_data_record_length;
        POINTS = obj.header.number_of_point_records;
        offsettable = [15 15 15 15 15 15 16 16 16 16 16];
        OFFSET =
offsettable(obj.header.point_data_format+1)+obj.header.offset_to_point_data;
        fseek(fid,OFFSET,-1);
        obj.classification= fread(fid,POINTS,'*uint8',LEN-1);
        fclose(fid);
        %apply filter
        if ~exist('donotfilter','var')
            obj.classification = obj.classification(obj.selection);
        end
    end
```

Ek 1'in devamı

```
    function obj = write_classification(obj)
        fid = fopen(obj.header.filename,'r+');
        LEN = obj.header.point_data_record_length;
        offsettable = [15 15 15 15 15 15 16 16 16 16 16];
        OFFSET = offsettable(obj.header.point_data_format+1);
        fseek(fid,double(obj.header.offset_to_point_data),-1);
        obj.columndatafwrite(fid,obj.classification,OFFSET,LEN);
        fclose(fid);
    end
    function obj = read_scan_angle(obj,donotfilter)
        fid = fopen(obj.filename);
        LEN = obj.header.point_data_record_length;
        POINTS = obj.header.number_of_point_records;
        offsettable = [16 16 16 16 16 16 18 18 18 18 18];
        datatypetable = {'*int8', '*int8', '*int8', '*int8', '*int8', ...
            '*int8', '*int16', '*int16', '*int16', '*int16', '*int16'};
        datasizetable = [1 1 1 1 1 1 2 2 2 2 2];
        OFFSET =
offsettable(obj.header.point_data_format+1)+obj.header.offset_to_point_data;
        DATATYPE = datatypetable{obj.header.point_data_format+1};
        DATASIZE = datasizetable(obj.header.point_data_format+1);
        fseek(fid,OFFSET,-1);
        obj.scan_angle= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
        fclose(fid);
        %apply filter
        if ~exist('donotfilter','var')
```

Ek 1'in devamı

```
            obj.scan_angle = obj.scan_angle(obj.selection);
        end
    end
    function obj = write_scan_angle(obj)
        fid = fopen(obj.header.filename,'r+');
        LEN = obj.header.point_data_record_length;
        offsettable = [16 16 16 16 16 16 18 18 18 18 18];
        OFFSET = offsettable(obj.header.point_data_format+1);
        datatypetable = {'int8', 'int8', 'int8', 'int8', 'int8', ...
            'int8', 'int16', 'int16', 'int16', 'int16', 'int16'};
        DATATYPE = datatypetable{obj.header.point_data_format+1};
        if ~isa(obj.scan_angle,DATATYPE)
            error(['Scan angle datatype is not: ' DATATYPE])
        end
        fseek(fid,double(obj.header.offset_to_point_data),-1);
        obj.columndatafwrite(fid,obj.scan_angle,OFFSET,LEN);
        fclose(fid);
    end
    function obj = read_user_data(obj,donotfilter)
        fid = fopen(obj.filename);
        LEN = obj.header.point_data_record_length;
        POINTS = obj.header.number_of_point_records;
        offsettable = [17 17 17 17 17 17 17 17 17 17 17];
        OFFSET = offsettable(obj.header.point_data_format+1);
        DATATYPE ='*uint8';
```

Ek 1'in devamı

```
        DATASIZE = 1;
        fseek(fid,OFFSET,-1);
        obj.user_data= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
        fclose(fid);
        %apply filter
        if ~exist('donotfilter','var')
            obj.user_data = obj.user_data(obj.selection);
        end
    end
    function obj = write_user_data(obj)
        fid = fopen(obj.header.filename,'r+');
        LEN = obj.header.point_data_record_length;
        DATATYPE ='uint8';
        offsettable = [17 17 17 17 17 17 17 17 17 17 17];
        OFFSET = offsettable(obj.header.point_data_format+1);
        if ~isa(obj.user_data,DATATYPE)
            error(['User data datatype is not: ' DATATYPE])
        end
        fseek(fid,double(obj.header.offset_to_point_data),-1);
        obj.columndatafwrite(fid,obj.user_data,OFFSET,LEN);
        fclose(fid);
    end
    function obj = read_point_source_id(obj,donotfilter)
        fid = fopen(obj.filename);
        LEN = obj.header.point_data_record_length;
```

Ek 1'in devamı

```
        POINTS = obj.header.number_of_point_records;
        offsettable = [18 18 18 18 18 18 20 20 20 20 20];
        OFFSET =
offsettable(obj.header.point_data_format+1)+obj.header.offset_to_point_data;
        DATATYPE = '*uint16';
        DATASIZE = 2;
        fseek(fid,OFFSET,-1);
        obj.point_source_id= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
        fclose(fid);
        %apply filter
        if ~exist('donotfilter','var')
            obj.point_source_id = obj.point_source_id(obj.selection);
        end
    end
    function obj = write_point_source_id(obj)
        fid = fopen(obj.header.filename,'r+');
        LEN = obj.header.point_data_record_length;
        offsettable = [18 18 18 18 18 18 20 20 20 20 20];
        OFFSET = offsettable(obj.header.point_data_format+1);
        DATATYPE = 'uint16';
        if ~isa(obj.point_source_id,DATATYPE)
            error(['Point source id datatype is not: ' DATATYPE])
        end
        fseek(fid,double(obj.header.offset_to_point_data),-1);
        obj.columndatafwrite(fid,obj.point_source_id,OFFSET,LEN);
        fclose(fid);
```

Ek 1'in devamı

```
    end
    function obj = read_gps_time(obj,donotfilter)
        %check if not in this point format
        if any(obj.header.point_data_format == [0 2])
            return;
        end
        fid = fopen(obj.filename);
        LEN = obj.header.point_data_record_length;
        POINTS = obj.header.number_of_point_records;
        offsettable = [20 20 20 20 20 20 22 22 22 22 22];
        OFFSET =
offsettable(obj.header.point_data_format+1)+obj.header.offset_to_point_data;
        DATATYPE = '*double';
        DATASIZE = 8;
        fseek(fid,OFFSET,-1);
        obj.gps_time= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
        fclose(fid);
        %apply filter
        if ~exist('donotfilter','var')
            obj.gps_time = obj.gps_time(obj.selection);
        end
    end
    function obj = write_gps_time(obj)
        fid = fopen(obj.header.filename,'r+');
        %check if not in this point format
        if any(obj.header.point_data_format == [0 2])
```

Ek 1'in devamı

```
            return;
        end
        LEN = obj.header.point_data_record_length;
        offsettable = [20 20 20 20 20 20 22 22 22 22 22];
        OFFSET = offsettable(obj.header.point_data_format+1);
        DATATYPE = 'double';
        if ~isa(obj.gps_time,DATATYPE)
            error(['GPS time datatype is not: ' DATATYPE])
        end
        fseek(fid,double(obj.header.offset_to_point_data),-1);
        obj.columndatafwrite(fid,obj.gps_time,OFFSET,LEN);
        fclose(fid);
    end
    function obj = read_color(obj,donotfilter)
        %check if not in this point format
        if any(obj.header.point_data_format == [0 1 2 4 6 9])
            return;
        end
        fid = fopen(obj.filename);
        LEN = obj.header.point_data_record_length;
        POINTS = obj.header.number_of_point_records;
        offsettable = [20 20 20 28 28 28 30 30 30 30 30];
        OFFSET =
offsettable(obj.header.point_data_format+1)+obj.header.offset_to_point_data;
        DATATYPE = '*uint16';
        DATASIZE = 2;
```

Ek 1'in devamı

```
        fseek(fid,OFFSET,-1);
        obj.red= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
        fseek(fid,OFFSET+2,-1);
        obj.green= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
        fseek(fid,OFFSET+4,-1);
        obj.blue= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
        if any(obj.header.point_data_format == [8 10])
            fseek(fid,OFFSET+6,-1);
            obj.nir= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
        end
        fclose(fid);
        %apply filter
        if ~exist('donotfilter','var')
            obj.red = obj.red(obj.selection);
            obj.green = obj.green(obj.selection);
            obj.blue = obj.blue(obj.selection);
            if ~isempty(obj.nir)
                obj.nir = obj.nir(obj.selection);
            end
        end
    end
    function obj = write_color(obj)
        fid = fopen(obj.header.filename,'r+');
        %check if not in this point format
        if any(obj.header.point_data_format == [0 1 2 4 6 9])
```

Ek 1'in devamı

```
            return;
        end
        LEN = obj.header.point_data_record_length;
        offsettable = [20 20 20 28 28 28 30 30 30 30 30];
        OFFSET = offsettable(obj.header.point_data_format+1);
        DATATYPE = 'uint16';
        if ~isa(obj.red,DATATYPE)
            error(['Color datatype is not: ' DATATYPE])
        end
        fseek(fid,double(obj.header.offset_to_point_data),-1);
        obj.columndatafwrite(fid,[obj.red obj.green obj.blue],OFFSET,LEN);
        if any(obj.header.point_data_format == [8 10])
            fseek(fid,double(obj.header.offset_to_point_data),-1);
            obj.columndatafwrite(fid,obj.nir,OFFSET+6,LEN);
        end
        fclose(fid);
    end
    function obj = read_point_wave_info(obj,donotfilter)
        %check if not in this point format
        if any(obj.header.point_data_format == [0 1 2 3 6 7 8])
            return;
        end
        fid = fopen(obj.filename);
        LEN = obj.header.point_data_record_length;
        POINTS = obj.header.number_of_point_records;
```

Ek 1’in devamı

```
offsettable = [28 28 28 28 28 28 28 28 28 30 38];
OFFSET = offsettable(obj.header.point_data_format+1)+obj.header.offset_to_point_data;
DATATYPE = '*uint8';
DATASIZE = 1;
fseek(fid,OFFSET,-1);
obj.wave_packet_descriptor= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
DATATYPE = '*uint64';
DATASIZE = 8;
fseek(fid,OFFSET+1,-1);
obj.wave_byte_offset= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
DATATYPE = '*uint32';
DATASIZE = 4;
fseek(fid,OFFSET+9,-1);
obj.wave_packet_size= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
DATATYPE = '*single';
DATASIZE = 4;
fseek(fid,OFFSET+13,-1);
obj.wave_return_point= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
DATATYPE = '*single';
DATASIZE = 4;
fseek(fid,OFFSET+17,-1);
obj.Xt= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
DATATYPE = '*single';
DATASIZE = 4;
fseek(fid,OFFSET+21,-1);
```

Ek 1'in devamı

```
        obj.Yt= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
        DATATYPE = '*single';
        DATASIZE = 4;
        fseek(fid,OFFSET+25,-1);
        obj.Zt= fread(fid,POINTS,DATATYPE,LEN-DATASIZE);
        fclose(fid);
        %apply filter
        if ~exist('donotfilter','var')
            obj.wave_packet_descriptor = obj.wave_packet_descriptor(obj.selection);
            obj.wave_byte_offset = obj.wave_byte_offset(obj.selection);
            obj.wave_packet_size = obj.wave_packet_size(obj.selection);
            obj.wave_return_point = obj.wave_return_point(obj.selection);
            obj.Xt = obj.Xt(obj.selection);
            obj.Yt = obj.Yt(obj.selection);
            obj.Zt = obj.Zt(obj.selection);
        end
    end
    function obj = write_point_wave_info(obj)
        fid = fopen(obj.header.filename,'r+');
        %check if not in this point format
        if any(obj.header.point_data_format == [0 1 2 3 6 7 8])
            return;
        end
        LEN = obj.header.point_data_record_length;
        offsettable = [28 28 28 28 28 28 28 28 28 30 38];
```

Ek 1'in devamı

```
DATATYPE = 'uint8';
if ~isa(obj.wave_packet_descriptor,DATATYPE)
   error(['Wave packet descriptor datatype is not: ' DATATYPE])
end
fseek(fid,double(obj.header.offset_to_point_data),-1);
OFFSET = offsettable(obj.header.point_data_format+1);
obj.columndatafwrite(fid,obj.wave_packet_descriptor,OFFSET,LEN);
DATATYPE = 'uint64';
if ~isa(obj.wave_byte_offset,DATATYPE)
   error(['Wave byte offset datatype is not: ' DATATYPE])
end
fseek(fid,double(obj.header.offset_to_point_data),-1);
OFFSET = offsettable(obj.header.point_data_format+1)+1;
obj.columndatafwrite(fid,obj.wave_byte_offset,OFFSET,LEN);
DATATYPE = 'uint32';
if ~isa(obj.wave_packet_size,DATATYPE)
   error(['Wave packet size datatype is not: ' DATATYPE])
end
fseek(fid,double(obj.header.offset_to_point_data),-1);
OFFSET = offsettable(obj.header.point_data_format+1)+9;
obj.columndatafwrite(fid,obj.wave_packet_size,OFFSET,LEN);
DATATYPE = 'single';
if ~isa(obj.wave_return_point,DATATYPE)
   error(['Wave return point datatype is not: ' DATATYPE])
end
```

Ek 1’in devamı

```
fseek(fid,double(obj.header.offset_to_point_data),-1);
OFFSET = offsettable(obj.header.point_data_format+1)+13;
obj.columndatafwrite(fid,obj.wave_return_point,OFFSET,LEN);
DATATYPE = 'single';
if ~isa(obj.Xt,DATATYPE)
   error(['Xt datatype is not: ' DATATYPE])
end
fseek(fid,double(obj.header.offset_to_point_data),-1);
OFFSET = offsettable(obj.header.point_data_format+1)+17;
obj.columndatafwrite(fid,obj.Xt,OFFSET,LEN);
DATATYPE = 'single';
if ~isa(obj.Yt,DATATYPE)
   error(['Yt datatype is not: ' DATATYPE])
end
fseek(fid,double(obj.header.offset_to_point_data),-1);
OFFSET = offsettable(obj.header.point_data_format+1)+21;
obj.columndatafwrite(fid,obj.Yt,OFFSET,LEN);
DATATYPE = 'single';
if ~isa(obj.Zt,DATATYPE)
   error(['Zt datatype is not: ' DATATYPE])
end
fseek(fid,double(obj.header.offset_to_point_data),-1);
OFFSET = offsettable(obj.header.point_data_format+1)+25;
obj.columndatafwrite(fid,obj.Zt,OFFSET,LEN);
fclose(fid);
```

Ek 1’in devamı

```
    end
    function obj = read_extradata(obj)
        LEN = obj.header.point_data_record_length;
        POINTS = obj.header.number_of_point_records;
        offsettable = [20 28 26 34 57 63 30 36 38 59 67]; %magic numbers from point record byte lengths
        OFFSET = offsettable(obj.header.point_data_format+1)+obj.header.offset_to_point_data;
        extralen = obj.header.point_data_record_length - offsettable(obj.header.point_data_format+1);
        if extralen>0 %unknown extra data exists
            fid = fopen(obj.filename);
            fseek(fid,OFFSET,-1);
            obj.extradata = zeros(POINTS,extralen,'uint8');
            for k=1:POINTS
                obj.extradata(k,:) = fread(fid,extralen,'*uint8');
                fseek(fid,LEN-extralen,0);
            end
            fclose(fid);
        end
        %apply filter
        if ~exist('donotfilter','var') && ~isempty(obj.extradata)
            obj.extradata = obj.extradata(obj.selection,:);
        end
    end
    function obj = write_extradata(obj)
        fid = fopen(obj.header.filename,'r+');
```

Ek 1'in devamı

```
    LEN = obj.header.point_data_record_length;
    offsettable = [20 28 26 34 57 63 30 36 38 59 67]; %magic numbers from point record byte lengths
    OFFSET = offsettable(obj.header.point_data_format+1);
    extralen = size(obj.extradata,2);
    if extralen %unknown extra data exists
      if ~isa(obj.extradata,'uint8')
        error(['Row extra data datatype is not: uint8'])
      end
      fseek(fid,double(obj.header.offset_to_point_data),-1);
      obj.columndatafwrite(fid,obj.extradata,OFFSET,LEN);
    end
    fclose(fid);
  end
  function plot_xyz(obj,pointlimit)
    if ~exist('pointlimit','var')
      pointlimit = 10000;
    end
    if pointlimit == -1
      pointlimit = length(obj.x);
    end
    if isempty(obj.x)
      warning('No points to plot, check filtering!')
      return;
    end
    sel = randi(length(obj.x),pointlimit,1);
```

Ek 1’in devamı

```
    set(gcf,'renderer','opengl')
    obj.get_intensity();
    scatter3(obj.x(sel),obj.y(sel),obj.z(sel),100,obj.intensity(sel),'r.')
  end
  function plot_intensity(obj,pointlimit)
    if ~exist('pointlimit','var')
      pointlimit = 10000;
    end
    if pointlimit == -1
      pointlimit = length(obj.x);
    end
    if isempty(obj.x)
      warning('No points to plot, check filtering!')
      return;
    end
    sel = randi(length(obj.x),pointlimit,1);
    set(gcf,'renderer','opengl')
    obj.get_intensity();
    scatter3(obj.x(sel),obj.y(sel),obj.z(sel),100,obj.intensity(sel),'.')
  end
  function plot_classification(obj,pointlimit,no_z)
    if ~exist('pointlimit','var')
      pointlimit = 10000;
    end
    if pointlimit == -1
```

Ek 1'in devamı

```
        pointlimit = length(obj.x);
    end
    if isempty(obj.x)
        warning('No points to plot, check filtering!')
        return;
    end
    sel = randi(length(obj.x),pointlimit,1);
    set(gcf,'renderer','opengl')
    obj.get_classification();
    if ~exist('no_z','var')
        scatter(obj.x(sel),obj.y(sel),100,obj.classification(sel),'.')
    else
        scatter3(obj.x(sel),obj.y(sel),obj.z(sel),100,obj.classification(sel),'.')
    end
end
function plot_waveforms(obj,pointlimit,alphaon)
    if ~exist('pointlimit','var')
        pointlimit = 1000;
    end
    if pointlimit == -1
        pointlimit = length(obj.x);
    end
    if isempty(obj.x)
        warning('No points to plot, check filtering!')
        return;
```

Ek 1'in devamı

```
    end
    %select last returns
    subset = find(obj.get_return_number() == obj.get_number_of_returns());
    %and random amount of these points or all
    subset = subset(randi(size(subset,1),pointlimit,1));
    xyz = obj.get_xyz();
    %remove points with no waves
    waves = obj.getwaveforms(subset);
    xyz = xyz(subset,:);
    amplitudemax = double(max(cellfun(@max,waves)));
    firstpeaks = zeros(length(waves),1);
    for k=1:length(waves)
       [~,firstpeaks(k)] =
findpeaks(double(waves{k}),'npeaks',1,'minpeakheight',mean(waves{k}));
    end
    %temporal spacing from descriptor (*1000000 scaling)
    step =
double([obj.wavedescriptors(obj.wave_packet_descriptor(subset)).temporal_sample_spaci
ng])*1000000;
    samples =
double([obj.wavedescriptors(obj.wave_packet_descriptor(subset)).number_of_samples]);
    %calculate height of the waveform
    waveheight = (298925574./step(:)).*samples(:)/2;
    wavegroundoffset = (298925574./step(:)).*firstpeaks(:)/2;
    set(gcf,'renderer','opengl')
    hold on
    colors = jet(amplitudemax+1);
```

Ek 1'in devamı

```
%generate ground
tri = delaunay(xyz(:,1),xyz(:,2));
trisurf(tri,xyz(:,1),xyz(:,2),xyz(:,3),'edgecolor','none')
faces = [];
vertices = [];
col = [];
valpha = [];
len = 0;
for k=1:length(subset)
   tmpx =
linspace(xyz(k,1),xyz(k,1),samples(k))'+double(waves{k})/amplitudemax/2;
   tmpy = linspace(xyz(k,2),xyz(k,2),samples(k))';
   tmpz = linspace(xyz(k,3)-wavegroundoffset(k),...
            xyz(k,3)+waveheight(k)-wavegroundoffset(k),samples(k))';
   tmpc = colors(double(waves{k})+1,:);
   p = [tmpx tmpy tmpz];
   fc = [(1:size(p,1)-1)' (1:size(p,1)-1)' (2:size(p,1))']+ len;
   len = len + size(p,1);
   faces = [faces; fc];
   vertices = [vertices; p];
   col = [col; tmpc];
   valpha = [valpha; double(waves{k})/amplitudemax];
end
if exist('alphaon','var')
   valpha = log(valpha);
   valpha = valpha -(min(valpha));
```

Ek 1'in devamı

```
        valpha = valpha / max(valpha);
        patch( 'Vertices', vertices, 'Faces', faces,'FaceColor','none',...
            'EdgeColor',[0 0 0],...
            'FaceVertexAlphaData', valpha,...
            'EdgeAlpha','interp','AlphaDataMapping','none');
    else
        patch( 'Vertices', vertices, 'Faces', faces,'FaceColor','none',...
            'FaceVertexCData',col,'EdgeColor','interp');
    End
end
function plot_waveform_layers(obj)
    if isempty(obj.x)
        warning('No points to plot, check filtering!')
        return;
    end
    %select last returns
    subset = find(obj.get_return_number() == obj.get_number_of_returns());
    xyz = obj.get_xyz();
    xyz = xyz(subset,:);
    xlimorg = [min(xyz(:,1)) max(xyz(:,1))];
    ylimorg = [min(xyz(:,2)) max(xyz(:,2))];
    ZADD = 30;
    zlimorg = [min(xyz(:,3)) max(xyz(:,3))+ZADD];
    xyz(:,1) = xyz(:,1)-min(xyz(:,1));
    xyz(:,2) = xyz(:,2)-min(xyz(:,2));
```

Ek 1'in devamı

```
xyz(:,3) = xyz(:,3)-min(xyz(:,3));

waves = obj.getwaveforms(subset);

removeempty = cellfun(@isempty,waves);

subset = subset(~removeempty);

xyz = xyz(~removeempty,:);

waves = waves(~removeempty);

%create voxel structure

NX = 256; NY = 256; NZ = 256;

layers = zeros(NX,NY,NZ,'uint8');

xlim = [min(xyz(:,1)) max(xyz(:,1))];

ylim = [min(xyz(:,2)) max(xyz(:,2))];

zlim = [min(xyz(:,3)) max(xyz(:,3))+ZADD];

amplitudemax = double(max(cellfun(@max,waves)));

firstpeaks = zeros(1,length(waves));

for k=1:length(waves)

   [~,firstpeaks(k)] =
findpeaks(double(waves{k}),'npeaks',1,'minpeakheight',mean(waves{k}));

end

%temporal spacing from descriptor (*1000000 scaling)

step =
double([obj.wavedescriptors(obj.wave_packet_descriptor(subset)).temporal_sample_spaci
ng])*1000000;

samples =
double([obj.wavedescriptors(obj.wave_packet_descriptor(subset)).number_of_samples]);

%calculate height of the waveform

waveheight = (298925574./step).*samples/2;

wavegroundoffset = (298925574./step).*firstpeaks/2;
```

Ek 1'in devamı

```
set(gcf,'renderer','opengl')
hold on
for k=1:length(subset)
   tmpx = linspace(xyz(k,1),xyz(k,1),samples(k))';
   tmpy = linspace(xyz(k,2),xyz(k,2),samples(k))';
   tmpz = linspace(xyz(k,3)-wavegroundoffset(k),...
               xyz(k,3)+waveheight(k)-wavegroundoffset(k),samples(k))';
   amp = (double(waves{k})/amplitudemax)*255;
   px = floor((tmpx)/(xlim(2))*(NX-1))+1;
   py = floor((tmpy)/(ylim(2))*(NY-1))+1;
   pz = floor((tmpz)/(zlim(2))*(NZ-1))+1;
   for r=1:length(px)
      if pz(r)<=NZ && pz(r)>0 && layers(px(r),py(r),pz(r)) < amp(r)
         layers(px(r),py(r),pz(r)) = amp(r);
      end
   end
end
alim([0 255]);
for k = 1:NZ
   zk = zlimorg(1)+(zlimorg(2)-zlimorg(1))/NZ*(k-1);
   surf([xlimorg(1) xlimorg(2); xlimorg(1) xlimorg(2)],...
      [ylimorg(1) ylimorg(1); ylimorg(2) ylimorg(2)],...
      [zk zk; zk zk],'CData',layers(:,:,k),'FaceColor','texturemap',...
      'EdgeColor','none','FaceAlpha', 'texturemap', ...
      'AlphaDataMapping', 'scaled', 'AlphaData',layers(:,:,k));
```

Ek 1'in devamı

```
    end
  end
  function obj = write_las(obj, filename, majorversion, minorversion, pointformat)
    %prevent overwriting
    %if you like to overwrite files, then read all variables to memory
    %at this point and disable check
    [pathtmp,filetmp,ext]=fileparts(obj.filename);
    if isempty(pathtmp); pathtmp = pwd; end
    orgfile = [pathtmp '/' filetmp ext];
    [pathtmp,filetmp]=fileparts(filename);
    if isempty(pathtmp); pathtmp = pwd; end
    newfile = [pathtmp '/' filetmp ext];
    if strcmpi(orgfile,newfile)
      error('Overwriting is not allowed.')
    end
    newheader = obj.header;
    oldheader = obj.header;
    if ~exist('filename','var')
      error('Please input target filename.')
    end
    if exist('majorvarsion','var')
      newheader.version_major = majorversion;
    end
    if exist('minorversion','var')
      newheader.version_minor = minorversion;
```

Ek 1’in devamı

```
    end
    if exist('pointformat','var')
       newheader.point_data_format = pointformat;
    end
    newheader.number_of_point_records = length(obj.x);
    newheader.max_x = max(obj.x);
    newheader.min_x = min(obj.x);
    newheader.max_y = max(obj.y);
    newheader.min_y = min(obj.y);
    newheader.max_z = max(obj.z);
    newheader.min_z = min(obj.z);
    newheader.filename = filename;
    fid = fopen(filename,'w');
    try
       obj.header = newheader;
       obj.writeheader(fid);
    catch err
       obj.header = oldheader;
       error(['Error writing las header: ' err.getReport]);
    end
    obj.header = oldheader;
    LEN = length(obj.x);
    if isempty(obj.intensity)
       obj.read_intensity();
       if isempty(obj.intensity)
```

Ek 1'in devamı

```
            warning('Adding zeros to intensity')
            obj.intensity = zeros(LEN,1,'uint16');
        end
    end
    if isempty(obj.bits)
        obj.read_bits();
        if isempty(obj.bits)
            warning('Adding zeros to bit values (return nr, scan dir. flag, edge of flight line)')
            obj.bits = zeros(LEN,1,'uint8');
        end
    end
    if any(newheader.point_data_format == [6 7 8 9 10])
        if isempty(obj.bits2)
            obj.bits2 = zeros(LEN,1,'uint8');
        end
        %convert to new point formats
        if oldheader.point_data_format < 6
            obj.bits = bitor(obj.get_return_number, bitshift(obj.get_number_of_returns,4));
            obj.bits2 = bitor(bitshift(obj.get_scan_direction_flag,6),
bitshift(obj.get_edge_of_flight_line,7));
        end
    end
    if any(oldheader.point_data_format > 5) && newheader.point_data_format < 6
        %convert to old point formats
        if oldheader.point_data_format > 5
```

Ek 1’in devamı

```
            obj.bits = bitor(bitand(obj.get_return_number,7),
bitand(bitshift(obj.get_number_of_returns,3),7));

            obj.bits = bitor(obj.bits, bitor(bitshift(obj.get_scan_direction_flag,6),
bitshift(obj.get_edge_of_flight_line,7)));

        end

    end

    if isempty(obj.classification)

        obj.read_classification();

        if isempty(obj.classification)

            warning('Adding zeros to classification')

            obj.classification = zeros(LEN,1,'uint8');

        end

    end

    if isempty(obj.scan_angle)

        obj.read_scan_angle();

        if isempty(obj.scan_angle)

            warning('Adding zeros to scan angle')

            obj.scan_angle = zeros(LEN,1,'uint8');

        end

    end

    if isempty(obj.user_data)

        obj.read_user_data();

        if isempty(obj.user_data)

            warning('Adding zeros to user data')

            obj.user_data = zeros(LEN,1,'uint8');

        end
```

Ek 1'in devamı

```
    end
    if isempty(obj.point_source_id)
       obj.read_point_source_id();
       if isempty(obj.point_source_id)
          warning('Adding zeros to point source id')
          obj.point_source_id = zeros(LEN,1,'uint16');
       end
    end
    if ~any(newheader.point_data_format == [0 2])
       if isempty(obj.gps_time)
          obj.read_gps_time();
          if isempty(obj.gps_time)
             warning('Adding zeros to gps time')
             obj.gps_time = zeros(LEN,1,'double');
          end
       end
    end
    if any(newheader.point_data_format == [3 5 7 8 10])
       if isempty(obj.red)
          obj.read_color();
          if isempty(obj.red)
             warning('Adding zeros to RGB color')
             obj.red = zeros(LEN,1,'uint16');
             obj.green = zeros(LEN,1,'uint16');
             obj.blue = zeros(LEN,1,'uint16');
```

Ek 1'in devamı

```
            end
            if any(newheader.point_data_format == [8 10])
               if isempty(obj.nir)
                  warning('Adding zeros to nir color')
                  obj.nir = zeros(LEN,1,'uint16');
               end
            end
         end
      end
      if any(newheader.point_data_format == [4 5 9 10])
         if isempty(obj.wave_return_point)
            obj.read_point_wave_info();
            if isempty(obj.wave_return_point)
               warning('Adding zeros to wave packet info')
               obj.wave_packet_descriptor = zeros(LEN,1,'uint8');
               obj.wave_byte_offset = zeros(LEN,1,'uint64');
               obj.wave_packet_size = zeros(LEN,1,'uint32');
               obj.wave_return_point = zeros(LEN,1,'single');
               obj.Xt = zeros(LEN,1,'single');
               obj.Yt = zeros(LEN,1,'single');
               obj.Zt = zeros(LEN,1,'single');
            end
         end
      end
      obj.header = newheader;
```

Ek 1’in devamı

```
%%% variable length records
try
   obj.write_variable_records(fid);
catch err
   error(['Error writing variable length records: ' err.getReport]);
end
if obj.header.version_major==1 && obj.header.version_minor == 0
   tmp = char([hex2dec('DD') hex2dec('CC')]); %write las 1.0 variable record start
   fwrite(fid,tmp,'uint8');
end
%find offset to point data and write it to header in file
tmppos = ftell(fid);
obj.header.offset_to_point_data = tmppos;
fseek(fid,96,-1);
fwrite(fid,uint32(tmppos),'uint32');
fseek(fid,tmppos,-1);
%calculate point record length and write it to header in file
record_lengths = [20 28 26 34 57 63 30 36 38 59 67 ];
obj.header.point_data_record_length = ...
   record_lengths(obj.header.point_data_format+1) + size(obj.extradata,2);
fseek(fid,105,-1);
fwrite(fid,obj.header.point_data_record_length,'uint16');
%update extendedvariable offset
if newheader.version_minor > 3 %1.4
   obj.header.start_of_extended_variable_length_record = ...
```

Ek 1’in devamı

```
            obj.header.offset_to_point_data +
length(obj.x)*obj.header.point_data_record_length;
        fseek(fid,235,-1);
        fwrite(fid,obj.header.start_of_extended_variable_length_record,'uint64');
    end
    fclose(fid);
    try
        obj.write_xyz();
        obj.write_intensity();
        obj.write_bits();
        obj.write_classification();
        obj.write_scan_angle();
        obj.write_user_data();
        obj.write_point_source_id();
        obj.write_gps_time();
        obj.write_color();
        obj.write_point_wave_info();
        obj.write_extradata();
    catch err
        error(['Error writing point data: ' err.getReport]);
    end
    fid = fopen(filename,'r+');

fseek(fid,double(obj.header.offset_to_point_data+length(obj.x)*obj.header.point_data_rec
ord_length),-1);
    try
        obj.write_extended_variables(fid);
```

Ek 1'in devamı

```
        catch err
            error(['Error writing extended variable data: ' err.getReport]);
        end
        fclose(fid);
        obj.header = oldheader;
    end
    function
add_waveform_packet_desc(obj,bits,compression,numberofsamples,samplespacing,gain,of
fset)
        obj.header.number_of_variable_records = obj.header.number_of_variable_records +
1;
        idx = obj.header.number_of_variable_records;
        obj.variablerecords(idx).reserved = uint16(0);
        tmp = 'LASF_Spec';
        obj.variablerecords(idx).user_id = [tmp zeros(1,16-length(tmp),'uint8')];
        %calculate new descriptor number
        if ~isempty(obj.wavedescriptors)
            tmp =  [obj.variablerecords.record_id];
            tmp(tmp<100 | tmp >355) = []; %remove non wavedescriptors
            val = max(tmp);
            obj.variablerecords(idx).record_id = uint16(val+1);
        else
            obj.variablerecords(idx).record_id = uint16(100); %first id
        end
        obj.variablerecords(idx).record_length = uint16(26);
        tmp = 'Waveform Packet Descriptor';
        obj.variablerecords(idx).description = [tmp zeros(1,32-length(tmp),'uint8')];
```

Ek 1'in devamı

```
        data(1) = uint8(bits);
        data(2) = uint8(compression);
        data(3:6) = typecast(uint32(numberofsamples),'uint8');
        data(7:10) = typecast(uint32(samplespacing),'uint8');
        data(11:18) = typecast(double(gain),'uint8');
        data(19:26) = typecast(double(offset),'uint8');
        obj.variablerecords(idx).data = data;
        obj.wavedescriptors = obj.decode_waveform_packet_desc(obj.variablerecords);
    end
    function
add_waveforms_to_external_file(obj,pointids,descriptorids,waves,xts,yts,zts,returnpoints)
        if size(waves,2)==1
            error('Input wavedatas as rows.')
        end
        if obj.waveformfilefid < 3
            [pathtmp,filetmp]=fileparts(obj.filename);
            if isempty(pathtmp)
                pathtmp = pwd;
            end
            obj.waveformfile = [pathtmp '\\' filetmp '.wdp'];
            obj.waveformfilefid = fopen(obj.waveformfile,'a+');
        end
        fseek(obj.waveformfilefid,0,'eof');
        %set external waveform file to global encoding
        obj.header.global_encoding =
bitor(bitand(uint16(obj.header.global_encoding),uint16(65533)) , bitshift(uint16(1),2));
```

Ek 1'in devamı

```
%load existing data if not loaded, or data is empty
if isempty(obj.Xt) || isempty(obj.Yt) || isempty(obj.Zt) || ...
    isempty(obj.wave_return_point) || isempty(obj.wave_packet_descriptor)...
    || isempty(obj.wave_byte_offset) || isempty(obj.wave_packet_size)
  obj.read_point_wave_info();
end
if size(obj.Xt) ~= size(obj.x); obj.Xt = zeros(size(obj.x),'single'); end
if size(obj.Yt) ~= size(obj.x); obj.Yt = zeros(size(obj.x),'single'); end
if size(obj.Zt) ~= size(obj.x); obj.Zt = zeros(size(obj.x),'single'); end
if size(obj.wave_return_point) ~= size(obj.x); obj.wave_return_point =
zeros(size(obj.x),'single'); end
if size(obj.wave_packet_descriptor) ~= size(obj.x); obj.wave_packet_descriptor =
zeros(size(obj.x),'uint8'); end
if size(obj.wave_byte_offset) ~= size(obj.x); obj.wave_byte_offset =
zeros(size(obj.x),'uint64'); end
if size(obj.wave_packet_size) ~= size(obj.x); obj.wave_packet_size =
zeros(size(obj.x),'uint32'); end
obj.Xt(pointids) = xts;
obj.Yt(pointids) = yts;
obj.Zt(pointids) = zts;
obj.wave_return_point(pointids) = returnpoints;
obj.wave_packet_descriptor(pointids) = descriptorids;
%multiple points with same wave
if size(waves,1)~= length(pointids)
  pos = ftell(obj.waveformfilefid);
  len = size(waves,2);
  obj.wave_byte_offset(pointids) = pos;
```

Ek 1'in devamı

```
            obj.wave_packet_size(pointids) = len;
            fwrite(obj.waveformfilefid,waves,'uint8');
        else
            %insert multiple points with different waves
            for k=1:length(pointids)
                pos = ftell(obj.waveformfilefid);
                len = size(waves,2);
                obj.wave_byte_offset(pointids(k)) = pos;
                obj.wave_packet_size(pointids(k)) = len;
                fwrite(obj.waveformfilefid,waves,'uint8');
            end
        end
    end
end
methods (Access=private)
    function obj = las_read( obj, file )
        if ~exist(file,'file')
            error('File not found.')
        end
        fid = fopen(file,'r');
        obj = readheader(obj,fid);
        obj.isLAZ  = 0;
        %check for LAZ compressed file
        if any(obj.header.point_data_format >=127) %127 laz shift on pointformat?
            obj.isLAZ  = 1;
```

Ek 1'in devamı

```
        fclose(fid);
        if ~exist('laszip.exe','file')
            error('Cannot decompress LAZ file without laszip.exe')
        end
        obj.filename = [file '_tmp.las'];
        system(['laszip -i ' file ' -o ' obj.filename]);
        fid = fopen(obj.filename,'r');
        obj = readheader(obj,fid);
    end
    %set filter off
    obj.selection = true(obj.header.number_of_point_records,1);
    obj = read_variable_records(obj,fid);


    if obj.header.version_major == 1 && obj.header.version_minor == 0
        check = fread(fid,2,'uint8');
        if check(1) ~= hex2dec('DD') && check(2) ~= hex2dec('CC')
            warning('File position do not match offset to the point data, continuing reading
using header position.')
            fseek(fid,double(obj.header.offset_to_point_data),-1);
        end
    end
    if ftell(fid) ~= obj.header.offset_to_point_data
        [ftell(fid) obj.header.offset_to_point_data]
        warning('File position do not match offset to the point data, continuing reading
using header position.')
        fseek(fid,double(obj.header.offset_to_point_data),-1);
```

Ek 1’in devamı

```
        end
        obj.read_xyz();
        obj.extendedvariables = [];
        if isfield(obj.header,'number_of_extended_variable_length_record') ...
                && obj.header.number_of_extended_variable_length_record > 0 ...
                && ~feof(fid)
            fseek(fid,obj.header.start_of_extended_variable_length_record,-1);
            read_extended_variables(obj,fid);
        end
        fclose(fid);
        if any(obj.header.point_data_format == [4 5])
            obj.wavedescriptors = obj.decode_waveform_packet_desc(obj.variablerecords);
            obj.setwaveformsource();
        end
    end
    function obj = createemptyheader(obj)
            obj.header.source_id = [];
            obj.header.global_encoding = [];
            obj.header.project_id_guid1 = [];
            obj.header.project_id_guid2 = [];
            obj.header.project_id_guid3 = [];
            obj.header.project_id_guid4 = [];
            obj.header.version_major = [];
            obj.header.version_minor = [];
            obj.header.system_identifier = [];
```

Ek 1'in devamı

```
obj.header.generating_software = [];
obj.header.file_creation_daobj.y = [];
obj.header.file_creation_year = [];
obj.header.header_size = [];
obj.header.offset_to_point_data = [];
obj.header.number_of_variable_records = [];
obj.header.point_data_format = [];
obj.header.point_data_record_length = [];
obj.header.number_of_point_records = [];
obj.header.number_of_points_by_return = [];
obj.header.scale_factor_x = [];
obj.header.scale_factor_y = [];
obj.header.scale_factor_z  = [];
obj.header.x_offset = [];
obj.header.y_offset = [];
obj.header.z_offset = [];
obj.header.max_x = [];
obj.header.min_x = [];
obj.header.max_y = [];
obj.header.min_y = [];
obj.header.max_z  = [];
obj.header.min_z  = [];
obj.header.start_of_extended_variable_length_record = [];
obj.header.number_of_extended_variable_length_record = [];
obj.header.start_of_waveform_data = [];
```

Ek 1'in devamı

```
    end
    function obj = readheader(obj,fid)
        str = fscanf(fid,'%c',4);
        if strcmp('LASF',str)==0
            fclose(fid);
            error([file ' is not a LAS file.'])
        end
        try
            obj.header.source_id = fread(fid,1,'uint16');
            obj.header.global_encoding = fread(fid,1,'uint16');
            obj.header.project_id_guid1 = fread(fid,1,'uint32');
            obj.header.project_id_guid2 = fread(fid,1,'uint16');
            obj.header.project_id_guid3 = fread(fid,1,'uint16');
            obj.header.project_id_guid4 = fread(fid,8,'int8');
            obj.header.version_major = fread(fid,1,'uint8');
            obj.header.version_minor = fread(fid,1,'uint8');
            obj.header.system_identifier = fscanf(fid,'%c',32);
            obj.header.generating_software = fscanf(fid,'%c',32);
            obj.header.file_creation_daobj.y = fread(fid,1,'uint16');
            obj.header.file_creation_year = fread(fid,1,'uint16');
            obj.header.header_size = fread(fid,1,'uint16');
            obj.header.offset_to_point_data = fread(fid,1,'uint32');
            obj.header.number_of_variable_records = fread(fid,1,'uint32');
            obj.header.point_data_format = fread(fid,1,'uint8');
            obj.header.point_data_record_length = fread(fid,1,'uint16');
```

Ek 1'in devamı

```
        obj.header.number_of_point_records = fread(fid,1,'uint32');
        obj.header.number_of_points_by_return = fread(fid,5,'uint32');
        obj.header.scale_factor_x = fread(fid,1,'double');
        obj.header.scale_factor_y = fread(fid,1,'double');
        obj.header.scale_factor_z  = fread(fid,1,'double');
        obj.header.x_offset = fread(fid,1,'double');
        obj.header.y_offset = fread(fid,1,'double');
        obj.header.z_offset = fread(fid,1,'double');
        obj.header.max_x = fread(fid,1,'double');
        obj.header.min_x = fread(fid,1,'double');
        obj.header.max_y = fread(fid,1,'double');
        obj.header.min_y = fread(fid,1,'double');
        obj.header.max_z  = fread(fid,1,'double');
        obj.header.min_z  = fread(fid,1,'double');
        if obj.header.version_minor > 2 %1.3
           obj.header.start_of_waveform_data = fread(fid,1,'uint64');
        end
         % add one EVLR for 1.3, if point format 4 or 5 and internal waveforms
        if obj.header.version_major == 1 && obj.header.version_minor == 3 ...
              && bitand(obj.header.global_encoding,2) &&
~bitand(obj.header.global_encoding,4)
           obj.header.number_of_extended_variable_length_record = 1;
        end
        if obj.header.version_minor > 3 %1.4
           obj.header.start_of_extended_variable_length_record = fread(fid,1,'uint64');
           obj.header.number_of_extended_variable_length_record = fread(fid,1,'uint32');
```

Ek 1’in devamı

```
            %copy legacy values to show
            obj.header.legacy_number_of_point_records_READ_ONLY =
obj.header.number_of_point_records;
            obj.header.number_of_point_records = fread(fid,1,'uint64');
            obj.header.legacy_number_of_points_by_return_READ_ONLY =
obj.header.number_of_points_by_return;
            obj.header.number_of_points_by_return = fread(fid,15,'uint64');
        end
    catch ex
        fclose(fid);
        disp('Error while processing header information.')
        throw(ex)
    end
    if obj.header.version_major > 1 || obj.header.version_minor > 4
        warning('Trying to parse unsupported LAS version.');
    end
end
function obj = writeheader(obj,fid)
    fseek(fid,0,-1);
    fprintf(fid,'LASF');
    fwrite(fid, obj.header.source_id,'uint16');
    fwrite(fid, obj.header.global_encoding,'uint16');
    fwrite(fid, obj.header.project_id_guid1,'uint32');
    fwrite(fid, obj.header.project_id_guid2,'uint16');
    fwrite(fid, obj.header.project_id_guid3,'uint16');
    fwrite(fid, obj.header.project_id_guid4,'int8');
```

Ek 1’in devamı

```
fwrite(fid, obj.header.version_major,'uint8');
fwrite(fid, obj.header.version_minor,'uint8');
tmp = obj.header.system_identifier;
tmp = [tmp zeros(1,32-length(tmp),'uint8')];
fprintf(fid, '%c',tmp);
tmp = obj.header.generating_software;
tmp = [tmp zeros(1,32-length(tmp),'uint8')];
fprintf(fid, '%c', tmp);
fwrite(fid, obj.header.file_creation_daobj.y,'uint16');
fwrite(fid, obj.header.file_creation_year,'uint16');
fwrite(fid, obj.header.header_size,'uint16');
fwrite(fid, obj.header.offset_to_point_data,'uint32');
fwrite(fid, obj.header.number_of_variable_records,'uint32');
fwrite(fid, obj.header.point_data_format,'uint8');
fwrite(fid, obj.header.point_data_record_length,'uint16');
if obj.header.number_of_point_records < 2^32 %if legacy compatible
   fwrite(fid, obj.header.number_of_point_records,'uint32');
else
   fwrite(fid, 0,'uint32');
end
%add lecagy only if possible by pointcount limited by uint32
if obj.header.number_of_point_records < 2^32 && ...
       (length(obj.header.number_of_points_by_return) == 15 && ...
        all(obj.header.number_of_points_by_return(6:15)==0))
   tmpp = obj.header.number_of_points_by_return;
```

Ek 1’in devamı

```
    if length(obj.header.number_of_points_by_return)==15
        tmpp = tmpp(1:5);
    end
    fwrite(fid, tmpp,'uint32');
else
    fwrite(fid, zeros(5,1,'uint32'),'uint32');
end
fwrite(fid, obj.header.scale_factor_x,'double');
fwrite(fid, obj.header.scale_factor_y,'double');
fwrite(fid, obj.header.scale_factor_z,'double');
fwrite(fid, obj.header.x_offset,'double');
fwrite(fid, obj.header.y_offset,'double');
fwrite(fid, obj.header.z_offset,'double');
fwrite(fid, obj.header.max_x,'double');
fwrite(fid, obj.header.min_x,'double');
fwrite(fid, obj.header.max_y,'double');
fwrite(fid, obj.header.min_y,'double');
fwrite(fid, obj.header.max_z,'double');
fwrite(fid, obj.header.min_z,'double');
if obj.header.version_minor > 2 %1.3
    if ~isfield(obj.header,'start_of_waveform_data')
        fwrite(fid, 0,'uint64');
    else
        fwrite(fid, obj.header.start_of_waveform_data,'uint64');
    end
```

Ek 1'in devamı

```
    end
    if obj.header.version_minor > 3 %1.4
       if ~isfield(obj.header,'start_of_extended_variable_length_record')
          fwrite(fid, 0,'uint64');
       else
          fwrite(fid, obj.header.start_of_extended_variable_length_record,'uint64');
       end
       if ~isfield(obj.header,'number_of_extended_variable_length_record')
          fwrite(fid, 0,'uint32');
       else
          fwrite(fid, obj.header.number_of_extended_variable_length_record,'uint32');
       end
       if ~isfield(obj.header,'number_of_point_records')
          fwrite(fid, 0,'uint64');
       else
          fwrite(fid, obj.header.number_of_point_records,'uint64');
       end
       if ~isfield(obj.header,'number_of_points_by_return')
          fwrite(fid, 15,'uint64');
       else
          fwrite(fid, obj.header.number_of_points_by_return,'uint64');
       end
    end
    %write header length
    pos = ftell(fid);
```

Ek 1'in devamı

```
        fseek(fid,94,-1);
        fwrite(fid,pos,'uint16');
        fseek(fid,pos,-1);
    end
    function obj = read_variable_records(obj,fid)
        for k=1:obj.header.number_of_variable_records
            obj.variablerecords(k).reserved = fread(fid,1,'*uint16');
            obj.variablerecords(k).user_id = fscanf(fid,'%c',16);
            obj.variablerecords(k).record_id = fread(fid,1,'*uint16');
            obj.variablerecords(k).record_length = fread(fid,1,'*uint16');
            obj.variablerecords(k).description = fscanf(fid,'%c',32);
            obj.variablerecords(k).data =
fread(fid,obj.variablerecords(k).record_length,'*uint8');
            obj.variablerecords(k).data_as_text = char(obj.variablerecords(k).data(:))';
        end
    end
    function obj = write_variable_records(obj,fid)
        for k=1:obj.header.number_of_variable_records
            if obj.header.version_major==1 && obj.header.version_minor == 0
                tmp = char([hex2dec('BB') hex2dec('AA')]); %write las 1.0 variable record start
                fwrite(fid,tmp,'uint8');
            end
            fwrite(fid,obj.variablerecords(k).reserved,'uint16');
            tmp = obj.variablerecords(k).user_id;
            tmp = [tmp zeros(1,16-length(tmp))];
            fprintf(fid,'%c',tmp);
```

Ek 1’in devamı

```
            fwrite(fid,obj.variablerecords(k).record_id,'uint16');
            fwrite(fid,length(obj.variablerecords(k).data),'uint16');
            tmp = obj.variablerecords(k).description;
            tmp = [tmp zeros(1,32-length(tmp))];
            fprintf(fid,'%c',tmp);
            fwrite(fid,obj.variablerecords(k).data,'uint8');
        end
    end
    function obj = read_extended_variables(obj,fid)
        for k=1:obj.header.number_of_extended_variable_length_record
            obj.extendedvariables(k).reserved = fread(fid,1,'*uint16');
            obj.extendedvariables(k).user_id = fread(fid,16,'*int8');
            obj.extendedvariables(k).record_id = fread(fid,1,'*uint16');
            obj.extendedvariables(k).record_length_after_obj_header = fread(fid,1,'*uint64');
            obj.extendedvariables(k).description = fread(fid,32,'*int8');
            if isfield(obj.header,'start_of_waveform_data') && ftell(fid) ==
obj.header.start_of_waveform_data
                warning('Skipping waveform data while reading extended variables. Access
waveforms with waveform reading function.');
                obj.extendedvariables(k).data = [];
            else
                obj.extendedvariables(k).data =
fread(fid,obj.extendedvariables.record_length_after_obj_header,'*uint8');
            end
        end
    end
    function obj = write_extended_variables(obj,fid)
```

Ek 1’in devamı

```
    if ~isfield(obj.header,'number_of_extended_variable_length_record')
       obj.header.number_of_extended_variable_length_record = 0;
    end
    for k=1:obj.header.number_of_extended_variable_length_record
        fwrite(fid,obj.extendedvariables(k).reserved,'uint16');
        fwrite(fid,obj.extendedvariables(k).user_id,'int8');
        fwrite(fid,obj.extendedvariables(k).record_id,'uint16');
        fwrite(fid,length(obj.extendedvariables(k).data),'uint64');
        fwrite(fid,obj.extendedvariables(k).description,'int8');
        fwrite(fid,obj.extendedvariables(k).data,'uint8');
    end
  end
  function obj = setwaveformsource(obj)
    %check if external waveform file
    if bitand(obj.header.global_encoding,4)
       [pathtmp,filetmp]=fileparts(obj.originalname);
       if isempty(pathtmp)
          pathtmp = pwd;
       end
       obj.waveformfile = [pathtmp '\\' filetmp '.wdp'];
       if ~exist(obj.waveformfile,'file')
          warning(['External waveform file ' obj.waveformfile ' not found!'])
          return;
       end
    else
```

Ek 1'in devamı

```
            obj.waveformfile = obj.filename;
        end
    end
end
methods(Static,Access=public)
    %line to triangle converter for plotting
    function arr = FlattenTriangleArrayComponent(arr)
        arr = padarray(arr',2,'post');
        d = arr(:);
        arr = d;
        arr = arr + [0; d(1:end-1)];
        arr = arr(1:end-3) + [0; 0; d(4:end-2)];
    end
    function [codes] = list_classification_codes(format,noprint)
        if format==1.0 || format==1.1 || format==1.2 || format==1.3
            codes={ 'Created, never classified',...
                'Unclassified','Ground', 'Low Vegetation',...
                'Medium vegetation', 'High vegetation', 'Building',...
                'Low point (noise)','Model key-point (mass point)',...
                'Water','Reserved', 'Reserved', 'Overlap Points'};
            if ~exist('noprint','var')
                fprintf('\nPoint Record Types 0-5\n');
                for k=1:length(codes)
                    fprintf('%d  %s\n',k-1,codes{k})
                end
```

Ek 1'in devamı

```
            fprintf('13-31 Reserved\n\n')
        end
        for k=13:31; codes{k} = 'Reserved'; end
    else
        codes = {'Created, never classified',...
            'Unclassified','Ground', 'Low Vegetation',...
            'Medium vegetation', 'High vegetation', 'Building',...
            'Low point (noise)','Reserved','Water',...
            'Rail', 'Road surface', 'Reserved',...
            'Wire - guard (Shield)','Wire - conductor (Phase)',...
            'Transmission tower','Wire-structure connector (insulator)',...
            'Bridge deck','High noise'};
        if ~exist('noprint','var')
            fprintf('\nPoint record types 6-10\n');
            for k=1:length(codes)
                fprintf('%d  %s\n',k-1,codes{k})
            end
            fprintf('19-63  Reserved\n')
            fprintf('64-255  User definable\n\n')
        end
        for k=13:31; codes{k} = 'Reserved'; end
        for k=64:255; codes{k} = 'User definable'; end
    end
end
%helper class not operating on class data
```

Ek 1'in devamı

```
function desc = decode_waveform_packet_desc(data)
    r=1;
    for k=1:length(data)
        if data(k).record_id >=100 && data(k).record_id < 356 %waveform record types
            desc(r).bits = typecast(data(k).data(1),'uint8');
            desc(r).compression = typecast(data(k).data(2),'uint8');
            desc(r).number_of_samples = typecast(data(k).data(3:6),'uint32');
            desc(r).temporal_sample_spacing = typecast(data(k).data(7:10),'uint32');
            desc(r).digitizer_gain = typecast(data(k).data(11:18),'double');
            desc(r).digitizer_offset = typecast(data(k).data(19:26),'double');
            r=r+1;
        end
    end
    if ~exist('desc','var')
        warning('Expecting waveform descriptors, but found none.')
        desc = [];
    end
end
function columndatafwrite(fid,data,columnpos,rowlength)
    %fwrite with block read/write to have faster column writes
    BLOCKROWS = 20000;
    insertdatalen = length(typecast(data(1,:),'uint8'));
    for k=1:BLOCKROWS:size(data,1)
        pos = ftell(fid);
        bend = k+BLOCKROWS-1;
```

Ek 1’in devamı

```
if bend > size(data,1)
   bend = size(data,1);
end
%find file size
fseek(fid,0,1);
filelen = ftell(fid);
fseek(fid,pos,-1);
%create empty space, because reading will fail otherwise
%next column write will be faster
if filelen < ftell(fid)+(bend-k+1)
   need_to_allocate = (bend-k+1)*rowlength - (filelen-pos);
   fwrite(fid,zeros(need_to_allocate,1,'uint8'));
end
fseek(fid,pos,-1);
%read block to memory
block = fread(fid,(bend-k+1)*rowlength,'*uint8');
block = reshape(block,rowlength,[])';
%add data in memory
tmp = data(k:bend,:)';
tmp = typecast(tmp(:),'uint8');
tmp = reshape(tmp,insertdatalen,[])';
block(:,columnpos+1:columnpos+insertdatalen) = tmp;
block = block';
%write block back to file
fseek(fid,pos,-1);
```

Ek 1’in devamı

```
        fwrite(fid,block(:));
    end
```

Ek.2 Optech-1200-38.las verisi ön işlem kodları

```
%% Verilerin okunması
[FileName,PathName] = uigetfile('*.las','C:\Users\MSI-011\Desktop\optech_1200');
A=lasdata('optech_1200_38.las','loadall');
B=[ A.x A.y A.z ];
figure;
pcshow(B);
%% intensity değerlerini double hale getirme
C=[ A.intensity ];
D=im2double(C);
E=[ A.x A.y A.z D ];
%% intensity değeri 40 ve üzeri olanları bulma (yansıma değeri filtrelemesi)
im40=double0.0006103608758678569
int=find (E(:,4)>=0.0006103608758678569);
F=E(int,:);
G=[F(:,1), F(:,2), F(:,3)];
pcshow(G);
%% ginput uygulama
plot(G(:,1),G(:,2),'.r','MarkerSize', 0.1);
```

Ek 2’in devamı

```
pause(30);

button = 1;

i = 0;

while button == 1;

   i=i+1;

[x(i),y(i),button] = ginput(1);% sağ tuş ile bitir..

End

% x=x(1:i-1);

% y=y(1:i-1);

koor=[x(:) y(:)];

clear x;clear y;

xmin=koor(1,1);

xmax=koor(2,1);

ymin=koor(1,2);

ymax=koor(2,2);

x=find (G(:,1)<xmax & G(:,1)>xmin);

y=find (G(:,2)<ymax & G(:,2)>ymin);

ind=intersect(x,y);

K_38=G(ind,:);

plot(K_38(:,1),K_38(:,2),'.r','MarkerSize',0.1)

%% pcdenoise uygulaması (gürültü filtreleme)
```

Ek 2’in devamı

```
figure;
pcshow(K_38);
title('Noisy Data');
ptCloud=pointCloud(K_38);
ptCloud38 = pcdenoise(ptCloud,'NumNeighbors',20,'Threshold',0.001);
figure;
pcshow(ptCloud38);
title('Denoised Data');
%% downsample uygulaması (çakışan veri filtrelemesi)
gridStep = 0.01;
ptCloudA = pcdownsample(ptCloud38,'gridAverage',gridStep);
figure;
pcshow(ptCloudA);
%% Mevcut Kooridnatların Gösterimi
X=ptCloudA.Location(:,1);
Y=ptCloudA.Location(:,2);
Z=ptCloudA.Location(:,3);
K38=[ X, Y, Z ];
%% Su Kotu ve Yakın Değerleri Bulma
M=min(K38(:,3));
D=M+0.20;
```

Ek 2’in devamı

```
z=find (K38(:,3)>M & K38(:,3)<D);

KS38=K38(z,:);

figure;

pcshow(KS38);

%% sonuçlar

save ('sonucKS38.txt','KS38','-ascii');

type('sonucKS38.txt');

figure;

plot(KS38(:,1),KS38(:,2),'.b','MarkerSize', 0.1);
```

Ek.3 Optech-1200-53.las verisi ön işlem kodları

```
%% Verilerin okunması

[FileName,PathName] = uigetfile('*.las','C:\Users\MSI-011\Desktop\optech_1200');

A=lasdata('optech_1200_53.las','loadall');

B=[ A.x A.y A.z ];

figure;

pcshow(B);

%% intensity değerlerini double hale getirme

C=[ A.intensity ];

D=im2double(C);

E=[ A.x A.y A.z D ];
```

Ek 3’ün devamı

```
%% intensity değeri 40 ve üzeri olanları bulma (yansıma değeri filtrelemesi)
im40=double0.0006103608758678569

int=find (E(:,4)>=0.0006103608758678569);

F=E(int,:);

G=[F(:,1), F(:,2), F(:,3)];

pcshow(G);

%% ginput uygulama

plot(G(:,1),G(:,2),'.r','MarkerSize', 0.1);

pause(30);

button = 1;

i = 0;

while button == 1;

   i=i+1;

[x(i),y(i),button] = ginput(1);% sağ tuş ile bitir..

end

% x=x(1:i-1);

% y=y(1:i-1);

koor=[x(:) y(:)];

clear x;clear y;

xmin=koor(1,1);

xmax=koor(2,1);

ymin=koor(1,2);
```

Ek 3’ün devamı

```
ymax=koor(2,2);
x=find (G(:,1)<xmax & G(:,1)>xmin);
y=find (G(:,2)<ymax & G(:,2)>ymin);
ind=intersect(x,y);
K_53=G(ind,:);
plot(K_53(:,1),K_53(:,2),'.r','MarkerSize',0.1)
%% pcdenoise uygulaması (gürültü filtreleme)
figure;
pcshow(K_53);
title('Noisy Data');
ptCloud=pointCloud(K_53);
ptCloud53 = pcdenoise(ptCloud,'NumNeighbors',20,'Threshold',0.001);
figure;
pcshow(ptCloud53);
title('Denoised Data');
%% downsample uygulaması (çakışan veri filtrelemesi)
gridStep = 0.01;
ptCloudA = pcdownsample(ptCloud53,'gridAverage',gridStep);
figure;
pcshow(ptCloudA);
%% Mevcut Kooridnatların Gösterimi
```

Ek 3’ün devamı

```
X=ptCloudA.Location(:,1);

Y=ptCloudA.Location(:,2);

Z=ptCloudA.Location(:,3);

K53=[ X, Y, Z ];

%% Su Kotu ve Yakın Değerleri Bulma

M=min(K53(:,3));

D=M+0.30;

z=find (K53(:,3)>M & K53(:,3)<D);

KS53=K53(z,:);

figure;

pcshow(KS53);

%% sonuçlar

save ('sonucKS53.txt','KS53','-ascii');

type('sonucKS53.txt');

figure;

plot(KS53(:,1),KS53(:,2),'.b','MarkerSize', 0.1);
```

Ek.4 Optech-1200-54.las verisi ön işlem kodları

```
%% Verilerin okunması

[FileName,PathName] = uigetfile('*.las','C:\Users\MSI-011\Desktop\optech_1200');

A=lasdata('optech_1200_54.las','loadall');
```

Ek 4’ün devamı

```
B=[ A.x A.y A.z ];

figure;

pcshow(B);

%% intensity değerlerini double hale getirme

C=[ A.intensity ];

D=im2double(C);

E=[ A.x A.y A.z D ];

%% intensity değeri 40 ve üzeri olanları bulma (yansıma değeri filtrelemesi)
im40=double0.0006103608758678569

int=find (E(:,4)>=0.0006103608758678569);

F=E(int,:);

G=[F(:,1), F(:,2), F(:,3)];

pcshow(G);

%% ginput uygulama

plot(G(:,1),G(:,2),'.r','MarkerSize', 0.1);

pause(30);

button = 1;

i = 0;

while button == 1;

   i=i+1;

[x(i),y(i),button] = ginput(1);% sağ tuş ile bitir..

end
```

Ek 4’ün devamı

```
% x=x(1:i-1);

% y=y(1:i-1);

koor=[x(:) y(:)];

clear x;clear y;

xmin=koor(1,1);

xmax=koor(2,1);

ymin=koor(1,2);

ymax=koor(2,2);

x=find (G(:,1)<xmax & G(:,1)>xmin);
```

Ek 4’ün devamı

```
y=find (G(:,2)<ymax & G(:,2)>ymin);

ind=intersect(x,y);

K_54_1=G(ind,:);

plot(K_54_1(:,1),K_54_1(:,2),'.r','MarkerSize',0.1)

%% pcdenoise uygulaması (gürültü filtreleme)

figure;

pcshow(K_54_1);

title('Noisy Data');

ptCloud=pointCloud(K_54_1);

ptCloud54_1 = pcdenoise(ptCloud,'NumNeighbors',20,'Threshold',0.001);

figure;
```

Ek 4’ün devamı

```
pcshow(ptCloud54_1);
title('Denoised Data');
%% downsample uygulaması (çakışan veri filtrelemesi)
gridStep = 0.01;
ptCloudA = pcdownsample(ptCloud54_1,'gridAverage',gridStep);
figure;
pcshow(ptCloudA);
%% Mevcut Kooridnatların Gösterimi
X=ptCloudA.Location(:,1);
Y=ptCloudA.Location(:,2);
Z=ptCloudA.Location(:,3);
K54_1=[ X, Y, Z ];
%% Su Kotu ve Yakın Değerleri Bulma
M=min(K54_1(:,3));
```

Ek 4’ün devamı

```
D=M+0.20;
z=find (K54_1(:,3)>M & K54_1(:,3)<D);
KS54_1=K54_1(z,:);
figure;
pcshow(KS54_1);
%% sonuçlar
```

Ek 4’ün devamı

```
save ('sonucKS54_1.txt','KS54_1','-ascii');

type('sonucKS54_1.txt');

figure;

plot(KS54_1(:,1),KS54_1(:,2),'.b','MarkerSize', 0.1);
```

Ek.5 Optech-1200-55.las verisi ön işlem kodları

```
%% Verilerin okunması

[FileName,PathName] = uigetfile('*.las','C:\Users\MSI-011\Desktop\optech_1200');

A=lasdata('optech_1200_55.las','loadall');

B=[ A.x A.y A.z ];

figure;

pcshow(B);

%% intensity değerlerini double hale getirme

C=[ A.intensity ];

D=im2double(C);

E=[ A.x A.y A.z D ];

%% intensity değeri 40 ve üzeri olanları bulma (yansıma değeri filtrelemesi)
im40=double0.0006103608758678569

int=find (E(:,4)>=0.0006103608758678569);

F=E(int,:);

G=[F(:,1), F(:,2), F(:,3)];

pcshow(G);
```

Ek 5’in devamı

```
%% ginput uygulama
plot(G(:,1),G(:,2),'.r','MarkerSize', 0.1);
pause(30);
button = 1;
i = 0;
while button == 1;
   i=i+1;
[x(i),y(i),button] = ginput(1);% sağ tuş ile bitir..
end
% x=x(1:i-1);
% y=y(1:i-1);
koor=[x(:) y(:)];
clear x;clear y;
xmin=koor(1,1);
xmax=koor(2,1);
ymin=koor(1,2);
ymax=koor(2,2);
x=find (G(:,1)<xmax & G(:,1)>xmin);
y=find (G(:,2)<ymax & G(:,2)>ymin);
ind=intersect(x,y);
K_55=G(ind,:);
```

Ek 5’in devamı

```
plot(K_55(:,1),K_55(:,2),'.r','MarkerSize',0.1)
%% pcdenoise uygulaması (gürültü filtreleme)
figure;
pcshow(K_55);
title('Noisy Data');
ptCloud=pointCloud(K_55);
ptCloud55 = pcdenoise(ptCloud,'NumNeighbors',20,'Threshold',0.001);
figure;
pcshow(ptCloud55);
title('Denoised Data');
%% downsample uygulaması (çakışan veri filtrelemesi)
gridStep = 0.01;
ptCloudA = pcdownsample(ptCloud55,'gridAverage',gridStep);
figure;
pcshow(ptCloudA);
%% Mevcut Kooridnatların Gösterimi
X=ptCloudA.Location(:,1);
Y=ptCloudA.Location(:,2);
Z=ptCloudA.Location(:,3);
K55=[ X, Y, Z ];
%% Su Kotu ve Yakın Değerleri Bulma
```

Ek 5’in devamı

```
M=min(K55(:,3));
D=M+0.20;
z=find (K55(:,3)>M & K55(:,3)<D);
KS55=K55(z,:);
figure;
pcshow(KS55);
%% sonuçlar
save ('sonucKS55.txt','KS55','-ascii');
type('sonucKS55.txt');
figure;
plot(KS55(:,1),KS55(:,2),'.b','MarkerSize', 0.1);
```

Ek.6 Optech-1200-70.las verisi ön işlem kodları

```
%% Verilerin okunması
[FileName,PathName] = uigetfile('*.las','C:\Users\MSI-011\Desktop\optech_1200');
A=lasdata('optech_1200_70.las','loadall');
B=[ A.x A.y A.z ];
figure;
pcshow(B);
%% intensity değerlerini double hale getirme
C=[ A.intensity ];
```

Ek 6’nın devamı

```
D=im2double(C);

E=[ A.x A.y A.z D ];

%% intensity değeri 40 ve üzeri olanları bulma (yansıma değeri filtrelemesi)
im40=double0.0006103608758678569

int=find (E(:,4)>=0.0006103608758678569);

F=E(int,:);

G=[F(:,1), F(:,2), F(:,3)];

pcshow(G);

%% ginput uygulama

plot(G(:,1),G(:,2),'.r','MarkerSize', 0.1);

pause(30);

button = 1;

i = 0;

while button == 1;

   i=i+1;

[x(i),y(i),button] = ginput(1);% sağ tuş ile bitir..

end

% x=x(1:i-1);

% y=y(1:i-1);

koor=[x(:) y(:)];

clear x;clear y;

xmin=koor(1,1);
```

Ek 6’nın devamı

```
xmax=koor(2,1);

ymin=koor(1,2);

ymax=koor(2,2);

x=find (G(:,1)<xmax & G(:,1)>xmin);

y=find (G(:,2)<ymax & G(:,2)>ymin);

ind=intersect(x,y);

K_70=G(ind,:);

plot(K_70(:,1),K_70(:,2),'.r','MarkerSize',0.1)

%% pcdenoise uygulaması (gürültü filtreleme)

figure;

pcshow(K_70);

title('Noisy Data');

ptCloud=pointCloud(K_70);

ptCloud70 = pcdenoise(ptCloud,'NumNeighbors',20,'Threshold',0.001);

figure;

pcshow(ptCloud70);

title('Denoised Data');

%% downsample uygulaması (çakışan veri filtrelemesi)

gridStep = 0.01;

ptCloudA = pcdownsample(ptCloud70,'gridAverage',gridStep);

figure;
```

Ek 6’nın devamı

```
pcshow(ptCloudA);
%% Mevcut Kooridnatların Gösterimi
X=ptCloudA.Location(:,1);
Y=ptCloudA.Location(:,2);
Z=ptCloudA.Location(:,3);
K70=[ X, Y, Z ];
%% Su Kotu ve Yakın Değerleri Bulma
M=min(K70(:,3));
D=M+0.30;
z=find (K70(:,3)>M & K70(:,3)<D);
KS70=K70(z,:);
figure;
pcshow(KS70);
%% sonuçlar
save ('sonucks70.txt','KS70','-ascii');
type('sonucks70.txt');
figure;
plot(KS70(:,1),KS70(:,2),'.b','MarkerSize', 0.1);
```

Ek.7 Optech-1200-71.las verisi ön işlem kodları

```
%% Verilerin okunması

[FileName,PathName] = uigetfile('*.las','C:\Users\MSI-011\Desktop\optech_1200');

A=lasdata('optech_1200_71.las','loadall');

B=[ A.x A.y A.z ];

figure;

pcshow(B);

%% intensity değerlerini double hale getirme

C=[ A.intensity ];

D=im2double(C);

E=[ A.x A.y A.z D ];

%% intensity değeri 40 ve üzeri olanları bulma (yansıma değeri filtrelemesi)
im40=double0.0006103608758678569

int=find (E(:,4)>=0.0006103608758678569);

F=E(int,:);

G=[F(:,1), F(:,2), F(:,3)];

pcshow(G);

%% ginput uygulama

plot(G(:,1),G(:,2),'.r','MarkerSize', 0.1);

pause(30);

button = 1;

i = 0;
```

Ek 7’nin devamı

```
while button == 1;
   i=i+1;
[x(i),y(i),button] = ginput(1);% sağ tuş ile bitir..
end
% x=x(1:i-1);
% y=y(1:i-1);
koor=[x(:) y(:)];
clear x;clear y;
xmin=koor(1,1);
xmax=koor(2,1);
ymin=koor(1,2);
ymax=koor(2,2);
x=find (G(:,1)<xmax & G(:,1)>xmin);
y=find (G(:,2)<ymax & G(:,2)>ymin);
ind=intersect(x,y);
K_71=G(ind,:);
plot(K_71(:,1),K_71(:,2),'.r','MarkerSize',0.1)
%% pcdenoise uygulaması (gürültü filtreleme)
figure;
pcshow(K_71);
title('Noisy Data');
```

Ek 7’nin devamı

```
ptCloud=pointCloud(K_71);
ptCloud71 = pcdenoise(ptCloud,'NumNeighbors',20,'Threshold',0.001);
figure;
pcshow(ptCloud71);
title('Denoised Data');
%% downsample uygulaması (çakışan veri filtrelemesi)
gridStep = 0.01;
ptCloudA = pcdownsample(ptCloud71,'gridAverage',gridStep);
figure;
pcshow(ptCloudA);
%% Mevcut Kooridnatların Gösterimi
X=ptCloudA.Location(:,1);
Y=ptCloudA.Location(:,2);
Z=ptCloudA.Location(:,3);
K71=[ X, Y, Z ];
%% Su Kotu ve Yakın Değerleri Bulma
M=min(K71(:,3));
D=M+0.20;
z=find (K71(:,3)>M & K71(:,3)<D);
KS71=K71(z,:);
figure;
```

Ek 7’nin devamı

```
pcshow(KS71);
%% sonuçlar
save ('sonucks71.txt','KS71','-ascii');
type('sonucks71.txt');
figure;
plot(KS71(:,1),KS71(:,2),'.b','MarkerSize', 0.1);
```

Ek.8 Sonuçların Birleştirilmesi ve Sınır Çizimi Kodları

```
%% Verilerin okunması
[FileName,PathName] = uigetfile('*.las','C:\Users\MSI-011\Desktop\optech_1200');
%S=lasread(FileName);
%A=[S.X S.Y S.Z];
C= lasdata('sonucKS38.las');
KS38=[C.x C.y C.z];
% Verilerin okunması
[FileName,PathName] = uigetfile('*.las','C:\Users\MSI-011\Desktop\optech_1200');
%S=lasread(FileName);
%A=[S.X S.Y S.Z];
C= lasdata('sonucKS53.las');
KS53=[C.x C.y C.z];
```

Ek 8’in devamı

```
% Verilerin okunması

[FileName,PathName] = uigetfile('*.las','C:\Users\MSI-011\Desktop\optech_1200');

%S=lasread(FileName);

%A=[S.X S.Y S.Z];

C= lasdata('sonucKS54_1.las');

KS54_1=[C.x C.y C.z];

% Verilerin okunması

[FileName,PathName] = uigetfile('*.las','C:\Users\MSI-011\Desktop\optech_1200');

%S=lasread(FileName);

%A=[S.X S.Y S.Z];

C= lasdata('sonucKS54_2.las');

KS54_2=[C.x C.y C.z];

% Verilerin okunması

 [FileName,PathName] = uigetfile('*.las','C:\Users\MSI-011\Desktop\optech_1200');

%S=lasread(FileName);

%A=[S.X S.Y S.Z];

C= lasdata('sonucKS55.las');

KS55=[C.x C.y C.z];

% Verilerin okunması

[FileName,PathName] = uigetfile('*.las','C:\Users\MSI-011\Desktop\optech_1200');

%S=lasread(FileName);
```

Ek 8’in devamı

```
%A=[S.X S.Y S.Z];

C= lasdata('sonucKS70.las');

KS70=[C.x C.y C.z];

% Verilerin okunması

[FileName,PathName] = uigetfile('*.las','C:\Users\MSI-011\Desktop\optech_1200');

%S=lasread(FileName);

%A=[S.X S.Y S.Z];

C= lasdata('sonucKS71.las');

KS71=[C.x C.y C.z];

%% Verilerin Birleştirilmesi

K=[KS38; KS53; KS54_1; KS54_2; KS55; KS70; KS71];

ptCloud=pointCloud(K);

pcshow(ptCloud);

x=ptCloud.Location(:,1);

y=ptCloud.Location(:,2);

z=ptCloud.Location(:,3);

plot3(x,y,z,'g.')

%% pcfitplane Uygulamsı

maxDistance = 0.20;  % Noktalar arası olması gereken maksimum mesafe.

referenceVector = [0,0,1]; % Z yönünde kısıtlama vektörü

maxAngularDistance = 5; % Maksimumu açısal mesafe.
```

Ek 8’in devamı

```
[model,inlierIndices,outlierIndices] =
pcfitplane(ptCloud,maxDistance,referenceVector,maxAngularDistance);

% Belirtilen şartları sağlayan noktaları temsil edenler inlierIndices,

% temsil etmeyenler outlierIndices olarak tanımlanır.

plane = select(ptCloud,inlierIndices);

remainPtCloud = select(ptCloud,outlierIndices);

pcshow(plane)

%pcshow(remainPtCloud)

%% Kestel Baraj Sınırını Oluşturan Noktalar

x1=plane.Location(:,1);

y1=plane.Location(:,2);

z1=plane.Location(:,3);

plot3(x1,y1,z1,'.','markersize',5)

%% Alpha Shape

xy = [x1,y1];

xy = unique(xy,'rows');

shp1 = alphaShape(xy,5,'HoleThreshold',5000000,'RegionThreshold',1);

pc = criticalAlpha(shp1,'one-region');

shp.Alpha = pc;

plot(shp)

[tri,xy2]=boundaryFacets(shp);
```

Ek 8’in devamı

```
figure
plot(xy2(:,1), xy2(:,2),'r-','MarkerSize',5)
%% Üçgenleme Gösterimi
patch(xy2(:,1),xy2(:,2),'g')
%% Sınır Alanının Gösterimi
plot(shp)
%% Detaylı Bilgiler (Çevre ve Alan Bilgisi)
cevre = perimeter(shp);
fprintf('Cevre = %6.3f\n',cevre)
alan = area(shp);
fprintf('Alan = %6.3f\n',alan)
%% Export (.shp uzantılı veri oluşturma)
S=mapshape(xy2(:,1),xy2(:,2));
S.FeatureName = 'kestel_optech_1200';
shapewrite(S,'kestel_optech_1200.shp');
```