YERSEL LAZER TARAYICILARDAN ELDE EDİLEN VERİLER KULLANILARAK OTOMATİK NESNE ÇIKARIMI

Resul ÇÖMERT
Yüksek Lisans Tezi

Uzaktan Algılama ve Coğrafi Bilgi Sistemleri Anabilim Dalı
Ocak-2014

ANADOLU ÜNİVERSİTESİ

JÜRİ VE ENSTİTÜ ONAYI

Resul ÇÖMERT' in **"Yersel Lazer Tarayıcılardan Elde Edilen Veriler Kullanılarak Otomatik Nesne Çıkarımı" başlıklı Uzaktan Algılama ve Coğrafi Bilgi Sistemleri** Anabilim Dalındaki, Yüksek Lisans Tezi 15.01.2014 tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği'nin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

	Adı-Soyadı	**İmza**
Üye (Tez Danışmanı)	**Yard. Doç. Dr. Uğur AVDAN**	
Üye	**Prof. Dr. Alper ÇABUK**	
Üye	**Doç. Dr. Saffet ERDOĞAN**	

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …… tarih ve …… sayılı kararıyla onaylanmıştır.

Enstitü Müdürü

ÖZET

Yüksek Lisans Tezi

YERSEL LAZER TARAYICILARDAN ELDE EDİLEN VERİLER KULLANILARAK OTOMATİK NESNE ÇIKARIMI

Resul ÇÖMERT

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Uzaktan Algılama ve Coğrafi Bilgi Sistemleri Anabilim Dalı

Danışman: Yard. Doç. Dr. Uğur AVDAN
2014, 177 sayfa

Coğrafi bilgi sistemleri çalışmaları, kültürel mirasın belgelenmesi ve kentsel planlama uygulamaları gibi birçok alanda yapılara ait 3 boyutlu modellere ihtiyaç duyulmaktadır. 3 boyutlu model oluşturmak için gerekli veriler, yersel lazer tarayıcılar ile hızlı ve hassas bir şekilde elde edilebilmektedir. Günümüzde yersel lazer tarayıcılar ile gerçekleştirilen veri toplama işlemi, küçük kullanıcı müdahaleleri ile otomatik olarak gerçekleştirilebilmektedir. Buna karşılık lazer tarayıcılardan elde edilen verilerin işlenmesi ile otomatik 3 boyutlu model oluşturma işlemi ise günümüzde araştırma konuları arasındadır. Yersel lazer tarayıcı verilerinden otomatik 3 boyutlu model oluşturma işlemi için genel olarak iki farklı yaklaşım vardır. Bunlar yüzey ağı modelleme ve geometrik temelli modellemedir. Yüzey ağı modelleme matematiksel olarak tanımlanması zor olan karmaşık yüzeylerin modellenmesinde kullanılan bir yöntemdir. Geometrik temelli modelleme ise matematiksel olarak ifadesi kolay olan yüzeylerin modellenmesinde kullanılan bir yaklaşımdır. Geometrik temelli modelleme işleminin en önemli aşaması, verilerin sınıflandırıldığı ve gruplandırıldığı segmentasyon aşamasıdır. Bu tez çalışmasında geometrik temelli modelleme yaklaşımının segmentasyon aşaması gerçekleştirilmiştir. Bu kapsamda Rastgele Örnek Konsensüsü (RANSAC: RANdom SAmple Consensus) algoritması yersel lazer tarayıcılardan elde edilen nokta bulutu verisine uygulanarak, nokta bulutu içinde yer alan düzlem, silindir, koni ve küre yüzeye sahip nesneler otomatik olarak çıkarılmıştır. RANSAC algoritmasının yüzey çıkarmadaki doğruluğunu değerlendirmek için otomatik olarak çıkarılan yüzeyler elle çıkarılan yüzeylerle karşılaştırılmıştır. Elde edilen sonuçlar RANSAC algoritmasının basit geometrik şekle sahip nesnelerin otomatik çıkarımında etkin bir araç olduğunu göstermiştir.

Anahtar Kelimeler: Yersel Lazer Tarayıcı, yüzey çıkarımı, RANSAC Algoritması, Segmentasyon, 3 Boyutlu Modelleme

ABSTRACT

Master of Science Thesis

AUTOMATİC OBJECT EXTRACTION USİNG TERRESTRIAL LASER SCANNER DATA

Resul ÇÖMERT

Anadolu University
Graduate School of Sciences
Remote Sensing and Geographic Information Systems Program

Supervisor: Assist. Prof. Dr. Uğur AVDAN

2014, 177 pages

Many study area such as GIS applications, cultural heritage documentation and urban planning applications needs 3D buildings models. The required data for 3D model generation can be obtained rapidly and accurately with terrestrial laser scanner. Nowadays, Data acquisition process with terrestrial laser scanner could be carried out automatically with little user response. On the other hand, Automatic 3D model which was generated using terrestrial laser scanner data, is a research subject. Generally, there are two different approach for automatic 3 model generation using terrestrial laser scanner data. These are meshing and geometric primitives modelling. Meshing is methot that is used for complex surfaces that are hard to define mathematically. Geometric primitives modelling is an approach that is used for simple surfaces which are easy to define mathematically. The most important step of geometric primitives modelling is segmentation that is done classification and clustering of data. In this thesis study, segmentation step of geometric primitives modelling approach is carried out. Within this scope, Random Sample Consensus (RANSAC) algorithm was applied on point cloud data that is obtained from terrestrial laser scanner. Then, objects that has planar, cylindrical, conical and spherical surface was automatically extracted in point cloud data. Automatic extracted surfaces was compared with manual extracted surface for the evaluation of accuracy of RANSAC algorithm on surface extraction. Obtained results shows that RANSAC algorithm is effective tool for automatic extraction of objects which have simple geometric shapes.

Keywords: Terrestrial Laser Scanner, surface extraction, RANSAC Algorithm, Segmentation, 3D modelling

TEŞEKKÜR

Tez çalışmam boyunca her türlü bilgi ve desteğini esirgemeyen, yardımları ile çalışmalarımı yönlendiren ve tezime danışmanlık yapan sevgili hocam Yrd. Doç. Dr. Uğur AVDAN'a,

Maddi ve manevi desteklerini her zaman hissettiğim sayın hocalarım Prof. Dr. Alper ÇABUK, Prof. Dr. Yücel Güney, Prof. Dr. Berkan Ecevitoğlu, Yrd. Doç. Dr. Hakan Uyguçgil, Yrd. Doç. Dr. Emrah PEKKAN, Arş. Gör. Dr. Muammer TÜN'e,

Arazi ve ofis çalışmalarımda kahrımı çeken ve bana yardımlarını hiç esirgemeyen kıymetli arkadaşlarım Sunay Mutlu, Serkan Kahraman, Emre Şenkal, Serhan Tuncer, Merve Ersoy, Esra Şenöz, Ebru Akdeniz, Emre Bektöre, Hasan Burak Özmen, Özge Bilget, Eren Bayrakçı, Melahat Cihan, Nuri Erkin Öçer'e

Programlama konusunda önümü açan ve desteklerini esirgemeyen Dr. Cüneyt Helvacı ve Araş. Gör. Ahmet Murat Türk'e

Ayrıca çalışmalarım sırasında her zaman yanımda ve destekçim olan Merve Köken'e ve bugünlere gelmemi sağlayan, tüm hayatım boyunca maddi ve manevi desteklerini esirgemeyen, her şeyin üzerinde tuttuğum değerli aileme en içten dileklerimle teşekkürlerimi sunarım.

Resul ÇÖMERT

Ocak-2014

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

KISALTMALAR DİZİNİ

ASCII: American Standard Code for Information Interchange

CAD: Computer Aided Design

EKK: En Küçük Kareler İlkesi

GPS: Global Positioning System

LIDAR: Light Detection And Ranging

LASER: Light Amplification by Stimulated Emission of Radiation

PCD: Point Cloud Data

PCL: Point Cloud Library

RANSAC: Random Sample Consensus

TLS: Terrestrial Laser Scanning

1. GİRİŞ

LAZER İngilizce; Light Amplification by Stimulated Emission of Radiation (LASER) (uyarılmış ışın salınımıyla ışığın kuvvetlendirilmesi) cümlesindeki kelimelerin baş harflerinden oluşmuş bir kelimedir. Lazer ışını 1960 senesinde ABD'de Theodore H. Maiman tarafından keşfedilmiştir. Normal ışık, dalga boyları muhtelif, rengârenk, yani farklı faz ve frekansa sahip dalgalardan meydana gelir. Lazer ışığı ise yüksek genlikli, aynı fazda, birbirine paralel, tek renkli, hemen hemen aynı frekanslı dalgalardan ibarettir (Demir, 2005). Lazer ışını endüstriyel süreçlerde, mühendislik alanında, tıpta, bilimsel araştırmalarda, meteorolojide, iletişimde, holografide ve savunma donanımlarında kullanılmaktadır.

Lazerin kullanım alanının geniş olması ve her geçen gün kullanım alanın artması lazerin oldukça önemli bir ışın olduğunu göstermektedir. Özellikle uygulama alanının genişliği, ışınların frekansların hassas bir şekilde kontrolünden ve yayılan ışının yayılma düzeninden kaynaklanmaktadır (Demir 2005). Lazer sahip olduğu özelliklerden dolayı ölçme bilimi alanında da kendine geniş bir kullanım alanı bulmuştur. Lazer ışını kullanılarak geliştirilen hava ve yersel lazer tarayıcı sistemler ile fiziksel yeryüzü nesneleri hassas ve detaylı olarak ölçülebilmektedir. Ölçüm sonucu elde edilen veriler kullanılarak 3 boyutlu modeller elde edilebilmektedir.

Fiziksel nesnelerin 3 boyutlu modelleri birçok alanda hızlı bir şekilde kullanışlı hale gelmektedir. Coğrafi Bilgi Sistemleri, sanal turizm uygulamaları, kentsel planlamalar, seyrüsefer sistemleri, kültürel mirasın belgelenmesi vb. birçok alanda 3 boyutlu modellere ihtiyaç duyulmaktadır. Bunun yanında gün geçtikçe, iki boyutlu veri sunumunun yerini 3 boyutlu veri sunumu almaktadır. Bunun en temel nedenlerinden birisi, 3 boyutlu modeller üzerinden yapılacak analizlerin ve elde edilecek bilgilerin 2 boyutlu haritalar üzerinden elde edilecek bilgilere göre avantajlı olmasıdır. Örneğin kentsel bir alanda yapılacak kent planlamasında, kenti gerçekçi bir şekilde yansıtan 3 boyutlu kent modeli üzerinden yapılan analizler, 2 boyutlu haritalardan elde edilecek bilgilerden çok daha etkilidir. Benzer şekilde, kamu güvenliği için acil durum anında 3 boyutlu bina modelleri üzerinden strateji geliştirmek daha başarılı sonuçlar vermektedir (Pu ve Vosselman, 2009b).

Fiziksel nesnelere ait 3 boyutlu modeller elde edilebilmesi için nesneleri temsil eden noktaların 3 boyutlu koordinatlarının bilinmesi gerekmektedir. Günümüzde 3 boyutlu veri elde etmek için çeşitli teknolojiler kullanılmaktadır. 3 Boyutlu modelle çalışmalarında sıklıkla kullanılan teknolojileri, fotogrametri ve lazer tarama teknolojileri olarak ikiye ayırmak mümkündür. Fotogrametri ve lazer tarama teknolojilerini de tarama konumuna göre hava ve yersel olmak üzere ikiye ayırmak mümkündür. Fotogrametri ve lazer tarama teknolojileri ayrı ayrı veya bütünleşik olarak kullanılarak, farklı alanlarda farklı amaçlar için 3 boyutlu modelleme çalışmaları gerçekleştirilmektedir.

Kentsel alanlarda kentin bütününe yönelik yapılan 3 boyutlu modelleme çalışmaları için gerekli verilerin elde edilmesinde hava fotogrametrisi ve hava lazer tarama (LIDAR) kullanılmaktadır. Bu tekniklerle elde edilen veriler ayrı ayrı veya birlikte kullanılarak 3 boyutlu kent modelleri ve sayısal arazi modelleri elde edilebilmektedir (Dorninger ve Pfeifer, 2008). Ancak hava fotogrametrisi ve LIDAR'dan elde edilen veriler yeryüzünde yer alan bina ve objelerin çatıları ile sınırlı kalmaktadır. Yeryüzü nesnelerine ait gerçekçi bir model oluşturabilmek için nesnelerin cephelerinin de modellenmesine ihtiyaç vardır. Bu işlem ise yersel fotogrametri ve yersel lazer tarama yöntemleri ile gerçekleştirilebilmektedir.

Son yıllarda, özellikle veri toplama hızı ve hassasiyetinden dolayı yersel lazer tarayıcılar 3 boyutlu modellemeye ihtiyaç duyan birçok alanda kullanılır hale gelmiştir. Tersine mühendislik uygulamaları, kültürel mirasın belgelenmesi, sanal müze uygulamaları, kentsel alanlarda bina modellemeleri, yersel lazer tarama tekniğinin yoğun bir şekilde kullanıldığı alanlardır. Özellikle kentsel alanlarda yapılan modelleme çalışmalarında yersel lazer tarayıcılar ve mobil lazer tarayıcılar kullanılarak yeryüzü nesnelerine ait hızlı ve hassas 3 boyutlu nokta bulutu verisi elde etmek mümkündür (Boulaassal ve ark., 2007).

Yersel lazer tarama tekniği ile yüzey şekillerine bağlı kalmadan veri elde edilmektedir. Bu teknikle, yersel lazer tarayıcının kurulu olduğu bir noktadan hızlı ve yüksek hassasiyete sahip, çok yoğun 3 boyutlu nokta bulutu verisi elde edilebilmektedir. Bir yeryüzü nesnesini tamamen kaplayacak nokta bulutu verisi elde etmek için ise birden fazla noktadan elde edilen nokta bulutu verisine ihtiyaç duyulmaktadır. Farklı noktalardan elde edilen nokta bulutlarının, referans olarak

seçilen bir koordinat sisteminde birleştirilmesi gerekmektedir. Nokta bulutları ortak bir koordinat sisteminde birleştirildikten sonra nesneye ait tam bir 3 boyutlu nokta bulutu modeli hazır hale gelmektedir (Abdelhafiz, 2009).

Yersel lazer tarayıcılar ile çok kısa sürede binalara ait çok fazla 3 boyutlu nokta verisi elde edilmesine rağmen, elde edilen nokta bulut verisi veri tabanlarında oldukça büyük yer kaplamaktadır. Bundan dolayı nokta bulutu veri boyutunu küçültmek için yapıların 3 boyutlu modellenmesi önemlidir (Boulaassal ve ark. 2010). Genellikle iki farklı modelleme tekniği vardır. Bunlar, yüzey ağı modelleme (meshing) ve geometrik temelli model oluşturmadır.

Yüzey ağı modelleme (meshing) lazer tarayıcılar tarafından elde edilen verilerin işlenmesinde kullanılan önemli bir yöntemdir. Bu yöntem kale veya heykel gibi karmaşık ve düzensiz objelerin modellenmesinde son derece kullanışlıdır. Yüzey ağı modelleme işleminde lazer tarayıcılardan elde edilen nokta bulutları alınır ve bir üçgen ağı oluşturulur. Oluşturulan bu üçgen ağı noktaların oluşturduğu yüzeye oldukça yakın bir yüzey ağını temsil eder. Taranan nesnenin büyüklüğü ve karmaşıklığına bağlı olarak oluşturulan yüzey ağı modelleri fazlaca çeşitlilik göstermektedir. Bundan dolayı taranan yüzeydeki tüm çeşitleri modellemek için yüzey ağı modelleme sistemi, otomatik modelleme temeli üzerine kurulmaktadır (Stephan ve ark., 2002).

Geometrik temelli model oluşturma yöntemi de yüzey ağı modelleme gibi yersel lazer tarayıcılardan elde edilen verilerin işlenmesinde kullanılan bir yöntemdir. Bu yöntem daha çok geometrik olarak tanımlanması kolay olan nesnelerin (düzlem, silindir, koni, küre vb.) modellenmesinde tercih edilmektedir. Bu konuda, yersel lazer tarayıcılardan elde edilen veriler kullanılarak yapılan çalışmalar incelendiğinde geometrik temelli modelleme yönteminin genellikle kent merkezlerinde düzlem yüzeye sahip binaların modellenmesi için kullanıldığı görülmektedir. Yöntem genel olarak verilerin düzenlenmesi, segmentasyon, yüzey yakalama ve modelleme aşaması olarak dört işlem adımında gerçekleşmektedir.

Geometrik temelli modellemede segmentasyon işlemi modellemenin en önemli aşamasını oluşturmaktadır. Bu aşamada geometrik özellikleri belirli nesneler, belirlenen bir yönteme göre sınıflandırılmakta ve gruplandırılmaktadır.

Yüzey yakalama ve modelleme işleminin doğruluğu segmentasyon işleminin başarısına bağlıdır.

Yersel lazer tarayıcı verilerinin segmentasyonu işleminde nokta bulutu içindeki veriler sahip oldukları geometrik özelliklere bağlı olarak gruplandırılıp sınıflandırılmaktadır. Günümüzde nokta bulutu verilerinin segmentasyonunda farklı yöntemler kullanılmaktadır. Bu yöntemlerden başlıcaları Hough dönüşümü, bölge geliştirme ve RANSAC algoritmasıdır.

Rastgele Örnek Konsensüsü (RANSAC: RANdom SAmple Consencus) algoritması uzun yıllardır etkili bir model tahminleme aracı olarak bilgisayar görselleştirme ve dijital görüntü işleme alanında yoğun bir şekilde kullanılmaktadır. Son yıllarda ise gerek hava lazer tarama sistemlerinden elde edilen verilerin segmentasyonunda, gerekse yersel lazer tarama sistemlerinden elde edilen verilerin segmentasyonunda kullanılmaktadır.

Bu tez çalışmasında yersel lazer tarayıcılardan elde edilen nokta bulutu verisi üzerinde RANSAC algoritması ile geometrik temelli model oluşturma yaklaşımının segmentasyon aşaması gerçekleştirilmiştir. Bu kapsamda;

- Çalışmanın ilk aşamasında Laboratuvar ortamında düzlem, silindir, koni ve küre gibi geometrik nesneleri temsil eden cisimler, yersel lazer tarayıcı ile taranarak bir test verisi elde edilmiştir,
- C++ ortamında derlenen RANSAC algoritması ile test verisine ait nokta bulutu içinden düzlem, silindir, koni ve küre yüzeylere ait noktalar otomatik olarak bulup çıkarılmıştır,
- Test verisi içinden RANSAC algoritması ile düzlem yüzey çıkarma işlemi için, nokta bulutu içinde yer alan noktaların yüzey normallerine bağımlı ve yüzey normallerinden bağımsız olmak üzere iki farklı yaklaşım izlenmiş ve aralarındaki farklılıklar ortaya konmuştur,
- Çalışmanın ikinci aşamasında C++ ortamında derlenen RANSAC algoritması araziden yersel lazer tarayıcı ile elde edilen nokta bulutları üzerine uygulanarak geometrik yüzeylerin çıkarım işlemi gerçekleştirilmiştir,
- Çalışmanın son aşamasında ise nokta bulutu içinden RANSAC algoritması ile otomatik olarak çıkarılan bazı yüzeyler, nokta bulutu içinden elle de çıkarılarak

birbirleri ile karşılaştırmıştır. Elle çıkarılan yüzeyler doğru kabul edilerek, otomatik çıkarılan düzlemlerin doğrulukları araştırılmıştır.

2. LİTERATÜR ÖZETİ

Bu tez çalışması kapsamında yersel lazer tarayıcı verilerinden geometrik temelli nesne çıkarma işlemi gerçekleştirilmiştir. Tez çalışmasının literatür araştırması bölümünde hem yersel hem de hava lazer tarama sistemlerinden elde edilen nokta bulutu verileri içinden otomatik nesne çıkarımını konu alan makaleler incelenmiştir. İncelenen makalelerde geometrik temelli model oluşturma çalışmalarının genellikle kentsel alanlarda düzlem yüzeye sahip binaların modellenmesinde kullanıldığı görülmüştür. Çalışma kapsamında incelenen ve yararlanılan kaynakların özetleri aşağıda sunulmuştur.

Vosselman ve Dijkman (2001) yapmış oldukları çalışmada, hava lazer tarama sisteminden elde ettikleri verileri ve çalışma alanlarına ait yer planını kullanarak 3 boyutlu bina modeli oluşturma işlemini gerçekleştirmişlerdir. Çalışmalarında düzensiz nokta bulutu içinden binaların düz çatı yüzeylerini çıkarmak için 3 boyutlu Hough dönüşümü yöntemini kullanmışlardır. Hough dönüşüm yöntemi kullanılarak çıkarılan yüzeylerden binaların 3 boyutlu modellenmesi için ise çalışmalarında iki farklı strateji izlemişlerdir. Bunlardan birincisi düzlem yüzeyler arasında kesişen kenarları bulma ve yükseklik farkı olan kenarlar üzerinden geçiş yapma temeline dayanmaktadır. İkinci stratejide ise Hough dönüşümü ile belirlenen tüm düzlem yüzeyler, binanın herhangi bir parçasının modeli olması kabulüne dayanmaktadır. Bu iki yaklaşım kullanılarak düzlem yüzeylerin birleştirilerek modellenmesi işlemi gerçekleştirilmiştir. Yapılan deneysel çalışmalar sonucunda araştırmacılar ikinci yaklaşımdan daha istikrarlı sonuçlar aldıklarını ortaya koymuşlardır.

Vosselman ve ark. (2004) yapmış oldukları çalışmada, hava lazer tarayıcılardan ve yersel lazer tarayıcılardan elde edilen nokta bulutu içinden nesne çıkarımında kullanılan yöntemleri incelemişlerdir. Araştırmacılar çalışmalarında, nokta bulutu içinde yumuşak yüzeylerin bölütlenmesi (segmentasyonu), düzlem yüzey çıkarmada kullanılan metotları ve parametreleri belirli şekillerin (silindir, küre, düzlem gibi) tespitinde kullanılan yöntemlerin neler olduğunu tanımlamışlardır. Tanımladıkları bu yöntemlerin sanayi uygulamaları, kentsel planlama, su yönetimi, orman envanterinin çıkarılması gibi birçok alanda kullanılabilir olduğunu örnek uygulamalar ile göstermişlerdir.

Overby ve ark. (2004) yapmış oldukları çalışmada, LIDAR verileri ile kadastro verilerini birleştirerek kentsel bir alan için otomatik 3 boyutlu bina modelleri üretmişlerdir. Yaptıkları çalışmada, LIDAR verilerinden Hough dönüşümü yöntemi ile binaların çatılarına ait düzlem yüzeyleri çıkarmışlardır. Çalışma kapsamında kullanılan LIDAR veri setinin çözünürlüğü düşük olduğundan çalışma alanına ait tüm binaların çatılarının çıkarılmasında problem yaşanmış, problem yaşanan alanlara sonradan elle müdahale edilmiştir. Daha sonra elde edilen çatılar poligon olarak mesh model haline dönüştürülmüştür. Çatılar ve teknik haritadaki bina sınırları birleştirilerek çalışma alanında yer alan binalara ait 3 boyutlu modeller üretilmiştir. Son olarak LIDAR verisi ve teknik haritanın birleştirilmesinde bazı binalarda kayıklıkların olduğu gözlemlenmiştir.

Vieira ve Shimada (2005) yapmış oldukları çalışmada, yersel lazer tarayıcılar ile elde edilen nokta bulutu içinde otomatik olarak yüzey çıkarabilen bir algoritma geliştirmişlerdir. Geliştirdikleri algoritma segmentasyon yöntemlerinden birisi olan bölge geliştirme algoritması üzerine kurulmuştur. Geliştirdikleri algoritma ile düzensiz nokta bulutu içinden geometrik olarak uyumlu yüzeylerin çıkarılma işlemini gerçekleştirilmiştir. Araştırmacılar elde ettikleri sonuçları değerlendirdiklerinde geliştirdikleri algoritmanın küçük kullanıcı müdahaleleri ile hızlı bir şekilde yüzey çıkarım işlemi içim kullanılabilir olduğu sonucuna ulaşmışlardır. Ayrıca algoritmanın tersine mühendislik, sanayi tasarımı, hızlı prototip üretme gibi alanlarda kullanılabilecek bir yöntem olduğunu belirtmişlerdir.

PU ve Vosselman (2006) yersel lazer tarayıcı ile elde edilen nokta bulutu verisi içinden binalara ait detayların otomatik olarak çıkarılmasına yönelik bir çalışma yapmışlardır. Yaptıkları çalışma nokta bulutu verisinin bölütlenmesi (segmentasyonu), detay tanıma ve kalite analizi işlemlerinde oluşmaktadır. Araştırmacılar çalışmalarının segmentasyon aşamasında düzlem yüzey geliştirme (planar surface growing) algoritmasını kullanarak nokta bulutu içinde bulunan potansiyel tüm düzlem yüzeyleri gruplandırmışladır. Detay tanıma aşamasında ilk önce bina detaylarına ait bazı sınırlamalar getirmişlerdir. Bu sınırlamalar detayların boyut sınırlaması, yönelim sınırlaması, konum sınırlaması ve topoloji sınırlamalarıdır. Daha sonra ise bölütlenen nokta bulutunun dışbükey boşluklarını (convex hull) elde etmişlerdir. Elde ettikleri dışbükey boşluklara bağlı olarak

segmente edilen nokta bulutları içinden bina detayları olan kapı, duvar, pencere, kiriş, çatı ve kolon gibi detayların çıkarılması işlemini gerçekleştirmişlerdir. Çalışmanın kalite analizi bölümünde ise araştırmacılar otomatik çıkarılan detay sayısı ile gerçek detay sayısını karşılaştırmışladır. Karşılaştırma sonucunda pencere gibi küçük detayların eksik çıkarıldığı görülmüştür. Bu eksikliğin nedenini ise pencere gibi detaylardan, yeteri kadar lazer noktasının geri yansımaması olduğunu belirtmişlerdir.

Schnabel ve ark. (2007) yapmış oldukları çalışmada, yersel lazer tarayıcı verileri içinden geometrik olarak tanımlanması kolay olan düzlem, silindir, koni, küre ve torus gibi nesnelerin otomatik olarak tanımlanıp çıkarılması işlemini gerçekleştirmişlerdir. Bu nesnelerin nokta bulutu içinden çıkarılması işlemi için araştırmacılar RANSAC algoritmasını kullanmışlardır. Çalışma sonucunda araştırmacılar, RANSAC algoritmasının lazer tarama verilerinden geometrik şekillerin çıkarılması için hızlı ve etkili bir algoritma olduğunu göstermişlerdir.

Pu ve Vosselman (2007) yapmış oldukları çalışmada, yersel lazer tarayıcılar ile elde edilen bina cephelerinde yer alan pencerelerin otomatik olarak çıkarılması işlemini gerçekleştirmişlerdir. Otomatik bina cephesi oluşturma işleminde bina cepheleri üzerinde yer alan pencerelerin tespit edilmesinde sorunlar yaşanmaktadır. Araştırmacılar bu çalışmalarında bina penceresinde perde olup olmamasını ele alan iki farklı yaklaşımla bina pencerelerinin otomatik çıkarılmasını sağlamışladır. Pencerede perde olmaması durumunda nokta bulutu camdan geri yansımadığından duvar üzerindeki pencere bölümü boş kalmaktadır. Perde olması durumunda ise yine perde olan bölge duvardan farklı bir düzlem olarak nokta bulutu içinden çıkarılmaktadır. Bu kapsamda araştırmacılar ilk önce nokta bulutu verisine düzlem bölge geliştirme (planar surface growing) algoritması uygulayarak binaların düzlem cephelerini çıkarmışladır. Daha sonra elde ettikleri binalara ait düzlem cephe verilerine üçgenleme yaparak TIN oluşturmuşladır. Oluşturulan TIN verisinde duvar üzerinde büyük kenara sahip kenarlar, duvar üzerinde boşluk alanlar olan pencereleri temsil etmektedir. Bu doğrultuda büyük kenarları temsil eden noktalar ham nokta bulutu içinden filtrelenerek çıkarılmıştır. Böylelikle sadece pencerelerin kenarlarını temsil eden nokta verileri çıkarılmıştır. Çalışmada araştırmacılar geliştirdikleri yöntemi farklı veri setlerine uygulayarak test etmişlerdir. Elde

ettikleri sonuçlar geliştirdikleri yöntemin nokta bulutu içinden pencere çıkarmada etkili bir yöntem olduğunu göstermiştir.

Boulaassal ve ark. (2007) yapmış oldukları çalışmada, bir bina cephesine ait yersel tarayıcı verileri içinden otomatik olarak binanın düzlem yüzeylerinin çıkarımı işlemini gerçekleştirmişlerdir. Düzlem yüzeylerin çıkarılması işleminde araştırmacılar RANSAC algoritmasını kullanmışlardır. RANSAC algoritmasını nokta bulutu üzerine ardışık olarak uygulayarak veri içinde yer alan mevcut tüm yüzeylerin çıkarılmasını sağlamışlardır. Araştırmacılar otomatik olarak çıkardıkları düzlem yüzeylerin doğruluğunu araştırmak içinde nokta bulutu içinden binaya ait bir düzlem cepheyi elle çıkarmışlardır. Aynı düzleme ait otomatik çıkarılan nokta bulutu ve elle çıkarılan nokta bulutu verileri karşılaştırdıklarında verilerin % 94.6 oranında tutarlı olduğu sonucuna ulaşılmıştır. Çalışma sonuçları RANSAC algoritmasının yersel lazer tarayıcı verilerinden düzlem yüzeylerin çıkarılması için uygun bir yöntem olduğunu göstermiştir.

Pu (2008) yapmış olduğu çalışmada, binalara ait yersel lazer tarayıcı verilerinden 3 boyutlu bina modeli üretme işlemini gerçekleştirmiştir. Bu kapsamda ilk önce yersel lazer tarayıcı verilerinde yer alan düzlem yüzeyler, düzlem yüzey geliştirme algoritması ile bölütlenmiştir. Daha sonra bina detayları olan duvar, kapı, pencere, çatı ve kiriş bölümleri çıkarılmıştır. Çıkarılan detayların hepsi düzlemsel yüzeyleri içeren nokta kümeleridir. Nokta kümeleri halinde bulunan düzlemler poligon model olarak modellenmiştir. Yapılan modelleme sonucunda balkon ve çatı penceresi gibi alanlarda modelleme işlemi başarısızlıklar olduğu gözlemlenmiştir. Ayrıca eğik yüzeylerin modellenme işlemi gerçekleştirilememiştir. Araştırmacı ilerleyen çalışmalarda eğik yüzeylerin çıkarılıp modellenmesi ve bunların düzlem yüzeyler ile birleştirilmesi üzerine çalışılması gerektiğini vurgulamıştır. Ayrıca binaların 3 boyutlu modellenmesi aşamasında, gerçek hayatta olan binalara ait bilgilere bağlı bir modellemenin esas alınmasını önermiştir. Bu şekilde gerçekçi 3 boyutlu bina modellerinin daha başarılı bir şekilde elde edilmesinin mümkün olacağını savunmuştur.

Rutzinger ve ark. (2008) yapmış oldukları çalışmada, hem hava lazer tarama sistemi hem de mobil lazer tarama sistemini kullanarak çalışma alanlarına ait 3 boyutlu nokta bulutu verisi elde etmişlerdir. Çalışma alanı 45 adet bina

içermektedir. Araştırmacılar elde ettikleri veri setlerine ayrı ayrı segmentasyon (bölütleme) işlemi uygulamışlardır. Her iki sistemden de elde edilen nokta bulutlarından binaların düzlem duvarlarını çıkarmak için düzlem yüzey geliştirme (planar surface growing) algoritması kullanılmıştır. Çalışma sonucunda mobil lazer tarayıcı verilerinden binalara ait toplam 135 düzlem bina cephesi çıkarılırken, hava lazer tarama sisteminden elde edilen verilerde 262 adet bina düzlem cephesi çıkarılmıştır. Mobil yersel lazer tarayıcı ile elde edilen bina cepheleri daha fazla nokta verisi içerirken, hava lazer tarama sisteminden elde edilen veriler duvarlara ait az sayıda nokta verisi içerdiği gözlemlenmiştir. Araştırmacılar son olarak gelecek çalışmalarda bu iki sistemden elde edilen veriler birleştirilerek 3 boyutlu kent modellerinin üretilmesinin oldukça kolay olacağını vurgulamışlardır.

Boulaassal ve ark. (2009) yapmış oldukları çalışmada, düzlem yüzeye sahip binalar ait yersel lazer tarayıcı verileri içinden, otomatik düzlem yüzey ve düzlem yüzeylerin köşe noktalarını çıkarma işlemini gerçekleştirmişlerdir. Araştırmacılar kullandıkları yersel lazer tarayıcı veri seti içinden düzlem yüzeylerin otomatik olarak çıkarılması için RANSAC algoritmasını kullanmışlardır. Daha sonra RANSAC algoritması ile çıkarılan düzlem yüzeylere Delaunay üçgenleme yöntemi uygulayarak her bir düzlemin kenar noktalarını çıkarmışlardır. Kenar nokta çıkarım işleminde düzlem cepheler üzerinde yer alan pencere ve kapı alanlarının da boşluklar oluştuğu için bu alanlarda kenar nokta çıkarım işleminde çıkarılmıştır. Araştırmacılar elde ettikleri sonuçları test etmek için ham lazer tarama verisi üzerinden çizim işlemi gerçekleştirmişlerdir. Otomatik çıkarılan kenar noktalar ve elle çıkarılan kenar çizgiler arasındaki farklar karşılaştırıldığında düz çizgilerde ±2.5 cm, yay şeklinde olan çizgilerde ± 3 cm fark olduğu görülmüştür. Elde edilen bu sonuçlar RANSAC algoritmasının düzlem yüzey çıkarmada ve Delaunay üçgenleme yönteminin kenar nokta çıkarımında kullanılabilir bir yöntem olduğunu göstermiştir.

Alshawa ve ark. (2009) yapmış oldukları araştırma projesi kapsamında, bir adet mobil lazer tarama sistemi geliştirmişlerdir. Geliştirdikleri mobil lazer tarama sistemi ile test sahası olarak seçtikleri alanda binalara ait lazer tarama verisi elde etmişlerdir. Araştırmacılar elde ettikleri bina verilerinden otomatik olarak düzlem yüzeylerin çıkarılması ve çıkarılan düzlem yüzeylerin kenar noktalarının otomatik

olarak elde edilmesi işlemini gerçekleştirmişlerdir. Düzlem yüzey ve kenar nokta çıkarım işleminde araştırmacılar, Boulaassal ve ark. 2009 yılında statik lazer tarama verilerine uygulamış olduğu yöntemi mobil lazer tarama sisteminden elde edilen verilere uygulamışladır. Kullandıkları yöntemde düzlem yüzey çıkarımı için RANSAC algoritması, kenar nokta çıkarımı için ise Delaunay üçgenleme algoritması uygulanmıştır. Uygulanan yöntemler sonucunda nokta bulutu içinden düzlem yüzeyler ve düzlem yüzeylerden de kenar nokta çıkarım işlemi başarı ile yapılmıştır. Ancak geliştirilen mobil lazer tarama sistemi hala geliştirme aşamasında olduğundan elde edilen lazer tarama verilerindeki hatalar otomatik nesne çıkarımını da etkilediği görülmüştür.

Pu ve Vosselman (2009a) yapmış oldukları çalışmada, yakın fotogrametri ile elde edilen görüntüler ile lazer tarama verilerinden elde edilen verileri kullanarak yarı otomatik bina cephe modellemesi işlemi gerçekleştirmişlerdir. Araştırmacılar çalışmalarında ilk önce yersel lazer tarayıcı verilerinden Pu ve Vosselman'ın 2009 yılında yapmış oldukları bilgi tabanlı 3 boyutlu bina modeli oluşturma çalışmalarında sundukları yöntemi kullanarak binalara ait 3 boyutlu çok yüzlü poligon model oluşturmuşlardır. Daha sonra model oluşturdukları binalara ait dijital kamera ile görüntüler elde etmişlerdir. Elde ettikleri görüntülerden binalara ait ortofoto görüntüler üretmişlerdir. Üretilen ortofoto görüntü üzerinden otomatik olarak bina cephelerinin kenarlarını çıkarmışladır. Görüntü üzerinden çıkarılan kenarlar ile yersel lazer tarayıcı verilerinden çıkarılan kenarlar birleştirilmiştir. Araştırmacılar son olarak görüntüleri doku olarak 3 boyutlu model üzerine atayarak 3 boyutlu foto realistik bir model elde etmişlerdir.

Pu ve Vosselman (2009b) yapmış oldukları çalışmada, bilgi tabanlı olarak yersel lazer tarayıcı verilerinden otomatik bina modelleme işlemi gerçekleştirmişlerdir. Çalışma kapsamında iki farklı veri seti kullanılmıştır. Kullanılan veri setleri birden fazla bina verisi içermektedir. Araştırmacılar otomatik olarak 3 boyutlu bina modeli üretmek için ilk önce düzlem yüzey geliştirme algoritmasını kullanarak nokta bulutu verisini düzlem yüzeylere bölütlemiştir. Bölütleme işleminden sonra anlamsal detay(kapı, pencere, duvar, çatı vb) çıkarım işlemi gerçekleştirilmiştir. Bu işlem için detaylara ait gerçekte bilinen boyut (pencere boyutu gibi), konum (çatının cephe duvarı üstünde olması gibi) yönelim

(duvarların dikey, çatıların asla dikey olmaması gibi), topoloji (duvarların çatı ile kesişmesi gibi), nokta yoğunluğu (perde olmayan pencerelerde lazer verisinin az olması gibi) gibi bilgiler formüle edilerek binalara ait detayların çıkarımında kullanılmıştır. Detay çıkarım işlemi sonucunda çatı, duvar, çıkma, kapı gibi detaylara ait nokta bulutları gruplanmıştır. Binalara ait detaylar çıkarıldıktan sonra her bir veri seti için poligon model oluşturma işlemi gerçekleştirilmiştir. Detaylara ait poligon model oluşturma işleminde detayların büyüklüklerine göre en küçük kareler ile model oluşturma, dışbükey boşluk yöntemi ile model oluşturma veya içbükey poligon oluşturma yöntemleri kullanılmıştır. Detaylara ait poligon modeller oluşturulduktan sonra tüm poligonlar birleştirilerek 3 boyutlu çok yüzlü bina modelleri elde edilmiştir. Elde edilen sonuçlar araştırmacılara tarafından geliştirilen yöntemin düzlem yüzeye ait binalarda oldukça başarılı sonuçlar verdiğini göstermiştir. Bununla birlikte karmaşık bir yapıya sahip veya eğri bir yapıda olan yüzeylerin modellenmesinde sorunlar olduğu gözlemlenmiştir.

Boulaassal ve ark. (2010) yaptıkları çalışmada, yersel lazer tarayıcılardan elde edilen nokta bulutu içinden binalar ait 3 boyutlu parametrik modeller üretmişlerdir. Bu kapsamda binalara ait düzlem yüzeyler RANSAC algoritması kullanılarak çıkarılmıştır. Daha sonra çıkarılan düzlemler üzerinden üçgenleme ile duvar ve pencerelere ait kenar noktaları çıkarılmıştır. Kenar nokta çıkarım işlemi Delaunay üçgenleme yöntemi ile gerçekleştirilmiştir. Düzlem yüzeylere ait kenar noktalar çıkarıldıktan sonra, çıkarılan kenar nokta veri setine tekrardan RANSAC algoritması uygulanarak kenar noktaların oluşturduğu düz ve eğri kenarlar çıkarılmıştır. Bu işlem aşamasında RANSAC algoritması ile belirlenen çizgi model üzerine düşen noktalar; düz kenara ait noktaları, diğer noktalar; eğri kenarlara ait noktaları temsil edecek şekilde gruplandırılmıştır. Çıkarılan kenar noktaları binanın cephe ve pencerelerine ait noktalardır. Çıkarılan noktaların oluşturduğu pencerelerin üst, alt mesafe değerlerinden pencerenin boyu, sağ ve sol mesafe değerlerinden ise genişlikleri hesaplanmıştır. Hesaplanan bu değerlere göre model üzerine elle binaya ait gerçek duvar ve pencere görüntüleri atanmıştır. Atama işlemi için bir kullanıcı ara yüzü geliştirilmiştir. Pencere ve duvarlara ait görüntüler bu ara yüz aracılığı ile model üzerine yerleştirilerek bina cephesinin 3 boyutlu parametrik modeli oluşturulmuştur.

Wang ve ark. (2011) yaptıkları çalışmada, kentsel alanda mobil lazer tarayıcı sistemi ile binalara ait elde ettikleri nokta bulutu verisi içinden bina cephelerinin ve pencerelerinin otomatik olarak bulunup nokta bulutu içinde çıkarılması uygulamasını yapmışlardır. Bu kapsamda araştırmacılar ilk önce noktalara ait yükseklik bilgilerine göre noktaların histogramlarını oluşturmuşlardır. Oluşan histograma göre nokta bulutu içinden zemine ait veriler filtrelenmiştir. Filtrelenen verilere RANSAC algoritması uygulanarak veri seti içindeki mevcut düzlem yüzeyler otomatik olarak çıkarılmıştır. Araştırmacılar çıkarılan düzlem yüzeyler üzerinden pencereleri ayrı olarak çıkarmak için bir tane pencere tanıma algoritması geliştirmişlerdir. Geliştirdikleri algoritma ile pencerelere ait kenar noktaların çıkarım işlemini gerçekleştirmişlerdir. Bu şekilde cephe ve pencereler ayrı ayrı sınıflandırılabilmiştir. Çalışma sonucunda pencere tanıma algoritması 5 farklı bina üzerinde denenmiştir. Elde edilen sonuçlar geliştirilen algoritmanın bazı eksikleri olmasına rağmen kullanılabilir olduğunu göstermiştir.

Boulaassal ve ark. (2011) yaptıkları çalışmada, mobil lazer tarama sistemi ile binalara ait elde ettikleri nokta bulutu verilerinde CAD yazılımlarında kullanılabilecek çizgi formatında vektör model oluşturmuşlardır. Araştırmacılar yapmış oldukları çalışmada, çalışma alanlarına ait mobil lazer tarama verisine RANSAC algoritması uygulayarak veri seti içinde yer alan düzlem yüzeyleri çıkarmışlardır. Çıkarılan düzlem yüzeylere Deleunay üçgenleme yöntemi kullanılarak cephelerin kenar noktaları çıkarılmıştır. Çıkarılan cephelere ait kenar noktalar binari görüntüye dönüştürülmüştür. Görüntü üzerinde yer alan noktalar bölge geliştirme ilkesine göre sıralanmış ve daha sonra ardışık noktalar bir çizgi ile birleştirilerek her bir düzlem yüzey için CAD model elde edilmiştir. Yapılan çalışmada elde edilen sonuçlar kullanılan yöntemlerin Mobil lazer tarayıcılardan CAD model elde etmede oldukça etkili olduğunu göstermiştir.

Arachchige ve ark. (2012) yapmış oldukları çalışmada, mobil lazer tarama sistemi ile elde ettikleri düzensiz nokta bulutu içinden bina cephelerinin otomatik olarak belirlenip, çıkarılması işlemini gerçekleştirmişlerdir. Bu kapsamda araştırmacılar ilk önce lokal yükseklik histogramını kullanarak nokta bulutu içinden, zemine ait noktaları filtreleyerek temizlemişlerdir. Daha sonra nokta bulutuna pürüzlü yüzey sınıflandırma algoritması uygulanmıştır. k-yakın komşuluk

temeline dayanan bu algoritma ile nokta bulutu içinde binayı temsil edene noktalar ile binaya ait olmayan farklı objeleri (ağaç, bitki vb.) temsil eden noktalar sınıflandırılmıştır. Son olarak binaları temsil eden nokta kümesi üzerine RANSAC algoritması ve bölge geliştirme algoritması uygulanarak bina cepheleri çıkarılmıştır. Elde edilen sonuçlar çalışma kapsamında uygulanan yöntemlerin, ham nokta bulutu içinden bina cephelerinin çıkarılması için etkin bir yöntem olduğunu göstermiştir.

Karslı ve Pfeifer (2012) yapmış oldukları çalışmada viyana şehir merkezi için üretilmiş LIDAR verilerini kullanılarak yalın yeryüzü ve bina detaylarının otomatik olarak çıkartılması işlemini gerçekleştirmiştir. Çalışma kapsamında veri seti olarak seçilen LIDAR verilerine, RANSAC algoritması uygulanarak, veri setinde yer alan düzlem yüzeylerin çıkartılmıştır. Çıkartılan düzlem yüzeyler zemine ve bina çatılarına ait düzlem yüzeyleri temsil etmektedir. Elde edilen sonuçlar RANSAC algoritmasının LIDAR veri seti içinden düzlem yüzey çıkarmada etkili bir algoritma olduğunu göstermiştir.

Núñez ve ark. (2012) yapmış oldukları çalışmada Selinute Yunan arkeolojik bölgesinde (Sicilya, İtalya) yer alan G tapınağının en büyük sütunun yeniden oluşturulması işlemini gerçekleştirmişlerdir. Çalışmaya konu olan sütun günümüzde devrilmiş ve sekiz parçaya bölünmüş halde bulunmaktadır. Araştırmacılar, çalışmaları kapsamında çalışmalarına konu olan sütunu yersel lazer tarayıcı ile tarayarak sütuna ait nokta bulutu verisi elde etmişlerdir. Elde edilen nokta bulutu verisine ikinci derece yüzeylerin hesaplanmasında kullanılan evrimsel algoritma uygulanarak sütün kırılma yüzeylerinden birleştirilmiştir. Daha sonra nokta bulutu verisin RANSAC algoritması uygulanarak konik bir şekilde olan sütun üzerine düşen nokta verileri tespit edilmiştir. Tespit edilen veriler kullanılarak da sütunun yeniden modellenme işlemi gerçekleştirilmiştir. Çalışma sonucunda elde edilen sonuçlar çalışmada kullanılan yöntemlerin arkeolojik alanlarda yer alan sütunların yeniden oluşturulmasında kullanılabilir bir yöntem olduğunu göstermiştir.

Çömert ve Avdan (2013) yapmış oldukları çalışmada kentsel alanda yersel lazer tarayıcı ile elde ettikleri nokta bulutu verisi içinde bina düzlem yüzeylerinin otomatik olarak çıkarılması işlemini gerçekleştirmişlerdir. Araştırmacılar yaptıkları

çalışmada bina cephelerine ait iki farklı veri seti içinden RANSAC algoritmasını kullanarak mevcut tüm düzlem yüzeyleri otomatik olarak çıkartmışlardır. Otomatik olarak çıkarılan düzlemlerin doğruluklarını araştırmak için otomatik olarak çıkarılan iki düzlem yüzey, ham nokta bulutu içinden elle de çıkarılmıştır. Elle ve otomatik olarak çıkarılan düzlemler karşılaştırıldığından her iki yöntemle çıkarılan düzlemlerin %98 oranında tutarlı olduğu görülmüştür. Yapılan karşılaştırma sonucunda RANSAC algoritmasının nokta bulutu içinden düzlem yüzeylerin çıkarılmasında etkin bir araç olduğu gözlemlenmiştir.

Bu tez çalışması kapsamında literatüre katkı amacıyla, RANSAC algoritması ile nokta bulutu içinden silindir, küre, düzlem ve koni yüzey sahip yüzeylerin çıkarılması gerçekleştirilecektir. Bu kapsamda;

- İlk önce RANSAC algoritması ile düzlem çıkarma işleminde, nokta normallerinin düzlem çıkarmaya etkisi araştırılacaktır,
- İkinci olarak, RANSAC algoritması ile model çıkarmada, nokta ile model arasındaki maksimum olabilecek eşik mesafesinin (t) laboratuvar ortamında elde edilen veriler ile araziden elde edilen verilerde nasıl değiştiği incelenecektir.
- Son olarak ise nokta bulutu içinden yüzey çıkarma işleminde karşılaşılan sorunlar değerlendirilecektir.

3. KURAMSAL TEMELLER

Yapılan tez çalışmasının kuramsal temeller bölümünde yersel lazer tarama teknolojileri, yersel lazer tarayıcılar ve lazer tarama sistemlerinden elde edilen veriler içinden, otomatik nesne çıkarımında kullanılan algoritmalar anlatılacaktır.

3.1. Yersel Lazer Tarama Teknolojisi

Son yıllarda mühendislik ölçmeleri alanında konumsal veri elde etme tekniklerinde oldukça büyük gelişmeler meydana gelmiştir. Bu gelişmelerden bir tanesi de kullanıcılara doğrudan ve otomatik 3 boyutlu veri elde olanağı sunan yersel lazer tarama (TLS: Terrestrial Laser Scanning) teknolojisinde olmuştur. TLS dolaylı mesafe belirleme ilkesine göre çalışmaktadır. Nesne yüzeyi üzerindeki bir nokta ile lazer tarayıcı arasındaki uzaklık veya menzil lazer ışınının nesneye çarpım gelmesi arasındaki zaman farkı ölçülmesi ile hassas bir şekilde belirlenebilmektedir (Reshetyuk, 2009).

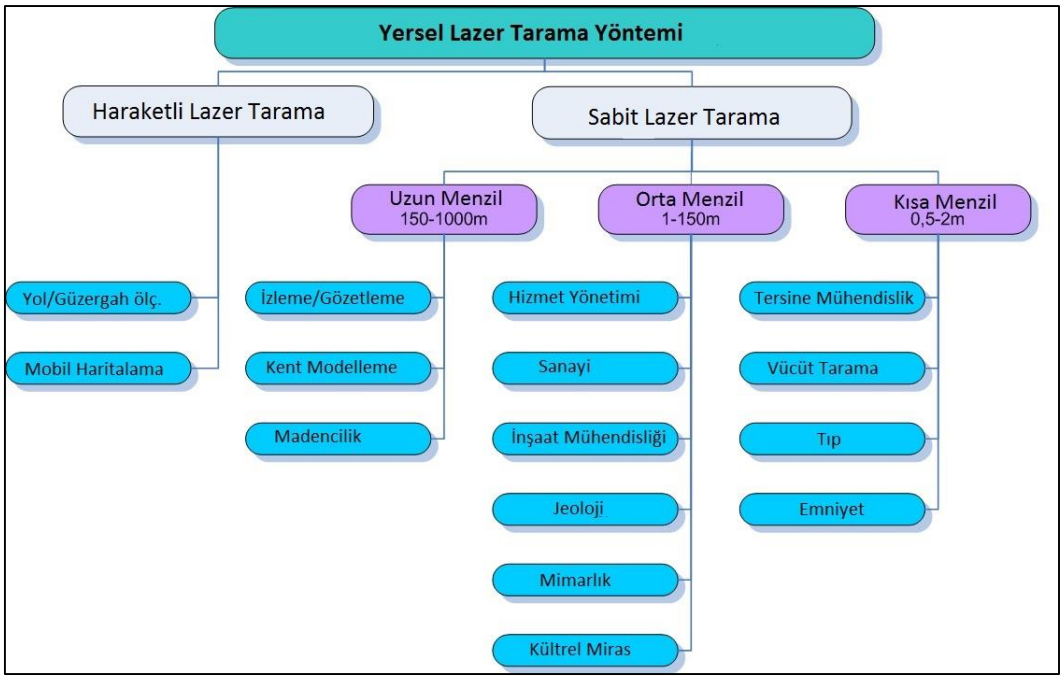
Lazer tarama yöntemi ile bir nesneyi taramak için tek konumdan nesneye ait binlerce nokta, mesafeye bağlı olarak hem dikey hem yatay yönlü olarak ölçülmektedir. Bu ölçüm sonucunda çıktı olarak ölçülen nesnenin yüksek detaylı görüntüsü elde edilmektedir. Bu yüksek detaylı görüntü milyonlarca yoğunlukta noktadan oluşan nokta bulutu olarak adlandırılmaktadır. Nokta bulutu içinde yer alan her bir noktanın tarayıcıya bağımlı olan 3 boyutlu koordinatları ve yansıyan lazer sinyalinin genliği (yoğunluk) kayıt edilmektedir. Günümüzde kullanımda bulunan, yüz metreye kadar ölçüm yapabilen birçok tarayıcı mevcuttur. Tarama işlemi hem sabit (statik) hem de haraketli platformlar kullanılarak gerçekleştirilebilmektedir. Günümüzde yersel lazer tarama tekniğinin uygulandığı alanlarda taranan objeler büyük ve kompleks şekillere sahiptirler. Bu objelerinin tamamına ait geometriyi elde etmek için farklı istasyonlarda bir dizi tarama işlemi yapmak gerekmektedir. Nesnelere ait bütünleşik bir veri elde edebilmek için farklı istasyonlardan elde edilen taramalar hassas bir şekilde birleştirilmeli (registration) ve bir jeodezik koordinat sistemine taşınmalıdır (konumlandırma). Verilerin jeodezik bir koordinat sistemine taşınması, TLS verilerinin diğer konumsal veriler ile birleştirilmesi için son derece önemlidir. Bununla birlikte birleştirilen

taramalardan elde edilen nokta bulutu kullanılarak, nesneye ait yüksek çözünürlüklü 3 boyutlu model üretmek mümkün olabilmektedir. Üretilen 3 boyutlu modeller farklı amaçlarda kullanılabilecek CAD formatına aktarılabilmektedir (Reshetyuk, 2009).

Yersel lazer tarama teknolojisinin diğer geleneksel ölçüm tekniklerine (Takeometri, GPS ve fotogrametri vd.) göre avantajları aşağıdaki gibi sıralanabilir (Reshetyuk, 2009).

- Nesne geometrisinin 3 boyutlu olarak doğrudan, hızlı ve detaylı bir şekilde elde edilebilmesi.
- Maliyetin belirgin şekilde azalması ve projenin hızlı bir şekilde tamamlanabilmesi,
- Geleneksel tekniklerin kullanılamadığı, heyelan alanlarının, ulaşılamayan ve karmaşık alanların uzaktan ölçülme imkânının olması,
- Veri toplama işleminde ışığa ihtiyaç duymaması,
- Tarama bütünlüğü ve kapsayıcılığı: Tarayıcının gördüğü her şey tek seferde taranır. Bundan dolayı kullanıcı eğer yeni bir veriye ihtiyaç duyarsa çalışma alanına geri dönmeden nokta bulutu üzerinden bu veriyi elde edebilir. Bu aynı zamanda sonuç ürün üzerindeki kullanıcı güvenini artırmaktadır.
- Verinin arşivlenme imkânın olması: Veri arşivlenerek gelecekte farklı amaçlar için kullanılabilir.

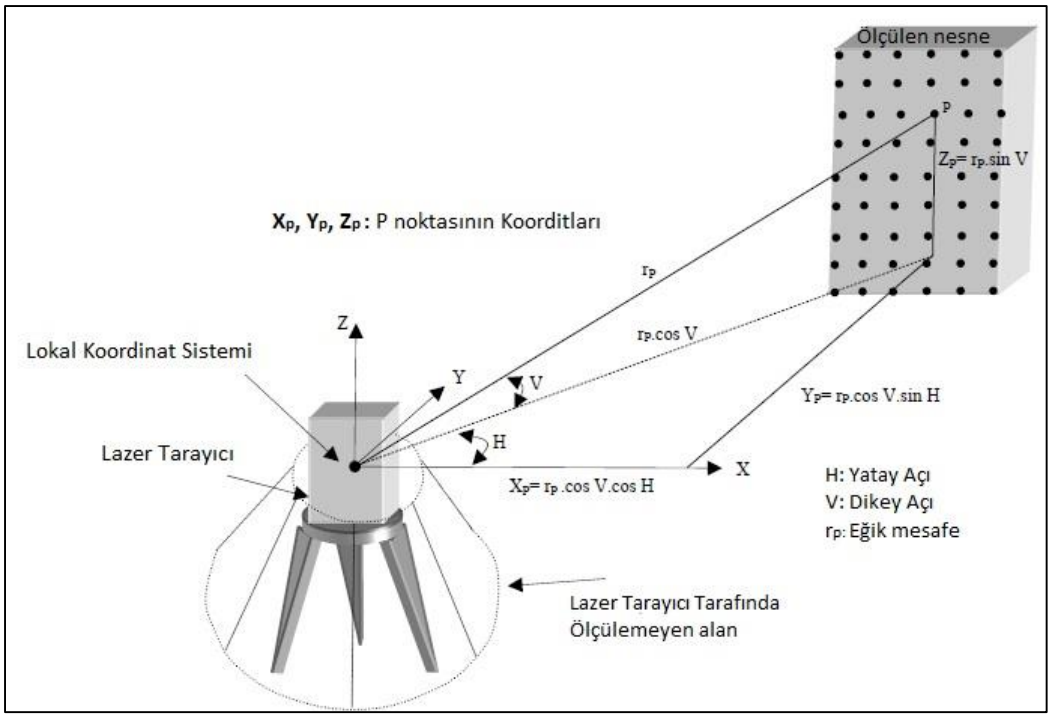
Bugün yersel lazer tarama mühendislik ölçmeleri tekniği olarak dikkate alındığında birçok farklı alanda, farklı amaçlara hizmet etmek için kullanılabilmektedir. Aynı zamanda her geçen günde kullanım alanı artmaktadır. Yersel lazer tarama yönteminin haraketli lazer tarama ve sabit lazer tarama olmak üzere iki farklı kolda incelemek mümkündür. Şekil 3.1'de yersel lazer tarama yönteminin tarayıcı ölçüm menzillerine göre kullanım alanları gösterilmiştir (Lerma Garcia, 2008).



Şekil 3. 1.Yersel lazer tarama yönteminin kullanım alanları (Lerma Garcia, 2008)

3.2. Yersel Lazer Tarayıcılar

Bir yersel lazer tarayıcı, dikey ve yatay alanda yer alan tüm noktaları motorize edilmiş şekilde ölçen bir elektronik uzunluk ölçer (total station) olarak tanımlanabilir. Ölçülen her bir nokta için, polar koordinatlar, eğik uzaklık ile birlikte noktaya olan yatay ve dikey açılar olarak kayıt edilmektedir. Ölçülen bu değerlere, nokta koordinatları, tarayıcı konumuna göre kolaylıkla hesaplanabilmektedir (Şekil 3.2) (Abdelhafiz, 2009).



Şekil 3. 2. Lazer tarayıcı ile nokta koordinatının ölçülmesi (Abdelhafiz, 2009)

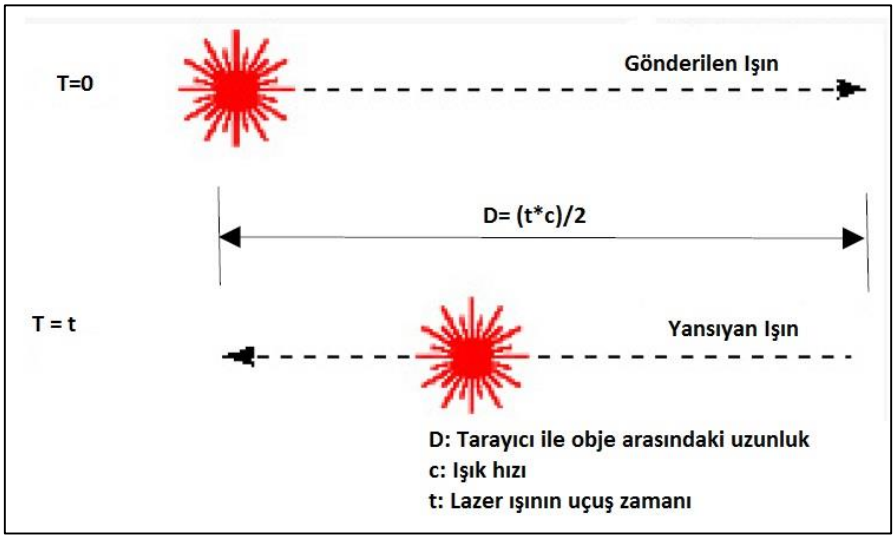
Şekil 3.2'de gösterildiği gibi bir nesne birkaç dakika içinde taranabilmektedir. Tarama işlemi dikey ve yatay yönde lazer tarayıcının görüş alanındaki objeleri tarayacak şekilde gerçekleşmektedir. Tarama sonucunda nesneye ait yüzbinlerce hatta milyonlarca ölçülen nokta verisi içeren 3 boyutlu nokta bulutu elde edilmektedir. Taranan her bir nokta en az üç koordinat (X,Y,Z) bilgisi sunmaktadır (Zogg, 2008). Nokta koordinatlarının yanında lazer tarayıcılar her nokta için bir yoğunluk (intensity) değeri de ölçmektedir. Yoğunluk değeri genellikle nokta bulutunun görsel analizinin desteklenmesi için kullanılmaktadır. Bununla birlikte yoğunluk değeri yüzey materyallerinin sınıflandırılması ve nokta bulutlarının birleştirilmesi gibi özel uygulamalarda kullanılabilmektedir (Abdelhafiz, 2009). Yersel lazer tarayıcılarla elde edilen nokta bulutu verisine, lazer tarayıcılara yerleştirilen dijital kameralar ile elde edilen renk bilgisi olan RGB (kırmızı, yeşil, mavi) değeri de atanabilmektedir. Nokta bulutuna renk bilgisi de atandıktan sonra bir nokta bulutu verisi X,Y,Z koordinat değerlerine ek olarak yoğunluk I, renk bilgisi, R, G, B değerleri ile temsil edilir hale gelmektedir.

3.2.1. Yersel Lazer Tarayıcıların Çalışma ilkeleri

Günümüzde farklı çalışma alanlarda farklı amaçlar için kullanılan çok sayıda yersel lazer tarayıcı mevcuttur. Lazer tarayıcıları ölçüm yöntemlerine göre üç farklı sınıfa ayırmak mümkündür. Bunlar lazer ışının gidiş-geliş zamanıyla işlem yapan (Uçuş zamanlı) tarayıcılar, faz farkı yöntemiyle işlem yapan lazer tarayıcılar ve Triangulasyon yöntemiyle işlem yapan lazer tarayıcılardır (Boehler 2002a).

Uçuş Zamanlı Lazer Tarayıcılar

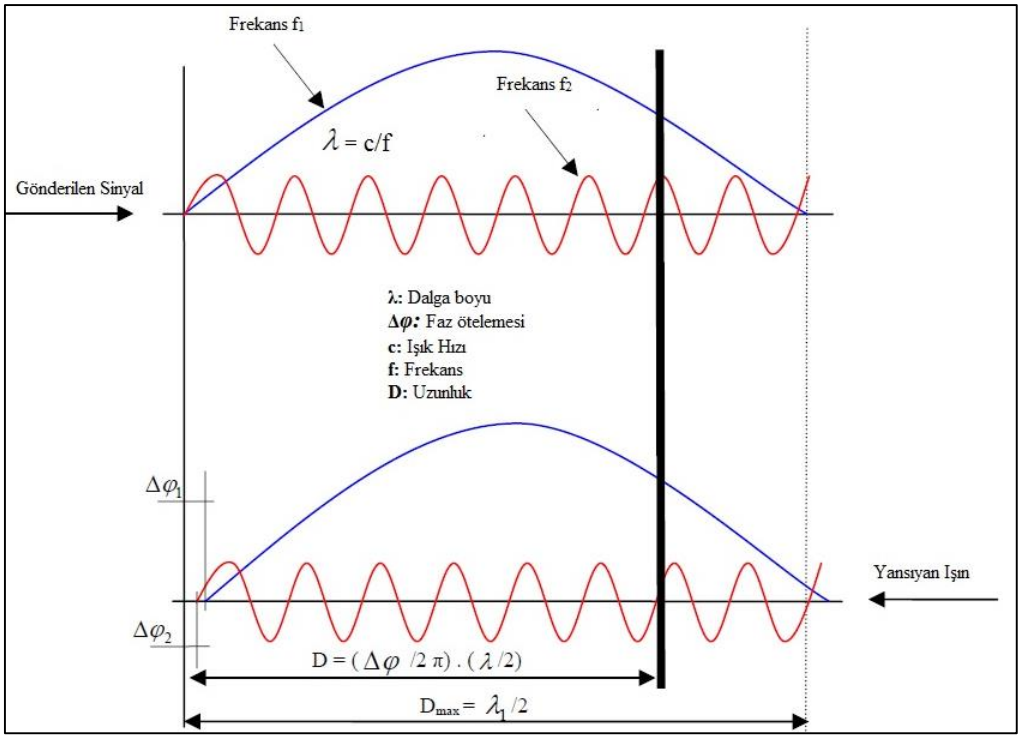
Bir lazer ışını nesneye gönderilir. Gönderici ile yüzey arasındaki uzunluk, sinyal iletimi ile alımı arasındaki uçuş zamanıyla ölçülür (Şekil 3.3). Bu prensip, total stationların çalışma prensibine benzemektedir. Total stationlar motorize bir şekilde tarama aleti olarak çalışmaya programlanabilir. Ancak aletin kütlesi nedeniyle eksen etrafındaki artan rotasyon basamakları yeterince hızlı değildir. Bununla birlikte sinyal süreci çok vakit alır ve açısal değerler kodlanmış çemberlerden zahmetli bir şekilde okunmaktadır. Bundan dolayı ölçüm oranları çok düşük olmaktadır. Tarayıcılar, lazer ışınının açısal sapması için küçük dönüş aletleri ve uzunluk hesaplaması için basit algoritmalar kullanırlar. Uzunluk ölçümlerinin tipik standart sapmaları, birkaç milimetre olmaktadır. 3B doğruluğu aynı zamanda, ışının açısal noktalama doğruluğundan etkilenir (Boehler, 2002a; Gümüş 2008).



Şekil 3. 3. Uçuş Zamanlı Tarama Yöntemi (Abdelhafiz, 2009)

Faz Farkı Yöntemiyle İşlem Yapan lazer Tarayıcılar

Faz farkı yönteminde, kesin bir dalga boyu (λ) ile gönderilen ve alınan sinyalin arasındaki faz ötelemesi ($\Delta\varphi$) belirlenmektedir (Şekil 3.4). Daha sonra ihtiyaç duyulan mesafe faz ötelemesine bağlı olarak ($D = (\Delta\varphi/2pi) * (\lambda/2)$) formülü ile hesaplanmaktadır. Kesin bir frekans ile ölçülebilen maksimum menzil, dalga boyu frekansının yarısıdır. Yüksek frekans ile yapılan ölçümlerde uzunluk, kısa mesafeli ölçümlerde, çok hassas bir şekilde ölçülebilmektedir. Faz farkı tarayıcılar ile yapılan tarama hızı uçuş zamanlı tarayıcılara göre daha hızlı olmasına rağmen, faz farkı ile işlem yapan tarayıcılardan elde edilen sonuç nokta bulutu uçuş zamanlı lazer tarayıcılar ile elde edilen nokta bulutunda daha fazla gürültü içermektedir. Bunun yanında uçuş zamanı ilkesine göre işlem yapan tarayıcıların ölçüm menzili faz farkı yöntemine göre işlem yapan lazer tarayıcılardan daha fazladır (Abdelhafiz, 2009).



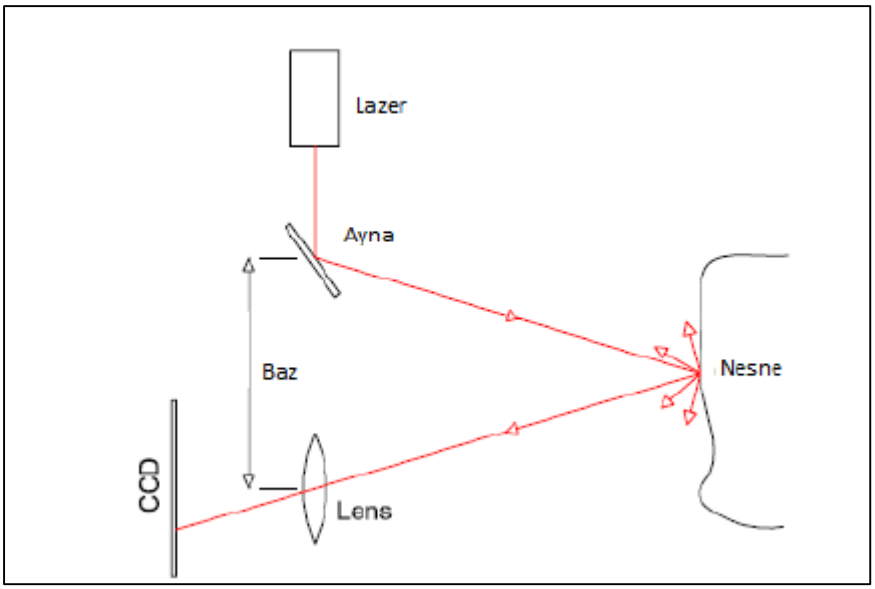
Şekil 3. 4. Faz Farkı yöntemi (Abdelhafiz, 2009)

Triangulasyon Lazer Tarayıcılar

Üçgenleme metodu ile işlem yapan tarayıcılarda konum belirlemek için tek kameralı ve çift kameralı çözümler uygulanır.

Tek Kamera Çözümü

Bu tarayıcı basit bir ışın yayma düzeneği içerir ve nesne üzerinde belirlenmiş baz sonunda aniden yayılma açısı değişmektedir ve diğer yandan CCD kamera bu baz üzerindeki lazer ışınını saptamaktadır (Şekil 3.5). Yansıyan yüzeyin 3 boyutlu konumu bu CCD kamera tarayıcı ve nesne arasında oluşturulan üçgen problemi ile çözülmektedir. Nesne ve tarayıcı arasındaki mesafenin doğruluk derecesi mesafenin karesiyle orantılıdır. Baz uzunluğu değişemeyeceği için bu tip tarayıcılar kısa mesafe ve küçük nesneler için iyi sonuç sağlamakta ve lazer ışınının gidiş dönüş prensibiyle ölçüm yapan tarayıcılardan daha doğru işlem sağlamaktadırlar (Boehler, 2002a; Demir, 2005).

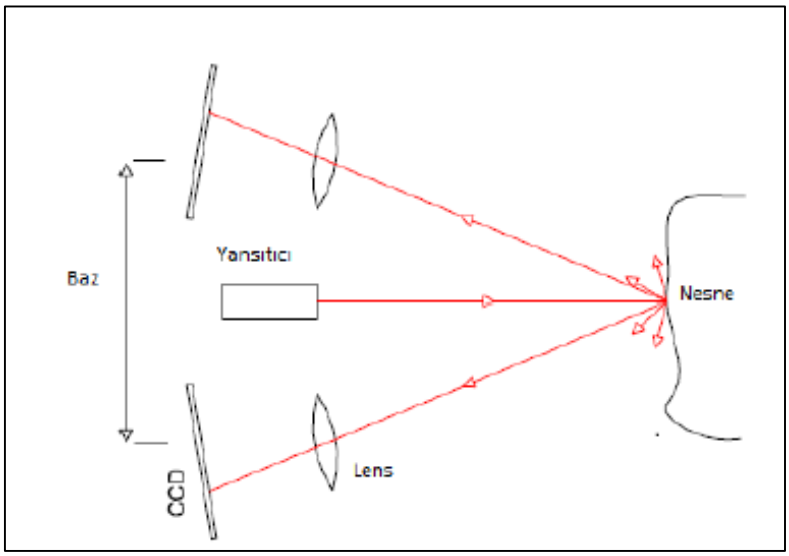


Şekil 3. 5. Tek kameralı çözüm (Boehler, 2002)

Çift Kamera Çözümü

Bu tip tarayıcılar iki CCD kamerası kullanırlar (Şekil 3.6). İncelenecek nokta ya da bölge, hiçbir ölçme fonksiyonu olmayan ayrı bir ışık projektörü ile üretilir. Projeksiyon, hareket eden şerit bölümlerinin bir ışık çizgisinden oluşur. Geometrik çözüm, tek kamera prensibiyle aynıdır ve aynı doğrulukta sonuç verir.

Bu tarayıcılar, yukarıda belirtilen tarama aletlerine bir alternatif olarak görülebilir (Boehler, 2002a; Gümüş, 2008)



Şekil 3. 6. Çift kameralı sistem (Boehler, 2002)

3.2.2. Yersel Lazer Tarayıcıların Genel Özellikleri

Kullanıcılar yersel lazer tarayıcıları karşılaştırırken tarayıcıların ölçüm hassasiyetini en önemli parametre olarak görmelerine rağmen, yapılacak projeye göre tarayıcılarda dikkate alınması gereken çok sayıda diğer karakteristik özellikler vardır. Bu özellikleri aşağıdaki gibi sıralamak mümkündür (Boehler, 2002a; Demir, 2005)

Hız: Yüksek çözünürlük için zamana bağlı olarak nokta taranabilmektedir. 100 nokta/saniye ile 1000 nokta/saniye kabul edilebilir normal hız sayılmaktadır (Boehler, 2002a; Demir, 2005).

Çözünürlük ve Işın Boyutu: Nesne çözünürlüğü teorik olarak lazer ışınının açısal çözünürlüğüne ve yansıyan ışının nesne üzerindeki alanına bağlıdır. Yüksek çözünürlüğün istendiği durumlarla lazer ışınının sağlayabildiği en iyi odaklama kabiliyeti dikkatlice saptanmalıdır (Boehler, 2002a; Demir, 2005).

Alım Uzaklığı Sınırlamaları ve Radyasyon Etkisi: Lazer tarayıcılar için verilen alım uzaklığı özelliklerinin pek çok parametreye bağlı olduğu gözden kaçırılmamalıdır. Bunlar; nesne yüzeyinin yansıma özelliğine, doğrudan güneş ışını almalarına ve ek olarak yansıyan güneş ışınına, nesne üzerindeki yapay radyasyona,

nesne yakınındaki radyasyon kaynaklarına bağlıdır. Genel olarak faz farkı prensibini kullanan tarayıcılarda CCD üzerinde sinyal saptanması ve faz farkı ölçümleri daha duyarlı olmasına karşın ışın zamanı prensibini kullanan tarayıcılar nispeten daha kuvvetlidir ve gece ölçüme olanak sağlamaktadırlar (Boehler, 2002a; Demir, 2005).

Görüş Alanı: Motorlu dönüş sistemleri bulunmayan tarayıcılarda FOV (görüş açısı) sınırlayıcı bir unsurdur. Genellikle 40 x 40 derecelik bir alanda işlem yapabilmektedirler (Boehler, 2002a; Demir, 2005).

Kayıt Araçları: Her tarama işlemi farklı bir konumdan gerçekleştirilmişse bunların tek bir koordinat sisteminde kaydı yapılarak bütünleştirilmesi gerekmektedir. Nesne üzerinde bazı hedef noktaları tarama yazılımlarında kolayca saptanıp bu işlem uygulanabilmektedir. Bazı sistemler kendi özel hedef noktalarını kullanmaktadırlar. Bu hedef noktaları aynı zamanda takeometrik ve fotogrametrik hedef noktaları olarak da uygundur (Boehler, 2002a; Demir, 2005).

Kameralar: Çoğu uygulama nesne üzerinde doku (texture) bilgisini içermektedir. Görüntülerin model üzerine uygulanmasıyla gerçekçi modeller sağlanabilmektedir. Bazı tarayıcılar geri dönen yansıma yoğunluğunu da ölçmektedirler. Bazıları ise doku haritalaması için yeterli kameraya sahip değillerdir. Üçgenleme prensibiyle çalışan tarayıcılarda kameranın ışın konumunu bulup belirlemek için doku eklemek uygun olmamaktadır. Yüksek kaliteli görüntü sağlamak için bugün için bir kamerayı tarayıcıya bağlamak uygun çözüm olarak görünmektedir. Bu durumda kamera ve tarayıcının, ilgili konumları tarama sonuçları ve görüntülere dayalı olarak kalibre edilmektedir (Boehler, 2002a; Demir, 2005).

Taşıma Kolaylığı: İdeal olarak tarama sistemi taşınabilir ve küçük olmalıdır. Fakat günümüzde çoğu sistem oldukça ağırdır. Özellikle yerleşim yerlerinden uzakta kültürel nesne uygulamalarında güç birimlerinin de birlikte taşınması önemli bir sorun olmaktadır (Barber, 2001; Demir, 2005).

Tarayıcılar Arasında Temel Yapısal Farklılıklar: Yersel lazer tarayıcılar arasında meydana gelen temel farklılıkları aşağıdaki gibi sıralamak mümkündür (Demir, 2005).

• Kamera benzeri tarayıcılar ve panoramik tarayıcılar

• Mesafe spektrum (min 1-2 m ve 25-800 m max)

• Hassasiyet (6mm-25mm)
• Hesaplama ve modelleme için sağlanan yazılımın verimi
• Ekstra bilgi (radyometrik çözünürlük) sağlıyor mu?

3.2.3. Yersel Lazer Tarayıcı Sistemlerinin Bileşenleri

Büyük nesnelerin taranması işleminde kullanılan yersel lazer tarayıcılar ekipmanları ile birlikte bütünleşik bir sistem olarak ele alınmaktadır. Genel olarak sabit bir yersel lazer tarayıcını bileşenlerini aşağıdaki gibi sıralamak mümkündür (Şekil 3.7) (Ergün, 2011).

- Tarayıcı Ünitesi
- Kontrol Ünitesi
- Güç Ünitesi
- Ayaklık, tripod vs.
- Hedefler

Şekil 3.7. Yersel lazer tarayıcıların sistem bileşenleri (Ergün, 2011)

Yersel lazer tarayıcı sisteminin, tarayıcı ünitesi geleneksel ölçme aletlerine göre daha büyüktür. Bundan dolayı güç ünitesi ayrıdır. Tarayıcının kontrolü genellikle dizüstü bir bilgisayar ile sağlanmaktadır. Tarayıcıların elektrik gücü

genellikle tarayıcı için özel tasarlanmış bir batarya ile sağlanmaktadır. Bu batarya bittiğinde araba aküsü, elektrik gibi harici güç üniteleri kullanılabilmektedir. Eğer lazer tarayıcı ile tam bir gün tarama işlemi gerçekleştirilecekse dizüstü bilgisayar içinde yedek bir bataryanın olması gerekmektedir. Bazı durumlarda taşınabilir bir jeneratör daha kullanışlı olabilmektedir. Tarayıcı genellikle ölçüm işlemleri için özel tasarlanmış bir üçayak üzerine kurulmaktadır. Tarayıcının üzerine kurulacağı altlık sistem tekerlekli bir sistem olarak da tasarlanabilir. Lazer tarayıcılar ilk geliştirildiklerinde iki kişinin kaldırabileceği ağırlık ve boyuta sahip cihazlardı. Ancak günümüzde hemen hemen hepsi tek kişinin kaldırıp taşıyabileceği kadar hafif hale gelmiştir. Gelecekte geliştirilecek lazer tarayıcı sistemleri daha küçük ve hafif olacaktır (Barber 2001).

3.2.4. Yersel Lazer Tarayıcıların Sınıflandırılması

Yersel lazer tarayıcılar için bir sınıflandırılma yapılması oldukça zor bir iştir. Çünkü sınıflandırma için ölçüm prensipleri (üçgenleme, uçuş zamanlı, faz vb.) veya sahip oldukları teknik özellikler gibi birçok sınıflandırma olasılığı vardır. Genel olarak tüm uygulamalarda kullanılabilecek tek bir lazer tarayıcı yoktur. Bazı lazer tarayıcılar, orta menzilli (100 m'ye kadar) ve iç mekân kullanımı için uygun olurken, bazı lazer tarayıcılar uzun menzilli (birkaç 100 metre) ve dış mekân uygulamaları için uygundur. Bunların yanında yüksek hassasiyette ve çok kısa menzilli (birkaç metre) ölçüm yapan yersel lazer tarayıcılar mevcuttur. Yapılacak uygulamaya göre uygun lazer tarayıcının seçilmesi uygun görülmektedir (Fröhlich ve Mettenleiter, 2004).

Günümüzde en çok kullanılan tarayıcılar uçuş zamanlı yersel lazer tarayıcılardır. Uçuş zamanlı lazer tarayıcılar ile birkaç yüz metrelik mesafelerde ölçüm yapılabilmektedir. Uçuş zamanlı lazer tarayıcıların yanında faz farkı prensibine göre çalışan yersel lazer tarayıcılar orta menzillik ölçüm işlemlerinde sıklıkla kullanılmaktadır. Ancak bu tarayıcıların menzili ise 100 metre ile sınırlıdır. Üçgenleme prensibine göre çalışan lazer tarayıcılar ise yakın mesafeli (birkaç metre) lazer tarayıcılardır. Bu lazer tarayıcılar yüksek mesafe ölçüm hassasiyetine sahip olduklarından dolayı genellikle sanayi uygulamaları ve tersine mühendislik işlemlerinde kullanılmaktadır (Fröhlich ve Mettenleiter, 2004).

Günümüzde birçok firma tarafından üretilen ve değişik özellikleri bulunan yersel lazer tarayıcı sistemleri bulunmaktadır. Bu lazer tarayıcılardan bazıları Şekil 3.8'de gösterilmiştir.

Şekil 3.8. Çeşitli yersel lazer tarayıcı sistemleri (Gümüş, 200)

Çizelge 3.1'de yersel lazer tarayıcıları ölçüm yöntemlerine göre sınıflandırılması gösterilmiştir. Bu sınıflandırma çok genel bir sınıflandırmadır (Fröhlich ve Mettenleiter, 2004).

Çizelge 3.1.Ölçüm yöntemlerine göre lazer tarayıcıların sınıflandırılması

Ölçüm Yöntemi	Menzil (M)	Hassasiyet (Mm)	Üretici Firma
Uçuş zamanlı	< 100	< 10	Callidus, Leicai Mensi, Optech, Riegl
	< 1000	< 20	Optech, Riegl
Faz Farkı	< 100	< 10	IQSun, Leica VisImage, Zoller+Fröhlich
Triangulasyon	< 5	< 1	Mensi, Minolta

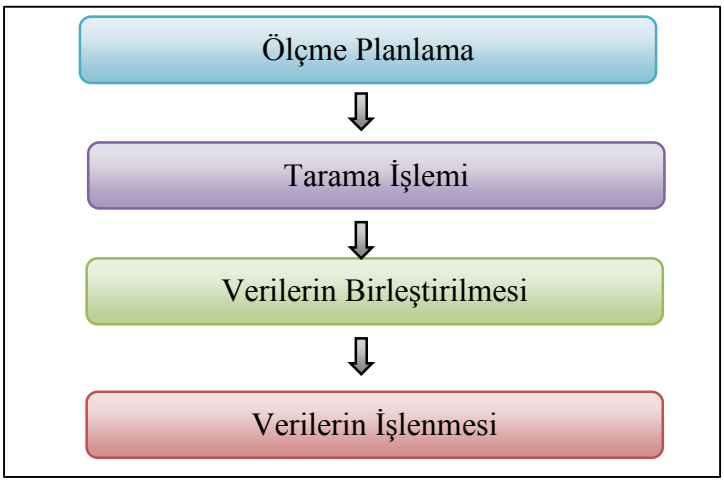
Sınıflandırma işlemi tarayıcının teknik özelliklerine göre de yapılabilir. Lazer tarayıcıların sahip oldukları teknik özellikleri aşağıdaki gibi sıralamak mümkündür (Fröhlich ve Mettenleiter, 2004).

- Tarama hızı, lazer ölçüm cihazının örnekleme oranı,
- Görüş açısı,
- Konumsal çözünürlüğü,
- Menzil ölçme hassasiyeti,
- Tarayıcı üzerine monte edilen diğer cihazlarla (fotoğraf makinası, GPS vb.) birleşimi,

EK-1'de bazı yersel lazer tarayıcıların teknik özellikleri gösterilmiştir.

3.3. Yersel Lazer Tarama Ölçme Prosedürleri ve Verilerin İşlenmesi

Bu bölümde yersel lazer tarayıcılar kullanılarak yapılan ölçüm işlemleri ve elde edilen verilerin değerlendirilmesi hakkında bilgi verilecektir. Yersel lazer tarama yöntemine göre verilerin elde edilmesi ve işlenmesinde izlenen işlem adımları Şekil 3.9'da gösterilmiştir.



Şekil 3.9. Yersel lazer Tarama işlem adımları

3.3.1 Ölçme Planlama

Nesnelerin taranması işlemine geçmeden önce tarama işleminin planlanması gerekmektedir. Planlama aşamasında tarama yapılacak istasyonların

konumları ve sayısı, taramanın konumsal çözünürlüğü ve taramanın referans sisteminin belirlenmesi gerekir (Riveiro, 2011).

Tarama yapılacak istasyonların konumları taranacak nesnenin tamamını ve tüm detayları kapsayacak şekilde yeterli sayıda belirlenmelidir. Bu istasyonların konumları belirlenirken bir istasyondan gölgede kaldığı için taranamayan bölgelerin taraması diğer istasyonlardan taranacak şekilde belirlenmeli ve taranacak nesne yüzeyinde eksik alan bırakılmamasına özen gösterilmelidir (Riveiro, 2011).

Taramanın konumsal çözünürlüğü, taramanın gerçekleştirileceği cihazın açısal çözünürlüğü ve cihazın konumsal hassasiyetine bağlıdır. Konumsal hassasiyet belirli bir mesafe için esas olan ve o mesafeden sonra tarayıcı ile obje arasındaki fark arttıkça değişen hassasiyettir. Örneğin Riegl 390i lazer tarayıcı cihazı için bu değer 50 metreye kadar 6 mm'dir. Açısal çözünürlük istasyon noktasından taranacak yüzey üzerindeki en uzak mesafeye göre seçilmelidir. Mesafe artırıldıkça açısal çözünürlükte artırılmalıdır (Riveiro, 2011).

Tarama işleminde farklı noktalardan elde edilen nokta bulutlarının ortak bir koordinat sisteminde birleştirilmesi gerekmektedir. Farklı istasyonlardan elde edilen her tarama verisi, tarayıcı merkezli bir koordinat sistemine sahiptir (Altuntaş, 2008). Tarama işleminin planlanması aşamasında, elde edilecek nokta bulutunun hangi koordinat sistemi referans alınarak birleştirileceği belirlenmelidir. Bu referans sistemi jeodezik bir koordinat sistemi veya tarayıcı merkezli yerel bir koordinat sistemi olabilir. Eğer taramanın referans sistemi jeodezik bir koordinat sistemi olacaksa, tarama işleminde bağlantı noktası olarak kullanılan reflektörlerin koordinatları bu koordinat sistemine göre belirlenmelidir (Çömert, 2012).

3.3.2 Tarama İşlemi

Günümüzde kullanılan birçok yersel lazer tarayıcı farklı istasyonlardan elde edilen nokta bulutu verilerinin birleştirilmesi (registration) için yapay veya doğal hedefler kullanılmaktadır. Tarama işlemine geçmeden önce bu hedefler taranacak nesne üzerinde veya çalışma alanında uygun yerlere yerleştirilir. Hedefler yerleştirildikten sonra ölçme işleminde kullanılacak yersel lazer tarayıcı, planlama aşamasında belirlenen istasyon yerlerinden taramanın yapılacağı istasyon üzerine kurulur. Tarayıcının kurulum işleminden sonra tarayıcının güç ünitesi ve kontrol

ünitesi ile bağlantısı sağlanır. Tarayıcı ile kontrol ünitesi arasındaki bağlantı kablosuz, modem veya kablolar aracılığı ile sağlanabilir. Hedef ve tarayıcının kurulum işleminden sonra tarama işlemine geçilir (Lerma Garcia, 2008).

Tarama işlemi, kontrol ünitesinde kurulu olan yazılım ile gerçekleştirilir. Tarama işleminin ilk aşamasında taramanın yapılacağı alanın yazılıma tanıtılması gerekmektedir. Bu işlem için farklı tarayıcılar farklı yöntemler kullanmaktadır. Bazı lazer tarayıcılarda taranacak alan tarayıcıya yerleştirilen bir video kamera aracılığı ile seçilirken, bazılarında ise tarayıcı tarafında çekilen bir fotoğraf üzerinden seçilebilmektedir. Bazı yersel lazer tarayıcılarda ise taranacak alanın belirlenmesi için düşük çözünürlüklü bir tarama işlemi gerçekleştirilmekte ve taranacak alan bu tarama verisi üzerinden seçilmektedir. Taranacak alan seçildikten sonra tarama parametreleri (çözünürlüğü, tarama mesafesi vb.) yazılıma girilir. Parametrelerin girilmesinden sonra tarama işlemi başlatılır. Tarama esnasında kontrol yazılımı üzerinden taranan alan eş zamanlı olarak izlenebilir. Tarama bitiminde veriler proje dosyasına kayıt edilmektedir (Lerma Garcia, 2008).

Nokta bulutlarının birleştirilmesinde yapay veya doğal hedefler kullanılacaksa, bu hedeflerin orta noktalarının koordinatlarının hassas bir şekilde belirlenmesi için en yüksek çözünürlükte hedef tarama işlemi gerçekleştirilir (Lerma Garcia, 2008).

3.3.3. Verilerin Birleştirilmesi

Nesnelerin tamamını temsil eden 3 boyutlu nokta bulutu elde etmek için birden fazla istasyondan tarama işleminin gerçekleştirilmesi gerekmektedir. Her bir istasyondan elde edilen nokta bulutları farklı koordinat sistemine sahiptir. Taranan nesneye ait tam bir veri sunumu yapabilmek için farklı koordinat sistemine sahip nokta bulutlarının ortak bir koordinat sisteminde birleştirilmesi gerekmektedir. Nokta bulutlarının birleştirilmesi için geçmiş yıllarda çok sayıda akademik çalışma yapılmıştır. Yapılan çalışmalar kapsamında (Besl ve McKay, 1992; Campbell ve Flynn, 2001; Chen ve Medioni, 1992; Gruen ve Akça, 2005; Elkharachy, 2008; Masuda ve Yokoya, 1995; Park ve Subbarao, 2003; Akça, 2003; Ripperda ve Brenner, 2005; Zhang, 1994; Al-Manasir ve Fraser, 2006; Akça, 2007; Akça ve Gruen, 2008; Wendt, 2008) elle, yarı otomatik ve otomatik olmak üzere farklı

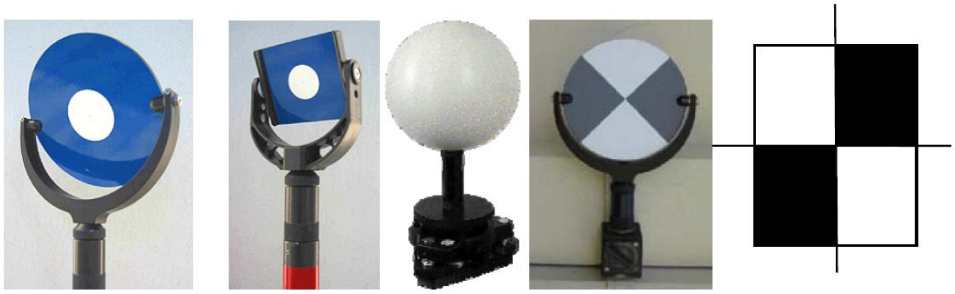
birleştirme yöntemleri geliştirilmiştir. Yapılan tüm bu çalışmalar temel alınarak nokta bulutu birleştirme yöntemleri üç farklı sınıfa ayrılabilir. Bunlar (Abdelhafiz, 2009);

- Hedef temelli birleştirme,
- Nesne temelli birleştirme,
- Görüntü temelli birleştirme,

şeklindedir.

Hedef Tabanlı Birleştirme

Hedef tabanlı birleştirme yönteminde iki nokta bulutunun birleştirilmesi, her iki nokta bulutu içinde ortak olan en az 3 ilişkili nokta aracılığı gerçekleştirilebilir. Üç ortak noktadan fazla noktanın ortak olması durumunda en küçük kareler ilkesine göre dengelemeli birleştirme işlemi gerçekleştirilebilir. Bu noktalar doğal veya yapay hedefler olabilir. Doğal hedefler elle belirlenirken, yapay hedefler yazılımda yer alan algoritmalar aracılığı ile otomatik olarak belirlenebilir. Hedeflerin otomatik belirlenmesi işlemi küre, siyah beyaz dairesel hedefler gibi özel şekle sahip hedeflerin geometrilerine göre belirlenebilmektedir. Şekil 3.10'da lazer tarama işlemlerinde kullanılan farklı şekle sahip yapay hedefler gösterilmiştir (Abdelhafiz, 2009).



Şekil 3.10. Farklı şekle sahip yapay hedefler (Abdelhafiz 2009)

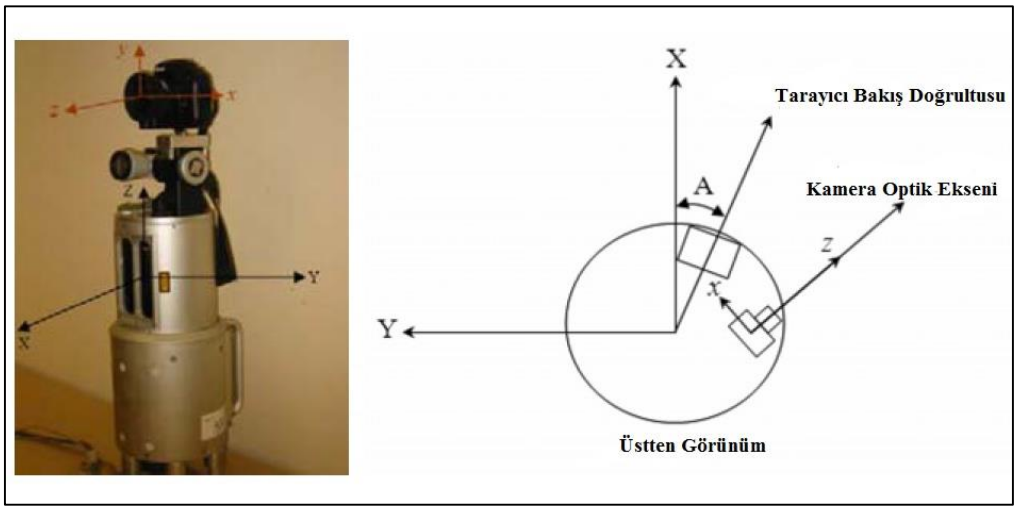
Nesne Tabanlı Birleştirme

Nokta bulutlarının nesne tabanlı olarak birleştirilmesinde kullanılan en başarılı ve en temel yaklaşım (Besl ve McKay, 1992; Chen ve Medioni, 1992; Zhang, 1994) tarafından önerilen iteratif en yakın nokta (Iterative Closest Point:

ICP) algoritmasıdır. ICP algoritması genel olarak iki temel adımdan oluşmaktadır. İlk adımda, iki taramanın kesiştiği alanda ilişkili aday noktalar belirlenir. İkinci adımda ise her iki taramada tespit edilen nokta setleri arasında rijit bir dönüşüm yapılarak noktalar arasındaki mesafe azaltılır. Mesafe azaltım işlemi belirlenen bir eşik değerinin altına inene kadar iteratif olarak devam ettirilir. Bu şekilde nokta bulutlarının birleştirilmesi sağlanır. ICP algoritmasında nokta çiftleri arasındaki mesafenin iteratif olarak belirlenmesi için farklı yaklaşımlar kullanılmaktadır. Bununla birlikte nesne tabanlı birleştirme amaçlı otomatik nesne eşleştirme teknikleri günümüzde aktif bir araştırma konusudur (Abdelhafiz, 2009).

Görüntü Tabanlı Birleştirme

Bu yöntem gelişmiş bazı lazer tarayıcılara monte edilmiş kamera kullanımı ile gerçekleştirilmektedir (Şekil 3.11). Yöntemin temeli kamera ve tarayıcı arasında değişmez bir ilişki olduğunu varsayarak rölatif yöneltmelerin hesaplanmasına dayanmaktadır. Sistemden kullanılan kameranın ilk önce laboratuvar ortamında kalibrasyonunun yapılması gerekmektedir. Daha sonra lazer tarayıcılardan elde edilen taramalar fotoğraflar arasındaki ilişkiye bağlı olarak birleştirilebilir. Fotoğrafların yöneltme parametreleri geleneksel fotogrametrik yöntemlerle hesaplanmaktadır (Abdelhafiz, 2009).



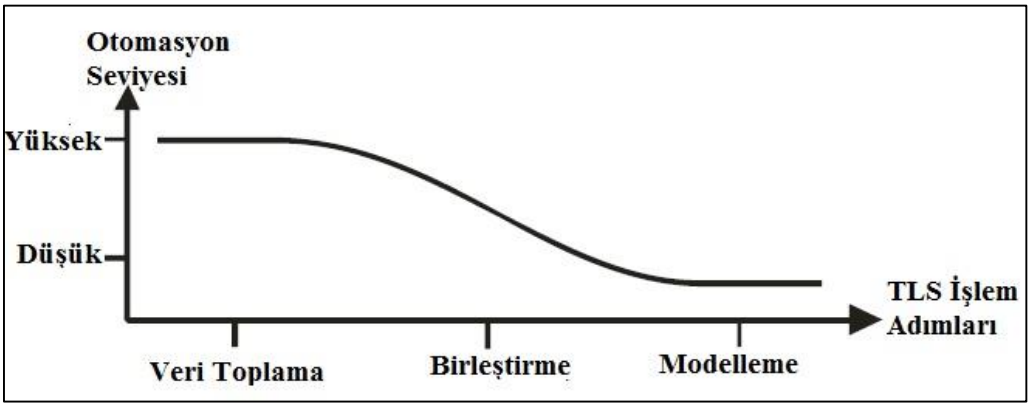
Şekil 3.11. Tarayıcı ve kamera koordinat sistemi (Abdelhafiz 2009)

Yöntem, sadece fotoğraflar arasında bindirme olmasını gerektirirken taramalar arasında herhangi bir bindirme olması şartını gerektirmemektedir. Al-

Manasir ve Fraser (2006) görüntü temelli birleştirme işlemini, iteratif en yakın nokta algoritması gibi mevcut yersel lazer tarayıcı verilerinin birleştirilmesine alternatif bir yaklaşım olduğunu belirtmişlerdir. Ayrıca yöntemde birleşim noktaların dijital görüntüler üzerinden doğrudan seçilmesi, nokta bulutu içinden seçilmesinden daha avantajlıdır. Çünkü görüntü üzerinden nokta seçim işlemi daha kolay ve daha az hata ile yapılabilmektedir (Abdelhafiz, 2009).

3.3.4. Verilerin İşlenmesi

Yersel lazer tarayıcılar ile elde edilen verilerin işlenmesi, lazer tarama yönteminde en fazla zaman alan işlemlerden birisidir. Bununla birlikte veri işleme aşaması, genel olarak elle yapılan bir işlem olarak karşımıza çıkmaktadır. Şekil 3.12'de yersel lazer tarama yönteminin işlem adımlarının otomasyonunu gösteren bir grafik sunulmuştur (Lerma Garcia, 2008).



Şekil 3.12. Yersel lazer tarama işlem adımlarının otomasyon seviyesi (Lerma Garcia, 2008)

Yersel lazer tarayıcılar ile elde edilen nokta bulutlarının işlenmesi, birleştirilen ham nokta bulutu verilerinden sonuç ürünlerin oluşturulması anlamını taşımaktadır. Nokta bulutlarının işlenmesi ile birçok sonuç ürün elde edilebilmektedir. Genel olarak nokta bulutlarından en fazla üretilen ürünler nokta bulutunun görsel olarak sunulması, standart 3 boyutlu CAD çizimleri (plan, kesit, yükseklik) ve 3 boyutlu modelleme çalışmalarıdır. Nokta bulutu verilerinden anlamlı bilgiler çıkarmak veya kullanılabilir sonuç ürünler elde etmek için belirli ön işlemlerin yapılması gerekir. Bu ön işlemler gereksiz verilerin ayıklanması,

gürültü verilerinin filtrelenmesi gibi işlemlerden oluşmaktadır (Lerma Garcia, 2008).

Tarama işlemi sonucunda elde edilen veri her biri X, Y, Z koordinatlarını ve nesnelerden yansıyan yoğunluk değerini içeren yoğun bir nokta bulutu verisidir. Bazı tarayıcılar ile elde edilen nokta bulutu verisi, renk değerlerini de içermektedir. Nokta bulutunun görsel olarak sunumu nokta bulutu içindeki tüm noktaları ekran üzerinde görüntülenmesi ile gerçekleştirilmektedir. Görüntüler ile nesneler hakkında ön bilgiler elde edilebildiği gibi görsel videolar da hazırlanabilmektedir. Ancak bunun yanında nokta bulutu içinde temel yapıların kullanıcı tarafından algılanmasında sorunlar yaşanabilmektedir (Lerma Garcia, 2008).

Nokta bulutu verilerini kullanarak standart 2 boyutlu CAD çizimi işlemi kültürel mirasın korunmasında rölöve ve restorasyon çalışmaları gibi alanlarda kullanıcılara çok büyük kolaylıklar sağlamaktadır. Nokta bulutundan CAD çizimi oluşturmak için birçok ticari yazılım paketi mevcuttur. Bu yazılım paketleri AutoCAD veya MicroStation gibi yazılımlara eklenti yapılabilmektedir. Yazılım paketlerinin sahip olduğu özel ara yüzler, kullanıcılara, standart CAD araçlarını kullanabilmeleri için yoğun nokta bulutunu bu programlara yüklenmesine olanak tanımaktadır. Çok kullanılan CloudWorx, Kubit PointCloud, LFM CAD link vb. yazılımlardır (Lerma Garcia, 2008).

Yersel lazer tarama yöntemi ile elde edilen nokta bulutu verilerinden, en çok üretilen ve birçok amaç için kullanılan ürünler, 3 boyutlu modellerdir. Modelleme çalışmaları için günümüzde geliştirilen birçok yazılım mevcuttur. Bu yazılımlarda taranan nesnelerin 3 boyutlu modellenmesinde nesnenin geometrik yapısına göre farklı yaklaşımlar uygulanabilmektedir. Eğer modellenecek obje karmaşık bir yapıya sahipse genellikle yüzey ağı modelleme (meshing) yöntemi uygulanmaktadır. Eğer taranan nesne basit bir geometrik şekli (düzlem, silindir, küre vb.) temsil ediyorsa geometrik temelli modelleme işlemi uygulanabilmektedir (Boulaassal, 2007).

Yersel lazer tarayıcılardan 3 boyutlu verilerin işlenmesinde kullanılan farklı türde birçok yazılım modülü vardır. Eğer yersel lazer tarayıcı verilerinin işlenmesi, verilerin elde edilmesinden sonuç ürünün elde edilmesine kadar bir bütün olarak

dikkate alınırsa, yazılım modülleri arasında aşağıdaki gibi kaba bir ayrım yapılabilir (Boehler 2002b; Demir 2005; Gümüş 2008).

- Tarayıcı kontrolü yazılımı,
- Nokta bulutunun düzenlenmesi yazılımı,
- Basit geometrik şekilleri, nokta bulutuna sabitleme yazılımı,
- Karmaşık yüzey modellerinin oluşturulması yazılımı,
- Doku ve görüntü ekleme yazılımı,
- Veri ve proje yönetimi yazılımı,

3 boyutlu tarama yazılımları, farklı amaçlar için geliştirilen modüllerin bütünleşik çalışarak oluşturdukları modüller topluluğudur. Yazılımların bazıları, belirli bir tarayıcıya özgüdür. Bazıları genel olarak 3 boyutlu verilerin değerlendirilmesi ve düzenlenmesi için, bazıları da 3B Modelleme çalışmaları için üretilmiştir. Her yazılımın kendine göre farklı özellikleri, avantaj ve dezavantajları vardır. Lazer tarama teknolojisinin gelişmesi ve modern ölçme teknikleri arasında yer almaya başlamasından itibaren, 3 boyutlu tarayıcıları için özel bazı bağımsız yazılım ürünleri geliştirilmiştir. Mevcut CAD ve 3 boyutlu modelleme yazılımları, ürün aktarım performanslarını ve yüksek boyutlardaki nokta verileri ile çalışabilme özelliklerini geliştirdikleri takdirde, lazer tarama teknolojisinde kullanılabilir (Gümüş, 2008).

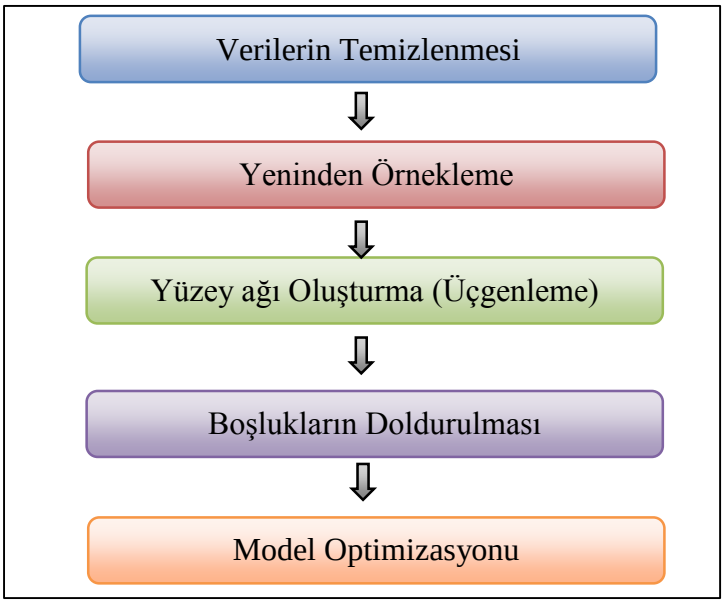
3.4. Yersel Lazer Tarayıcı Verilerinden 3 Boyutlu Model Oluşturma

Bu bölümde yersel lazer tarayıcılar ile taranan objelerin geometrilerine göre uygulanan 3 boyutlu modelleme yaklaşımları üzerinde durulacaktır. Bu yaklaşımlar yüzey ağı modelleme ve geometrik temelli modelleme olarak iki başlık altında ele alınacaktır.

3.4.1. Yüzey Ağı Modelleme

Yüzey ağı oluşturma karmaşık yüzeylerin geometrik tanımlamalarını elde etmek için kullanılan standart bir yöntemdir. Yöntem fotogrametri ve ölçme bilimi ile uğraşanlar tarafından topografik uygulamalarda sıklıkla kullanılmaktadır. Bu yöntemde iki boyutlu düzlem üzerine yükseklik verileri eklenerek 2.5 boyut adı

verilen üç boyutlu gösterim oluşturulmaktadır. Bazı tarayıcı yazılımları alternatif olarak 2.5 boyutlu yüzey ağı oluşturma modülünü içermektedirler (Boehler, 2002b; Demir, 2005). Yüzey ağı modelleme işlemi, yazılım tarafından otomatik yapılabildiği gibi, kullanıcı müdahaleleri ile de yapılabilmektedir. Yüzey ağı oluşturma işlemi günümüzde birçok yazılım tarafından yapılabilmektedir. Model oluşturma işlemi yazılım tarafından otomatik olarak yapıldığında objenin bazı bölümlerinde bozukluklar olabilmektedir. Bu bozukluklar kullanıcı tarafında düzeltilebilmektedir (Gümüş, 2008). Genel olarak yüzey ağı modelleme işlem adımları Şekil 3.13'de gösterilmiştir (Lerma Garcia, 2008).



Şekil 3.13. Yüzey ağı modelleme tekniğinin işlem adımları(Lerma Garcia, 2008)

Verilerin Temizlenmesi

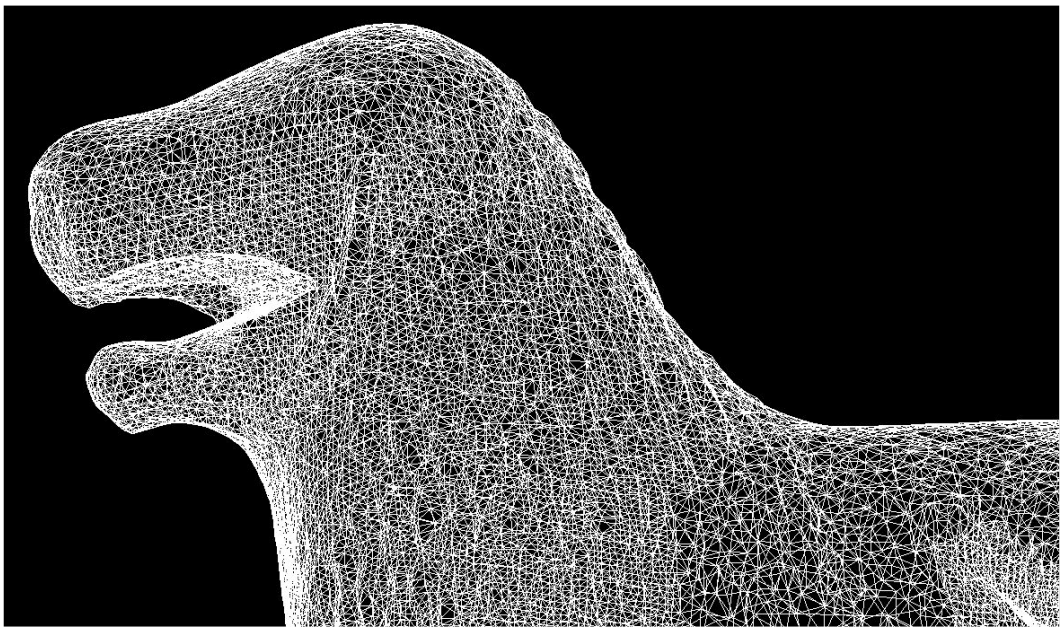
Yüzey ağı oluşturma işleminin ilk adımı verilerin temizlenmesi işlemidir. Yersel lazer tarayıcılardan elde edilen nokta bulutuna belirli filtreler uygulayarak tarama işleminden kaynaklanan gürültü, taranan objenin yüzeyinden bozuk yansımalardan kaynaklanan hatalı noktaların temizlenmesi gerekir. Noktaların gereksiz noktalardan temizlenmesi, üçgenleme aşamasında doğru bir üçgenleme yapılabilmesi için son derece önemlidir (Lerma Garcia, 2008).

Yeniden Örnekleme

Yersel lazer tarayıcılar ile obje taraması yapıldığında nesnelerin yüzeylerine ait çok sayıda nokta verisi elde edilmektedir. Elde edilen bu nokta bulutu verisi milyonlarca nokta içermektedir. Yüzey ağı modelleme işleminin üçgenleme aşamasında ise nokta yoğunluğunda dolayı çok sayıda üçgen oluşturulmaktadır. Oluşturulan bu üçgenlerin boyutları fazla yer kapladığından yazılımların çalışma performansları düşmektedir. Bundan dolayı nokta bulutu üzerine uygulanacak nokta örnekleme algoritmaları ile yüzeyin geometrisini koruyarak nokta azaltma işlemleri uygulanması gerekmektedir (Lerma Garcia 2008).

Yüzey Ağı Oluşturma /Üçgenleme

Nokta bulutlarında yüzey ağı oluşturmak için kullanılan farklı algoritmalar mevcuttur. Taranan objeleri temsil eden noktalar arasındaki bağlantı genellikle üçgenler aracılığı ile oluşturulmaktadır. Yüzey ağı oluşturma işleminde en yaygın kullanılan üçgenleme tekniği Delaunay üçgenleme tekniğidir. Deluanay üçgenleme kriterine göre oluşturulan bir yüzey ağı Şekil 3.14'de gösterilmiştir (Abdelhafiz 2009).



Şekil 3.14. Deluanay üçgenleme kriterine göre oluşturulan bir yüzey ağı (Abdelhafiz 2009)

Boşlukların doldurulması

Nesnelerin taranmasında çoklu istasyon kullanılması ile taranan nesne üzerinde boşluk alan kalmaması sağlanmaya çalışılmaktadır. Ancak yüzey ağı oluşturma aşamasında, veri eksikliğinde dolayı model üzerinde küçük boşluklar oluşabilmektedir. Bu boşlukların doldurulması için son yıllarda birçok algoritma geliştirilmiştir. Bu algoritmalar aracılığı ile enterpolasyona bağlı olarak yüzeyler oluşturulmaya çalışılmaktadır. Boşluk alanlar, düz olarak veya eğri temelli olarak doldurulabilmektedir (Lerma Garcia, 2008).

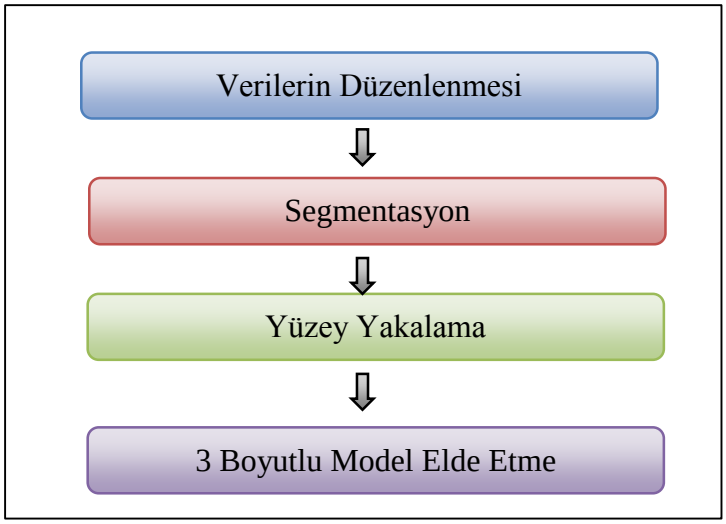
Model Optimizasyonu

Yeniden örnekleme aşamasında üçgen sayısını azaltmak için nokta sayısı düşürülmesine rağmen, üçgenleme aşamasından sonra modelin diskte kapladığı alanı azaltmak için, üçgen sayısının tekrardan azaltılması gerekmektedir. Bu işlem yüzey ağı seyreltilmesi olarak adlandırılmaktadır.

3.4.2. Geometrik Temelli Modelleme

Ham nokta bulutu verisi üzerinden elle nesne çıkarımı veya çizim işlemi hem zaman alıcı hem de hata miktarının fazla olabileceği bir seçenektir. Bununla birlikte yersel lazer tarayıcılar ile elde edilen nokta bulutu verisi proje dosyalarına düzensiz bir şekilde kayıt edilmektedir. Bu düzensiz nokta bulutu verileri bilgisayar disklerinde fazla yer kaplamaktadır. Bundan dolayı verilerden anlamlı bilgiler çıkarmak ve veri boyutu azaltmak için otomatik modelleme çalışmaları son derece önemlidir.

Yersel lazer tarayıcılar ile taranan nesneler, matematiksel olarak kolay tanımlanabilen basit geometrik yüzeylere sahipse, bu nesnelerin modellenmesinde geometrik temelli modelleme işlemi uygulanabilmektedir. Geometrik temelli modelleme çalışmaları ile ilgili literatür incelendiğinde modelleme işleminin genellikle kentsel alanda düzlem yüzeye sahip binaların otomatik modellenmesi üzerine olduğu görülmektedir. Geometrik temelli model oluşturmada standart bir iş akışı olmamakla birlikte genel olarak aşağıdaki gibi sıralanabilir (Şekil 3.15).



Şekil 3.15. Geometrik temelli model oluşturma iş akışı (Woo 2002)

Verilerin Düzenlenmesi

Modelleme işlemi için gerekli olan nokta bulutu verisi yersel lazer tarayıcılardan elde edildikten sonra, modelleme işleminde kullanılabilmesi için ön bir işleme tabi tutulması gerekmektedir. Bu ön işleme adımında veri içinde yer alan gürültülerin filtrelenmesi, nokta sayısının uygun bir yöntemle azaltılması ve istenmeyen noktaların temizlenmesi gibi işlemler yapılmaktadır (Woo, 2002).

Segmentasyon

Segmenetasyon kavramı için farklı araştırmacılar farklı tanımlar yapmışlardır. Tovari (2006) segmentasyonu değişmez benzerliklere sahip nesne yüzeylerinin geometrik devamlılıklarına göre bölütlenmesi olarak tanımlamıştır. Rabbani ve ark. (2006) göre segmentasyon, nokta bulutu içinde yer alan her bir noktanın etiketlendirme işlemidir ve bu işlemde aynı yüzeye veya bölgeye düşen noktalara aynı etiket değeri verilir (Sopkata, 2008).

Segmentasyon işlemi geometrik temelli modellemede nokta bulutlarının sınıflandırılması ve gruplandırılması için son derece önemli bir aşamadır. Bu aşamada nokta bulutu içinden sınıflandırılarak veya gruplandırılarak ayrıştırılan noktalar sahip oldukları geometrik özelliğe göre yüzey yakalama işlemine tabi tutulmaktadır.

Yüzey Yakalama

Yüzey yakalama aşamasında segmentasyon sonucu sınıflandırılan nokta bulutları için uygun geometrik yüzeyler oluşturulmaktadır. Yüzey oluşturma işleminde farklı yaklaşımlar izlenmektedir. Bazı araştırmacılar segmentasyon sonucu elde edilen sınıflandırılmış nokta bulutlarına uygun bir algoritma uygulayarak, segmentasyon sonucu elde edilen yüzeyin kenar noktalarını elde etmektedir. Elde edilen kenar noktalar birleştirilerek nokta formatından çizgi formatına geçiş yapılmaktadır. Yüzey yakalama işleminde kullanılan bir diğer yaklaşım ise, segmentasyon sonucu elde edilen sınıflandırılmış nokta bulutundan yüzey kenarlarının doğrudan elde edilmesidir.

3 Boyutlu Model Elde Etme

Geometrik temelli modelleme işleminin 3 boyutlu model elde etme aşamasında, yüzey yakalama sonucu elde edilen yüzeyler birleştirilerek nesnelere ait 3 boyutlu model üretimi işlemi gerçekleştirilmektedir. Modelleme işleminde farklı yüzeylerin birbiri ile birleştirilmesi ve boşlukta kalan yüzeylerin uygun yöntemlerle doldurulması oldukça önem arz etmektedir.

3.5. Yersel Lazer Tarayıcıların Segmentasyonunda Kullanılan Yöntemler

Yersel lazer tarayıcılardan elde edilen nokta bulutunun geometrik temeller üzerine segmentasyonunda birçok yöntem kullanılmaktadır. Yersel lazer tarayıcı verilerinin segmentasyonuna yönelik literatür incelendiğinde uygulanan yöntemlerin düzlem yüzey çıkarmaya yönelik olduğu görülmektedir. Segmentasyon yöntemlerini dört ana başlık altında incelemek mümkündür (Sapkota, 2008). Bunlar;

- Nesnelerin sınıflandırılması temeline dayanan segmentasyon,
- Yüzey geliştirme temeline dayanan segmentasyon,
- Model yakalama temeline dayanan segmentasyon,
- Hibrit segmentasyon tekniği,

3.5.1. Nesnelerin Sınıflandırılması Temeline Dayanan Segmentasyon

Bu yöntemde, nesneler ilk önce her bir noktanın geometrik ve radyometrik karakterleri temeline göre tanımlanır. Nesneler genellikle her bir noktanın konumunu, lokal olarak tahminlenen yüzey normallerini, yüzeyi en iyi yakalayan artık değerleri ve lazer noktalarının yansıma değerlerini içerir. Her noktanın n-nesneli değeri N-boyutlu nesne uzayı haritası üzerine inşa edilir. Daha sonra sınıflandırma işlemi nesne uzayında tanımlanır. Aynı sınıfa düşen noktalara nesne uzayında tek bir değer atanır. Bu yöntemin sonuçları nesne uzayında yapılan sınıflandırma metodu kadar seçilen nesneye ve türevlerine bağlıdır. Bağımsız nokta özellikleri genellikle lokal komşuluk içinde bulunan noktalar kullanılarak tanımlanır. Bu segmentasyon yöntemi aynı zamanda veri içinde gürültüye karşı duyarlı ve tanımlanan komşuluk ilişkisinden etkilenmektedir (Sapkota, 2008).

3.5.2. Yüzey Geliştirme Temeline Dayanan Segmentasyon

Bu yöntemde algoritma bir noktadan başlar ve kesin benzerlik ilkesine bağlı olarak komşu noktalar çevresinde gelişir. Vosselman ve ark. (2004) yüzey geliştirme yöntemini aşağıdaki gibi açıklamıştır (Sapkota, 2008).

Örneklem Yüzeyinin Tanımlanması

Bir örneklem yüzey bir düzlem yüzeyi en iyi şekilde yakalayan komşu noktalardan oluşmaktadır. Örneklem yüzey seçimi için bir grup bitişik noktalar tanımlanır ve bu noktaların bir düzlemi iyi bir şekil tanımlayıp tanımlamadığı test edilir. Eğer veri seti içinde önceden tanımlanmış eşik değeri sınırları içinde bir düzlem bulunursa, bu düzlem örneklem yüzey olarak kabul edilir, aksi takdirde başka bir nokta test edilir (Sapkota, 2008).

Örneklem Yüzeyin Geliştirilmesi

Örneklem yüzey seçildikten sonra örneklem yüzey içindeki her nokta düzleme düşen komşu noktaları bulmak için incelenir. Bu işlem temel olarak örneklem yüzeyi komşu noktalara doğru geliştirmek için yapılır. İncelenen noktalar belirlenen kriterler uygunsa geliştirilen yüzeye atanır. Gelişim yüzeyine eklenen her

bir noktadan sonra düzlem eşitliği güncellenir. Bir noktayı düzlem noktası olarak kabul etmek için aşağıda gösterilen kriterlerden birini veya daha fazlasını sağlaması gerekmektedir (Sapkota, 2008).

- **Nokta yakınlığı:** Veri seti içindeki noktalar ile örneklem yüzey arasındaki mesafe belirlenen bir mesafe sınırı içinde ise, noktalara geliştirme yüzeyine atanır.
- **Global düzlemsellik:** Segment içinde yer alan tüm noktalar için her defasında düzlem hesaplanır. Bir aday nokta için, nokta ile düzlem arasındaki öklid mesafesi belirli bir eşik mesafesi altında ise kabul edilir.
- **Yüzey düzgünlüğü:** Bu kriteri güçlendirmek için, nokta bulutu içinde yer alan her noktanın lokal yüzey normali tahmin edilir. Eğer noktanın lokal yüzey normali ve gelişen yüzeyin normali arasındaki açı belirli bir eşik değeri altında ise aday nokta gelişen yüzeye kabul edilir.

3.5.3. Model Yakalama Temeline Dayanan Segmentasyon

Bu yöntem insan yapımı nesnelerin; düzlem, silindir, küre gibi geometrik temelli bileşenlerine ayırma üzerine kurulmuştur. Bu yaklaşım nokta bulutu içinde yer alan geometrik şekilleri yakalamaya çalışmakta ve matematiksel olarak bir geometrik şekli tanımlayan noktalar ayrı bir segment olarak etiketlenmektedir. Hough dönüşümü ve Rastgele Örneklem Konsensüsü (Random Sample Consensus: RANSAC) bu yaklaşımın en önemli yöntemleri arasında yer almaktadır (Sapkota, 2008).

3.5.3.1. Hough Dönüşümü

2 boyutlu Hough dönüşümü tekniği (Hough 1962) geometrik temelleri tespit etmek için sayısal görüntü işleme alanında sıklıkla kullanılan bir tekniktir. Hough dönüşümü, sayısal görüntülerdeki doğruların ve diğer şekillerin (genellikle dairesel) tespiti işlemi için sıkça kullanılan bir görüntü analiz algoritmasıdır. Temel olarak siyah beyaz görüntülerdeki düzgün doğruları tespit etmek için geliştirilmiştir. Bunun yanı sıra, doğrusal olmayan düzensiz eğrilerin tespitinde de kullanılmaktadır (Temiz, 2011).

2 boyutlu Hough dönüşüm yöntemi, başlangıç olarak Öklid uzayında tanımlanan nokta setlerinin başka bir uzayda sunulmasıdır. Bu dönüşüm, özel geometrik temellere sahip noktaların birleştirilip belirlenmesine imkân tanımaktadır. Örneğin Bir OXY sisteminde bir doğrunun formülü Eşitlik 3.1'de gösterildiği gibidir (Tarsha-Kurdi, 2007).

$$Y = aX + b \tag{3.1}$$

Burada a ve b doğru parametrelerini temsil etmektedir.

Doğru parametre uzayında (O'ab) koordinatları (a,b) olan bir nokta ile sunulabilir. Tam tersi bir yolla, OXY uzayında (X_i, Y_i) ile temsil edilen bir nokta parametre uzayında (O'ab) Eşitlik 3.2 de gösterilen denklem gibi gösterilebilir (Tarsha-Kurdi, 2007).

$$b = -X_i a + Y_i \tag{3.2}$$

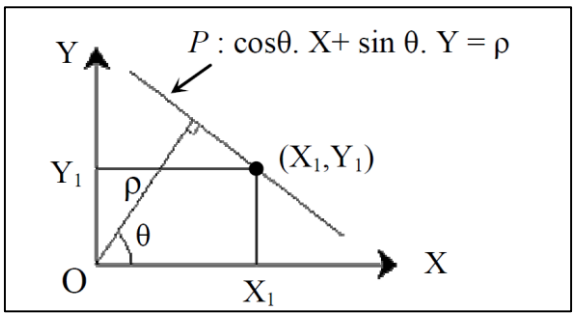
Burada X_i ve Y_i doğru parametrelerini temsil etmektedir.

M_1, M_2, … M_n (OXY) uzayında yer alan nokta setleri olduğu ve bu noktaların Eşitlik 3.3'de sunulan bir p doğrusuna ait olduğu varsayılsın. Bu noktaların her bir parametre uzayında bir doğru sunacaktır. Parametre uzayında (O'ab) bu doğruların kesişimi 2 boyutlu uzayda p doğrusunun parametrelerini sunan bir (a_1, b_1) noktasıdır (Tarsha-Kurdi, 2007).

Eğer doğru eşitliği X=değişmez forma sahipse, bu parametre uzayında (O'ab) temsil edilemez. Çünkü Y eksenin katsayısı 0'a eşittir. Bu problemi çözmek için doğrunun normalinin kullanılması önerilir (Eşitlik 3.3) (Tarsha-Kurdi, 2007).

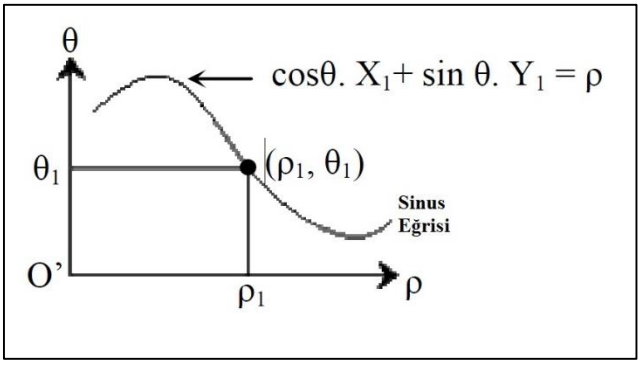
$$\cos\theta . X + \sin\theta . Y = p \tag{3.3}$$

Burada θ ve p orijinden geçen normale ait parametrelerdir (Şekil 3.16).



Şekil 3.16. 2 boyutlu uzayda bir doğru ve doğru normalinin gösterimi (Tarsha-Kurdi, 2007)

θ ve p bir doğru için değişmezdir. Bu durumda parametre uzayı (O'θp) dir. Bundan dolayı, 2 boyutlu uzayda bir (X_1, Y_1) noktası parametre uzayında bir sinüs eğrisi olarak sunulacaktır (Şekil 3.17) (Tarsha-Kurdi, 2007).



Şekil 3.17. Nokta normalinin kullanılması ile bir noktanın parametre uzayındaki gösterimi (Tarsha-Kurdi, 2007)

2 boyutlu uzayda anlatılan Hough dönüşümü 3 boyutlu uzaya da uygulanabilir. Bir düzlemin OXYZ uzayına ait olduğu dikkate alınan bir düzlemin parametre uzayında (O'abc) bir nokta (a, b, c) ile sunulabilir (Eşitlik 3.4) (Tarsha-Kurdi, 2007).

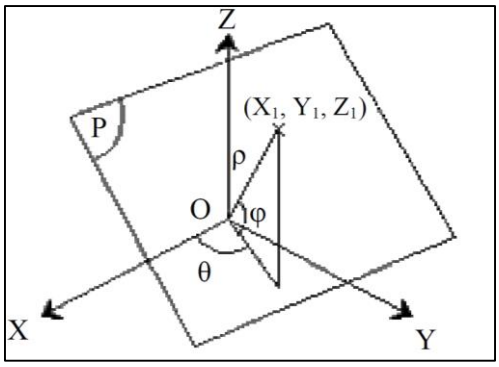
$$Z = a.X + b.Y + c \tag{3.4}$$

Aynı mantıkla eğer düzlem eşitliği Eşitlik 3.5'deki gibi sunulduğunda bu düzlem parametre uzayında gösterilemez. Çünkü Z ekseninin kat sayısı 0'a eşittir. Bu problemi çözmek için Overby ve ark., (2004), düzlem eşitliği için düzlem normalinin de kullanımını önermiştir (Eşitlik 3.6) (Tarsha-Kurdi, 2007).

$$aX + bY + c = 0 \tag{3.5}$$

$$\cos\theta.\cos\varphi.X + \sin\theta\cos\varphi.Y + \sin\varphi.Z = p \tag{3.6}$$

Burada θ, φ ve p orijinden geçen düzlem normalinin parametreleridir (Şekil 3.18). θ, φ ve p değerleri değişmez ve parametre uzayı (O'θφp)'dir. Bundan dolayı 3 boyutlu uzayda yer alan bir (X_1,Y_1,Z_1) noktası parametre uzayında sinusoidal bir yüzey olarak gösterilir (Tarsha-Kurdi, 2007).



Şekil 3.18. Normal formda düzlem eşitliği elemanlarının gösterimi (Tarsha-Kurdi, 2007)

3 boyutlu Hough dönüşümü, 3 boyutlu nokta bulutu içinden genellikle düzlem yüzeylerin çıkarılması için kullanılmaktadır. Yöntem en iyi düzlemi belirlemek için kesin bir matematiksel ilke sunmaktadır. Ancak yöntemle düzlem çıkarma işlemi zaman alıcıdır (Tarsha-Kurdi, 2007).

3.5.3.2. RANSAC Algoritması

Rastgele Örnek Konsensüsü (RANSAC: RANdom SAmple Consensus) Fischler ve Bolles (1981) tarafından geliştirilmiş bir model yakalama algoritmasıdır. Hough dönüşümü gibi dijital görüntü işleme alanında, görüntüler üzerinde yer alan ve 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 ve Bolles 1981; Hartley ve Zisserman 2003).

RANSAC yöntemi geleneksel model kestirim yöntemlerinin tersine bir algoritmadır. Geleneksel model kestirim algoritmalarında, başlangıç olarak olabildiğince fazla girdi veri seçilip daha sonra geçersiz veriler, model içinden elenirken, RANSAC algoritmasında, olabildiğince az sayıda girdi verisi kullanıp bu girdi veriyi modele uygun oluncaya kadar artırılmaktadır. 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 ve Bolles 1981).

RANSAC algoritmasını çalışma prensibi Şekil 3.19'da gösterilmiştir (Fischler and Bolles 1981; Hartley ve Zisserman 2003).

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) Modele belirlenen bir eşik mesafesi 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, 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 işlemleri tekrar et.

(v) N sayıda deneme sonucunda en büyük S_i konsensüsü S veri seti içinde arana modelin uygun konsensüsü olarak seçilir. Model S_i veri seti içinde yer alan tüm noktalar kullanılarak yeniden hesaplanır ve işlem bitirilir

Şekil 3.19. RANSAC algoritmasının model yakalamada çalışma ilkesi (Hartley ve Zisserman 2003)

Şekil 3.19'da gösterilen algoritmada 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.

t Eşik Değeri Mesafesi

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. Ancak ölçüm hatalarının standart sapması normal Gauss eğrisi şekilde olduğu var sayılırsa, t eşik mesafesi matematiksel olarak hesaplanabilir (Fischler ve Bolles 1981; Hartley ve Zisserman 2003).

Modelin Belirlenmesi için Gerekli Deneme Sayısı

Veri seti içinde uygun modelin seçilmesi için tüm noktaların denenmesi hesaplama açısından uygun değildir. Bunun yerine belirlenecek bir p olasılığındaki değere göre N defa s sayıda örnek nokta seçimi yapılarak modelin hesaplanması sağlanır. RANSAC algoritması için N deneme sayısı Eşitlik 8'de gösterildiği gibi hesaplanmaktadır (Fischler ve Bolles 1981; Hartley ve Zisserman 2003).

$$N = \log(1 - p) / \log(1 - (1-\in)^s) \quad (3.8)$$

Burada; p doğru modelin seçilme olasılığı, $\in$ 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. Çizelge 3.2'de p = 0.99 olasılığına göre s ve $\in$ değerine göre yapılması gereken deneme sayısı gösterilmiştir (Fischler ve Bolles 1981; Hartley ve Zisserman 2003).

Çizelge 3.2. Model parametrelerinin hesaplanması için seçilmesi gereken s nokta sayısına göre ve veri seti içinde yer alan $\in$ aykırı değer oranına göre N deneme sayısını gösteren çizelgedir

Örnek nokta sayısı	**Veri Seti İçindeki Aykırı Değer Oranı ($\in$)**						
s	**%5**	**%10**	**%20**	**%25**	**%30**	**%40**	**%50**
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	6	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Kabul Edilebilir Konsensüs Seti Genişliği

RANSAC yöntemi ile model belirleme uygulamasında, N sayısı kadar deneme yapılmadan işlemi sonlandırmak için, modele atanması gereken nokta sayısını gösteren T eşik değerinin belirlenmesi gerekmektedir. T eşik değeri girdi veri içindeki nokta sayısını ve veri seti içinde tahmin edilen ∈ aykırı değer oranına göre hesaplanmaktadır. n sayıda veri içeren bir veri seti için T eşik değeri Eşitlik 3.9'daki gibi hesaplanır (Fischler ve Bolles 1981; Hartley ve Zisserman 2003).

$$T = (1-\in).n \tag{3.9}$$

3.5.4. Hibrit Segmentasyon Tekniği

Bu teknikte genellikle birden fazla segmentasyon yöntemi birleştirilerek nokta bulutunun segmentasyonu gerçekleştirilir. Genellik bölge geliştirme yöntemi ile diğer teknikler birleştirilmektedir. Burada diğer tekniklerden kasıt noktalar arasında bulunan doğal konumsal yakın ilişkisini dikkate alan algoritmalardır (Sapkota, 2008).

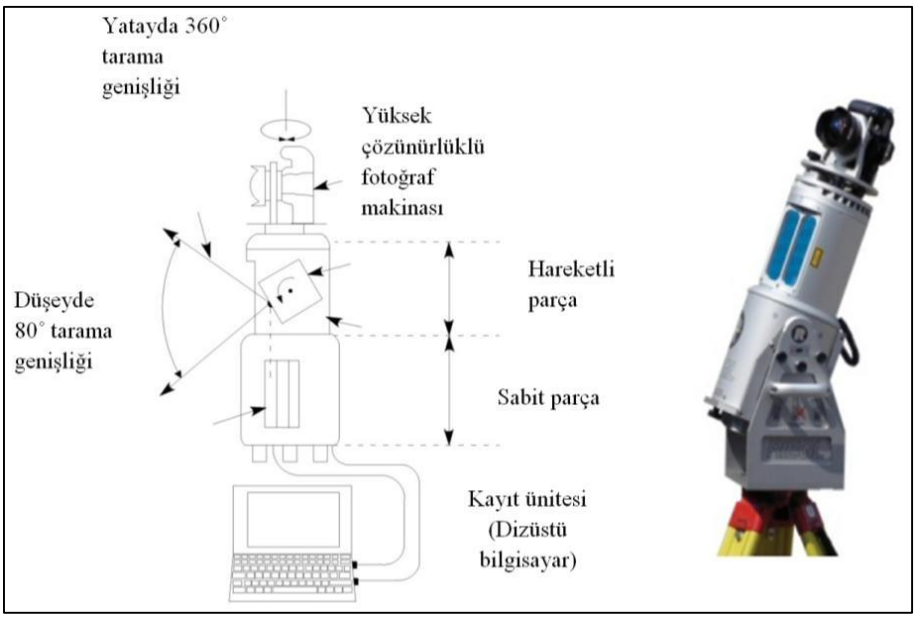
4. MATERYAL VE YÖNTEM

4.1. Materyal

Bu tez çalışması kapsamında yersel lazer tarayıcılardan elde edilen nokta bulutu verisinden geometrik şekle sahip yüzeyler otomatik olarak çıkarılmıştır. Tez çalışmasının materyal bölümünde çalışma kapsamında kullanılan materyallerden bahsedilecektir.

4.1.1. Çalışmada Kullanılan Yersel Lazer Tarayıcı

Tez çalışmasında kullanılan nokta bulutu verilerinin elde edilmesinde Riegl marka LMS Z-390i model 3B lazer tarayıcı kullanılmıştır. Bu lazer tarayıcı lazer ışını gidiş geliş zamanı ilkesine göre çalışmaktadır. Normal ışık ve yansıtma şartları altında 50 metre mesafede 6 mm hassasiyete sahip ve 1.5 - 400 metre arasında ölçüm yapabilmektedir. Tarayıcıdan çıkan lazer ışını yakın kızılötesi ve 0.7 μm – 1.3 μm arasında değişen dalga boyuna sahiptir. Tarayıcı 80° düşey eksen ve 360° yatay eksen yönünde dönme kabiliyetine sahiptir. Cihazın açısal çözünürlüğü 0.001° değerine kadar artırılabilmekte ve cihaz ile saniyede 8000 - 11000 arası nokta verisi elde edebilmektedir (www.riegl.com.tr) (Şekil 4.1).



Şekil 4.1. Veri elde etmede kullanılan yersel lazer tarayıcı

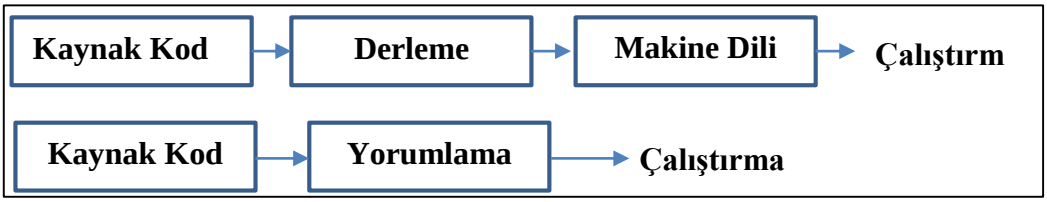
Tarayıcı, objelere ait gerçek renk verisini elde etmek için kalibre edilmiş Nikon D200 kamera ile bütünleşik bir şekilde çalışmaktadır. Bu kamera 10.3 megapiksel DX formatlı CCD sensore sahiptir. Nikon D200 kamera ile tarama işleminden sonra taranan alanın bindirmeli olarak fotoğrafları çekilir. Çekilen fotoğraflar nokta bulutunun renklendirilmesi için kullanılır. Riegl LMS Z-390i lazer tarayıcı Riscan Pro yazılımı tarafından kontrol edilmektedir. Bu yazılım tarayıcının ve kameranın kontrolü, verilerin toplanması, verilerin görselleştirilmesi, verilerin işlenmesi ve verilerin arşivlenmesi gibi birçok işlevi yerine getirmektedir.

4.1.2. C ++ Programlama Dili

C++ programlama dili C programlama dili üzerinde nesne-tabanlı yaklaşımı gerçekleştiren, sistem programlamada yaygın bir şekilde kullanılan, Bell Laboratuarında geliştirilmiş bir dildir (Turhan, 2012).

Bir problemi çözmek için yazılan C++ programının, öncelikle bir metin editörü ile bilgisayara yazılması gerekir. Genellikle bu metin editörü kullanılan derleyicilerin içinde bulunmaktadır. C++ aracılığı ile yazılan programa kaynak kod adı verilir (Turhan, 2012).

Oluşturulan kaynak kodun bilgisayar tarafından çalıştırılması sürecinde kullanılan programlama diline bağımlı olarak farklı yöntemler uygulanmaktadır. En çok kullanılan iki yöntem derleme ve yorumlama yaklaşımlarıdır. Şekil 4.2'de görüldüğü gibi derleme yönteminde kaynak kod, derleyici tarafından bilgisayarın anlayacağı tek dil olan makine diline çevrilir. Makine dili, ikili tabanda 1 ve 0'lardan oluşan alt seviyede bir dildir. Daha sonra, makine diline çevrilen program bilgisayar tarafından çalıştırılır. Yorumlama yönteminde ise kaynak koddaki her komut makine diline çevrilip çalıştırılır. Programın makine diline çevrilmiş hali saklanmaz (Turhan, 2012).



Şekil 4.2. Program çalıştırma yöntemleri

C++ programlama dilinde derleme yöntemi kullanılmaktadır. Aşağıda C++ derleyicisinin gerçekleştirdiği aşamalar detaylı olarak anlatılmıştır. C++ programlama dilini makine diline çevirme süreci aşağıda anlatılan üç aşamadan oluşur (Turhan 2012).

- Önişlemci kaynak kodunu okuyup ilk işlemleri gerçekleştirir,
- Değiştirilmiş kaynak kodda yer alan her komut, derleyici tarafından kontrol edilir ve eğer kaynak kodda herhangi bir söz dizimi hatası (syntax error) varsa, derleyici tarafından programcıya belirtilir. Hatalardan arındırılmış kaynak kod derleyici tarafından makine diline dönüştürülür. Derleyicinin oluşturduğu makine koduna nesne kodu ismi verilir,
- Programın kullanacağı tüm kütüphane fonksiyonları makine kodları bağlayıcı yardımı ile programın içine katılır ve program bilgisayar tarafından çalıştırılabilir koda dönüştürülür. Program çalıştırıldığında varsa mantıksal hata mesajları, yoksa çıktı görüntülenir,

C ++ derleyicilerinde yukarıda bahsedilen tüm aşamalar menüler yardımı ile son derece kolay bir şekilde gerçekleştirilebilmektedir. Önişleme, derleme, bağlama ve hatta programı çalıştırma işlemleri tek bir butona basılarak otomatik olarak yapılabilmektedir (Turhan 2012).

4.1.3. Nokta Bulutu Kütüphanesi (Point Cloud Library: PCL)

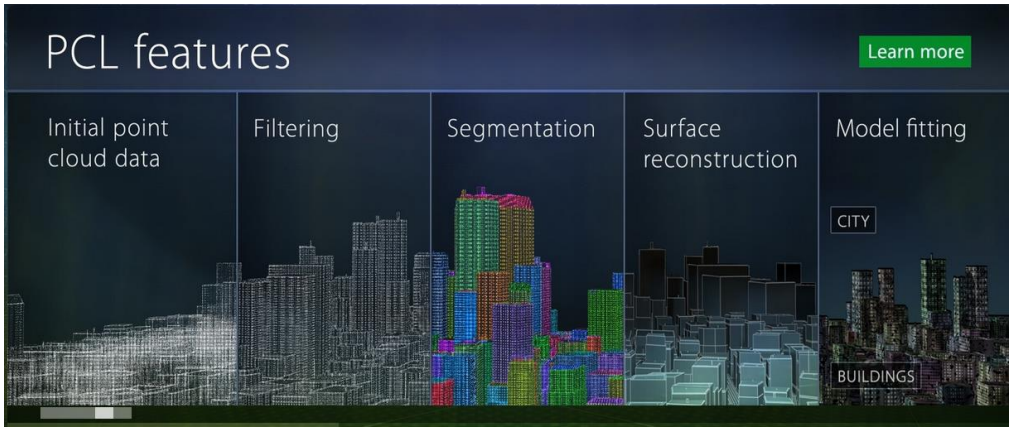
Tez çalışması kapsamında kullanılan RANSAC algoritmasının kodları nokta bulutu kütüphanesinden (PCL: Point Cloud Library) elde edilmiştir. Nokta bulutu kütüphanesi tamamen ücretsiz, BSD (Berkeley Software Distribution) özgür yazılım lisasına sahip, n boyutlu nokta bulutu ve 3 boyutlu geometri işleme kütüphanesidir (http://pointclouds.org). PCL tamamen Robot Yönetme Sistemi (ROS: Robot Operating System- http://ros.org) ile bütünleşik olarak çalışmakta ve şu anda robotik çalışmalarla ilgili, birçok projede kullanılmaktadır (Rusu, 2011).

PCL Mimarisi ve Uygulaması

PCL 3 boyutlu nokta bulutu işleme için tamamen şablonlardan oluşan modern bir C++ kütüphanesidir. Kütüphanede yer alan birçok matematiksel operatörler lineer cebir için açık kaynaklı şablon kütüphanesi olan Eigen temeli

üzerine kurulmuştur (http://eigen.tuxfamily.org/index.php?title=Main_Page). Buna ek olarak PCL OpenMP (http://openmp.org) ve çok çekirdekli paralelizasyon için Intel Threading Buildings Blocks (TBB) kütüphanesi destek sunmaktadır. PCL temel hızlı k-yakın komşuluk araştırma operatörleri için FLANN (Fast Library for Approximate Nearest Neighboor: Yaklaşık en yakın komşuluk için hızlı kütüphane) kütüphanesinden destek almaktadır. Ayrıca PCL Kütüphanesi nokta bulutlarının görüntülenmesi için VTK (Visualization Toolkit: Görselleştirme araçtakımı) kütüphanesini kullanmaktadır. PCL kütüphanesi Windows, MacOS ve Linux işletim sistemleri üzerinde çalışabilmektedir (Rusu 2011).

PCL kütüphanesine algoritmik bir pencereden bakılacak olursa, PCL çok sayıda 3 boyutlu nokta bulutu işleme algoritmalarını birlikte çalışabilirliğini sağlamaktadır. Bu algoritmalar filtreleme, nesne tahminleme, yüzey yeniden inşa etme, model oturtma, segmentasyon, registirasyon vb. algoritmaları içermektedir (Rusu 2011). Şekil 4.3'de PCL web sitesine ait bir görüntü sunulmuştur.



Şekil 4.3. PCL nokta bulutu işleme kütüphanesi

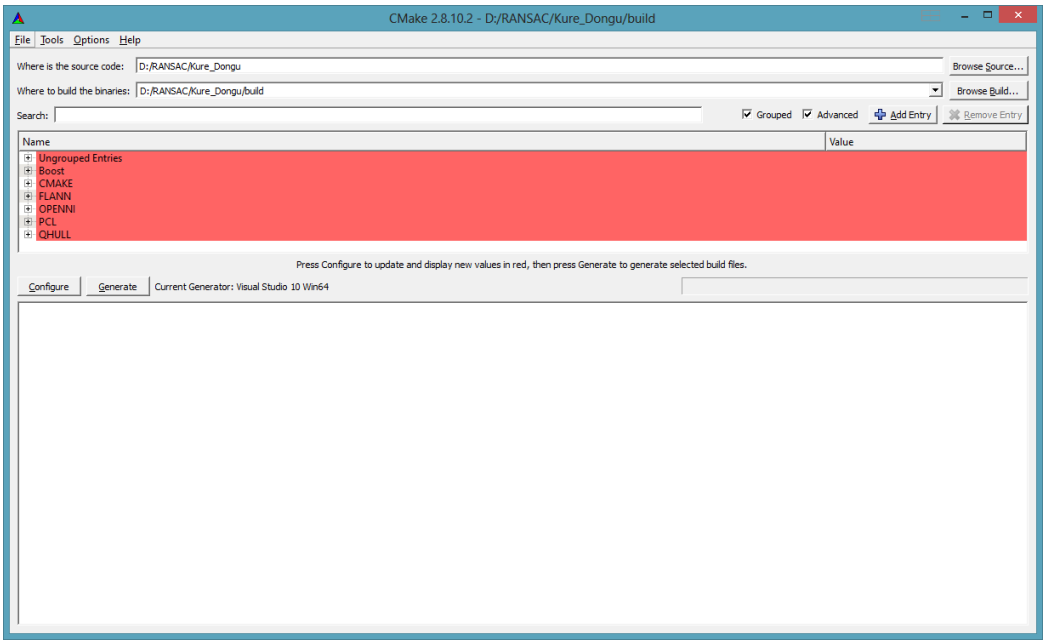
PCL kütüphanesinde kullanılan operatör ve algoritmalarının doğruluğundan emin olmak için kütüphanede yer alan her bir sınıf ve yöntem üzerinde birim ve regresyon testleri yapılmaktadır. PCL içinde yer alan kodların kullanımı ve uygulanması ile ilgili birçok örnek ve uygulama PCL web sitesinde mevcuttur. Bu uygulamalarda C++ ortamında kaynak kodlar ve uygulamada kullanılacak veri seti sunulmakta ve uygulamalar adım adım açıklanmaktadır (Rusu, 2011).

Bu tez çalışması kapsamında PCL web sitesinde yer alan ve Windows işletim sistemi Microsoft Visial Studio 2010 C++ ortamında, 64 bitlik bilgisayarlar için geliştirilmiş PCL 1.6.0 kaynak kod kütüphanesi kullanılmıştır.

4.1.4. CMake Yazılımı

PCL 1.6.0 kaynak kod kütüphanesi kişisel çalışmalarda kullanabilmek için kütüphane ile kişisel kaynak kodun ilişkilendirilmesi gerekmektedir. Bu işlemi gerçekleştirmek için CMake yazılımı kullanılmıştır (http://www.cmake.org/).

Cmake yazılımı daha önceden derlenmiş birbirinden bağımsız yazılımların birlikte çalışabilirliğini sağlayan ücretsiz bir yazılımdır. Bu tez çalışmasında PCL kütüphanesinde yer alan kaynak kodların kullanımı için CMake yazılımı aracılığı ile PCL Kütüphanesi kullanılabilir hale getirilmiştir (Şekil 4.4).



Şekil 4.4. CMake yazılımı arayüzü

4.1.5. Verilerin İşlenmesinde Kullanılan Bilgisayar

Yersel lazer tarayıcılar ile elde edilen nokta bulutu çok sayıda nokta verisi içerdiği için, bu verilerin bilgisarlarda işlenmesi zaman almaktadır. Bundan dolayı nokta bulutu verilerinin hızlı bir veri işlenebilmesi için, veri işlemede kullanılacak bilgisayarın özelliklerinin iyi olması gerekmektedir. Çalışma kapsamında veri işleme işlemi için işlemci hızı ve bellek özellikleri güçlü olan bir bilgisayar tercih edilmiştir. Bu doğrultuda yersel lazer tarayıcılardan elde edilen nokta bulutu verileri

DELL Precision M 6700 mobil iş istasyonu kullanılarak işlenmiştir. Bilgisayara ait teknik özellikler Çizelge 4.1'de sunulmuştur.

Çizelge 4.1. DELL Precision M 6700 teknik özellikleri

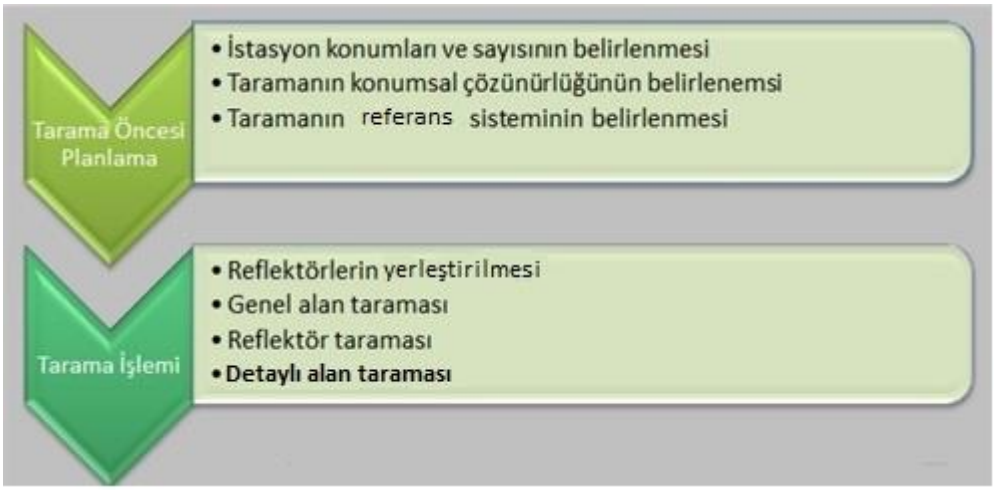
İşlemci ve hızı	Intel Core i7 ; 2.90 GHz
Bellek	16 GB
İşletim Sistemi	Windows 8 Professional 64 bit
Ekran Kartı	2 GB

4.2. Yöntem

Çalışmanın yöntem bölümünde yersel lazer tarayıcı ile verilerin nasıl elde edildiği ve elde edilen verilerden RANSAC algoritması kullanılarak düzlem, silindir, küre ve koni gibi yüzeylere sahip nesnelerin otomatik olarak çıkarılmasında izlenen yol açıklanacaktır.

4.2.1. Yersel Lazer Tarayıcılar ile Verilerin Elde edilmesi

Yer ve Uzay Bilimleri Enstitüsü, Uzaktan Algılama ve Fotogrametri Araştırma Birimi tarafından farklı zamanlarda Riegl marka LMS Z-390i model 3 boyutlu lazer tarayıcı kullanılarak, farklı projeler kapsamında tarihi yapı ve binalara ait tarama işlemi gerçekleştirilmiştir. Bu tez çalışması kapsamında ise, laboratuvar ortamında yapılan bir test taraması ile daha önceki projeler kapsamında elde edilen nokta bulutu verileri kullanılmıştır. Nokta bulutlarının elde edilmesinde aşağıdaki yöntem izlenmiştir (Şekil 4.5).



Şekil 4.5. Riegl LMS Z-390i 3 Boyutlu lazer tarayıcı ile yapılan ölçüm işlemine ait iş akışı

Tarama Öncesi Planlama

Taramaya söz konusu nesnelerin taranması işlemine geçilmeden önce arazide tarama işlemine ait planlamanın yapılması gerekmektedir. Planlama aşamasında nesneyi taramak için kullanılacak istasyon konumlarının belirlenmesi,

taramanın konumsal çözünürlüğü ve istasyonlardan elde edilecek nokta bulutlarının birleştirileceği konumsal referans sisteminin belirlenmesi gerekir.

İstasyon sayısı taranacak nesnenin taramaya konu alan tüm yüzeylerini tarayacak ve gölgede alan bırakmayacak şekilde belirlenir. Taramanın konumsal çözünürlüğü, tarayıcının mesafe ölçüm hassasiyeti ile hangi sıklıkta nokta ölçmesini belirten açısal çözünürlüğe bağlıdır (Riverio, 2011). Çalışmada kullanılan lazer tarayıcı 50 metreye kadar 6 mm hassasiyetinde mesafe ölçümü yapabilmektedir. Çalışma alanında tarama istasyonları ile taranan en uzak alan arasındaki mesafe 50 metreden daha küçüktür. Açısal çözünürlük tarayıcı ile taranan obje arasında değişen mesafeye göre belirlenmesi gerekir. Taranan nesne ile tarayıcı arasında mesafe artıkça daha sık aralıklarla ölçüm yapabilmek için açısal çözünürlüğün artırılması gerekir. Tarayıcı ile taranan nesne arasındaki mesafe 10 metre olduğu düşünülürse, açısal çözünürlük 0.12° seçildiğinde, nokta örnekleme aralığı 2.1 cm, 0.11° seçildiğinde ise 1.9 cm olmaktadır. Mesafe 10 metreden fazla olduğunda nokta örneklem aralığı 2.1 cm ve 1.9 cm daha büyük, mesafe 10 metreden az olduğunda nokta örneklem aralığı 2.1 cm ve 1.9 cm daha küçük olmaktadır.

Tarama iş planlaması aşamasında taramanın hangi koordinat sisteminden yapılacağı belirlenmelidir. Tarama koordinat sistemi jeodezik veya lokal bir koordinat sistemi olabilir. Bu tez çalışması kapsamında kullanılan tüm nokta bulutu verileri lokal koordinat sistemine sahip verilerdir. Bu lokal koordinat sistemi genellikle taramada kullanılan ilk istasyonun koordinat sistemidir.

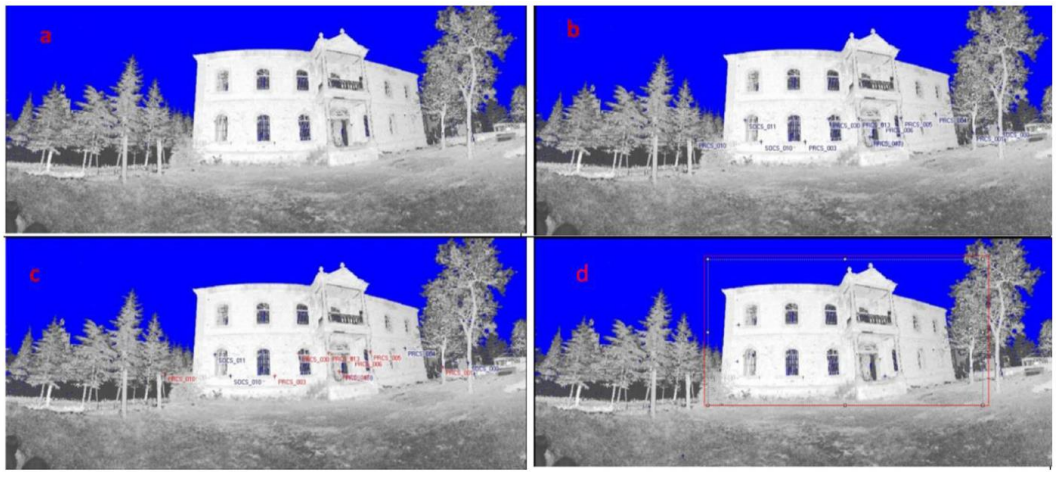
Tarama İşlemi

Tarama işlemine geçilmeden önce farklı istasyonlardan elde edilen nokta bulutu verilerini ortak bir koordinat sistemine dönüştürmek için kullanılan reflektörler uygun yerlere yerleştirilmelidir. Bu uygun yerler cihazı tam karşıdan ve dik olarak gören ve cihaza yakın yerler olarak belirlenmelidir. Reflektörler uygun yerlere yerleştirilmediğinde, sağlıklı taranamamakta ve farklı istasyonlardan elde edilen nokta bulutu verilerinin birleştirilmesi işleminde sorunlar yaşanmaktadır.

Taranan nesnenin tamamına ait 3 boyutlu nokta bulutunu elde etmek için planlama aşamasında belirlenen her bir istasyondan tarama işlemi gerçekleştirilir.

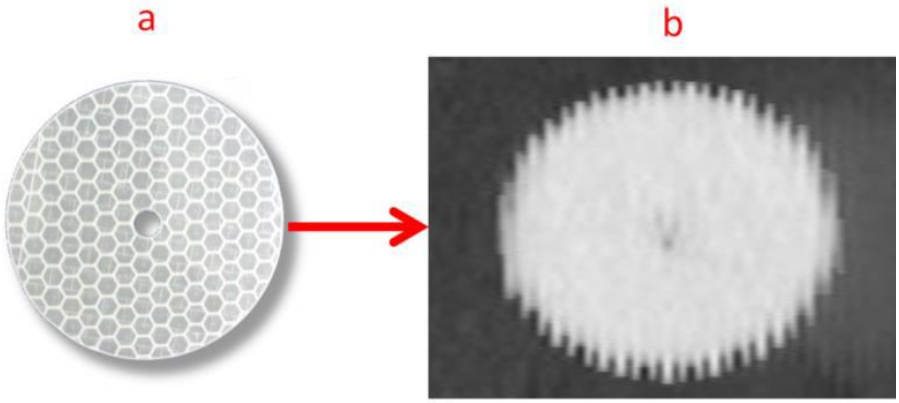
Her bir istasyonda 3 farklı tarama işlemi yapılır. Bu taramalardan ilki tarayıcının bulunduğu pozisyondan gördüğü genel alan taraması, ikincisi reflektörler için yapılan reflektör taraması, üçüncüsü ise tarayıcının bulunduğu pozisyondan yapılması gereken detaylı alan taramasıdır. Farklı istasyonlardan elde edilen nokta verilerinin birleştirilmesi için ardışık taramalarda en az 3 reflektörün ortak olması gerekir.

Genel alan taraması tarayıcının bulunduğu pozisyondan gördüğü alanın minimum 0.2° açısal çözünürlük ile taranması işlemidir. Taranacak genel alan 0°-360° arasında olabileceği gibi kullanıcı tarafından seçilen belirli bir açısal aralık da (30°-270° gibi) olabilir. Burada dikkat edilmesi gereken genel alan taramasının, detaylı alan taraması yapılacak alanı ve reflektörlerin bulunduğu alanları içine almasıdır. Yazılımda, tarama işlemine başlamadan önce tarayıcının reflektörleri algılama modu aktif hale getirildiğinde reflektörler otomatik olarak belirlenmektedir. Ancak minimum çözünürlükte yapılan genel alan taramasında yazılım birçok parlak cismi reflektör olarak algılayabilmektedir. Bundan dolayı dönüşüm işleminde kullanılacak reflektörler genel alan taraması üzerinden kullanıcı tarafından seçilmesi gerekmektedir. Şekil 4.6 a'da ilk istasyondan yapılan genel alan taraması, Şekil 4.6 b'de genel alan taraması sonucunda cihazın reflektör olarak algıladığı cisimler, Şekil 4.6 c'de genel alan taraması üzerinden reflektörlerin kullanıcı tarafında seçilmesi ve Şekil 4.6 d'de ise genel alan taraması üzerinden detaylı taranacak alanın seçilmesi gösterilmektedir.



Şekil 4.6. a: Genel alan taraması; b: Genel alan taraması üzerinde cihaz tarafından reflektör olarak algılanan parlak cisimler; c: Genel alan taraması üzerinden hassas olarak taranacak reflektörlerin seçilmesi; d: Genel alan üzerinden detaylı taranacak alanın seçilmesi

Reflektör taraması işlemi genel alan taraması üzerinden seçilen reflektörlerin maksimum çözünürlükte taranması işlemidir. Reflektörlerin maksimum çözünürlükte taranmasındaki amaç, orta noktalarının en iyi şekilde belirlenmesi içindir. Reflektörler ne kadar fazla piksel ile temsil edilirlerse orta noktaları o derece iyi belirlenmektedir. Reflektörler 500 piksel altında tarandığında ve bu reflektörler dönüşüm işleminde kullanıldıklarında nokta bulutlarının birleştirilirken hatalar ile karşılaşılmaktadır. Bundan dolayı 500 pikselin altında taranan reflektörler dönüşüm işlemine dahil edilmemelidir. Şekil 4.7'de nokta bulutlarını birleştirmek için kullanılan 5 cm çapındaki reflektör ve bunun maksimum çözünürlükte taranmış görüntüsü yer almaktadır.



Şekil 4.7. a: Nokta bulutlarını birleştirmek için kullanılan reflektör; b: hassas olarak taranmış görüntüsü

Detaylı alan taraması, cihazın kurulu olduğu istasyondan taranacak alanın belirlenerek yüksek çözünürlüklü olarak taranması işlemidir. Bu işlem için genel alan taraması üzerinden detaylı taranacak alan seçilir (Şekil 6.7d). Belirlenen bir açısal çözünürlükte tarama işlemi gerçekleştirilir. Tarama işleminin hemen ardından nokta bulutlarını renklendirmek için kullanılacak olan fotoğraflar çekilir.

4.2.2. Verilerin Hazırlanması

Riegl LMS Z-390i yersel lazer tarayıcının kontrolü ve elde edilen verilerin işlenmesi RiScan Pro yazılımı ile gerçekleştirilmektedir. Bu çalışma kapsamında kullanılan yersel lazer tarayıcı verilerinin birleştirilmesi, filtrelenmesi, fazlalık

verilerden arındırılması ve ASCII formatında dışarı aktarılması gibi işlemler RiScan Pro yazılımında yapılmıştır.

RiScan Pro yazılımında farklı istasyonlardan elde edilen verilerin birleştirilmesi işlemi, istasyonlarda ortak olarak taranan reflektörler aracılığı ile gerçekleştirilmektedir. Yazılım, farklı istasyonlardan elde edilen nokta bulutları reflektörler aracılığı ile birleştirirken iteratif en yakın nokta (ICP) yöntemi kullanılmaktadır. Yazılım aynı zamanda, nokta bulutu içindeki kullanılmayacak verilerin elle temizlenmesine olanak tanımaktadır. Bununla birlikte yazılımda nokta bulutunun seyreltilmesinde kullanılabilecek birçok filtreleme algoritması vardır.

Tez çalışmasında nokta bulutu içinde yer alan noktaların sadece X,Y ve Z koordinatları kullanılmıştır. Çalışmada uygulama verisi olarak kullanılan nokta bulutları gerekli birleştirme ve temizleme işleminden sonra ASCII formatında dışarı aktarılmıştır. Dışarı aktarılan dosyalar PCL kütüphanesinin kullandığı PCD dosya formatına dönüştürülmüştür. PCD dosya formatının özellikleri EK-2'de sunulmuştur.

4.2.3. RANSAC Algoritması ile Yersel Lazer Tarayıcı Verilerinden Otomatik Yüzey Çıkarımı

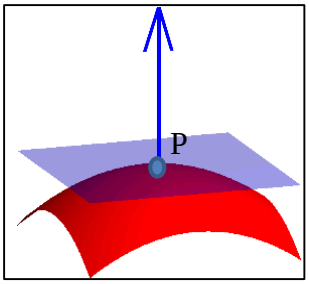
Bölüm 3.5.3.2 RANSAC algoritmasının anlatımında nokta bulutu içinden geometrik bir yüzey modeli çıkarılırken belirlenecek bir t eşik mesafesi değerine göre nokta ile model arasındaki mesafe hesaplanarak, noktanın modele düşüp düşmediği belirlenmekte olduğu anlatılmıştı. Bununla birlikte RANSAC algoritması ile yüzey çıkarım işleminde her bir noktanın yüzey normalini de dikkate alarak yüzey çıkarım işlemi gerçekleştirebilmektedir. Bu çalışma kapsamında nokta bulutu içinde yüzey çıkarımı yapılırken noktaların yüzey normalleri de dikkate alınacaktır.

4.2.3.1. Nokta Bulutu İçinde Yüzey Normalinin Belirlenmesi

Nokta bulutu içinden RANSAC algoritması ile belirlenen bir t eşik mesafesi değerine göre yüzey çıkarma işleminde sadece model ile modele atanacak nokta arasındaki mesafe dikkate alınmaktadır. Çıkarılacak düzlemin doğruluğunu

artırmak için nokta bulutu içindeki her bir noktanın yüzey normali ve model normali arasındaki farkta dikkate alınarak işlem yapılabilmektedir.

3 boyutlu uzayda, bir yüzeye P noktasında teğet olan tanjant düzlemine dik olan doğruya yüzey normali adı verilmektedir. Yüzey normali normal vektör olarak da adlandırılmaktadır (Şekil 4.8).

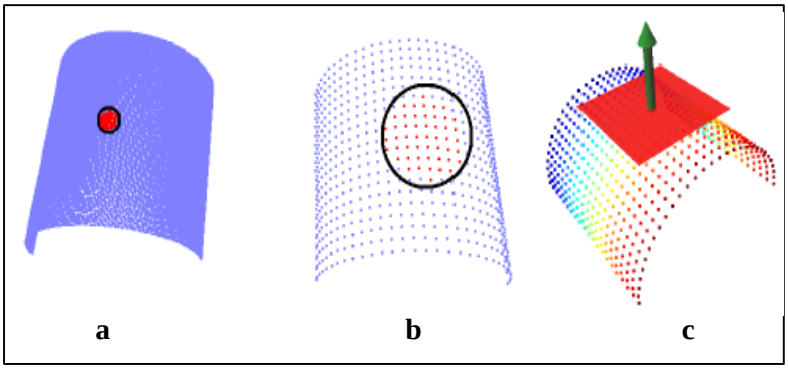


Şekil 4.8. Yüzey normali (http://tr.wikipedia.org/wiki/Normal_(geometri))

Yüzey normali geometrik yüzeyler için son derece önemli bir özelliktir ve bilgisayar grafik uygulamalarında yoğunlukla kullanılmaktadır. Gerçek dünyayı temsil eden nokta bulutu içinden noktaların yüzey normallerini hesaplamak için çeşitli yaklaşımlar vardır. Bu yaklaşımlardan bir tanesi en küçük kareler düzlem yakalama ilkesidir. Bu ilke ile nokta bulutu içinde yer alan bir noktanın yüzey normalini hesaplamak için ilk önce noktaya komşu olan en yakın noktalar bulunmaktadır. Daha sonra seçilen noktaları temsil eden en uygun tanjant düzlemi En Küçük Kareler (EKK) düzlem yakalama ilkesi ile belirlenmektedir. (Rusu, 2008). Verilen k sayıda noktaya sahip bir P_k veri seti içinde p_i noktasının normali en küçük kareler ilkesine göre M kovaryans matrisini minimum yapan birim değer vektörü, noktanın normal vektörünü temsil etmektedir (Eşitlik 4.1).

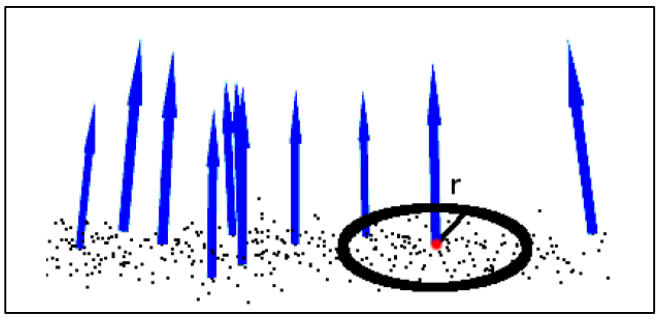
$$1 \leq i \leq k \;; \qquad M = \frac{1}{k} \sum_{i=1}^{k} p_i p_i^T - \bar{p}\bar{p}^T \;; \; \bar{p} = \frac{1}{k} \sum_{i=1}^{k} p_i \tag{4.1}$$

Şekil 4.9'da nokta bulutu içinde yer alan bir noktanın, yüzey normalinin hesaplanması gösterilmiştir.



Şekil 4.9. a: nokta seçimi; b: komşu noktaların belirlenmesi; c: EKK ile yüzey normalinin hesaplanması

PCL kütüphanesini kullanarak nokta bulutu içinde yer alan noktaların yüzey normallerini belirlemek için yüzey normali hesaplanacak noktaya komşu noktaların bilinmesi gerekmektedir. Komşu noktaların belirlenmesi için kütüphane k- en yakın komşuluk yöntemi (k-nearest neighbor estimation) kullanılmaktadır. En yakın komşuluk ilkesine göre işlem yapılacak noktaya en yakın noktaların belirlenmesi için, verilen nokta sayısına göre komşu noktaları bulma (k-Search) ve belirlen bir yarıçapa göre arama yapma (RadiusSearch) olmak üzere iki farklı seçenek sunulmaktadır. Belirlenen komşu nokta arama seçeneğine göre bulunan noktaları içeren düzlem normali en küçük kareler düzlem yakalama ilkesine göre belirlenmektedir. Şekil 4.10'da yarıçapa göre belirlenen komşu noktalara bağlı olarak hesaplanan yüzey normali örneği sunulmuştur (Rusu 2008).



Şekil 4.10. Sorgulama noktasına (kırmızı renkli) komşu noktaları belirlemede kullanılan yarıçap r arama örneği ve yarıçapın oluşturduğu daire içine düşen noktalardan hesaplanan yüzey normali.

RANSAC algoritması ile nokta bulutu içinden geometrik yüzey çıkarımı işleminde, noktaların yüzey normalleri dikkate alındığında, bir noktanın modele

atanıp atanamayacağı kararı verilirken iki kritere dikkat edilmektedir. Bunlar nokta ile model arasındaki uzaklık ve model normali ile nokta normali arasındaki açısal mesafedir.

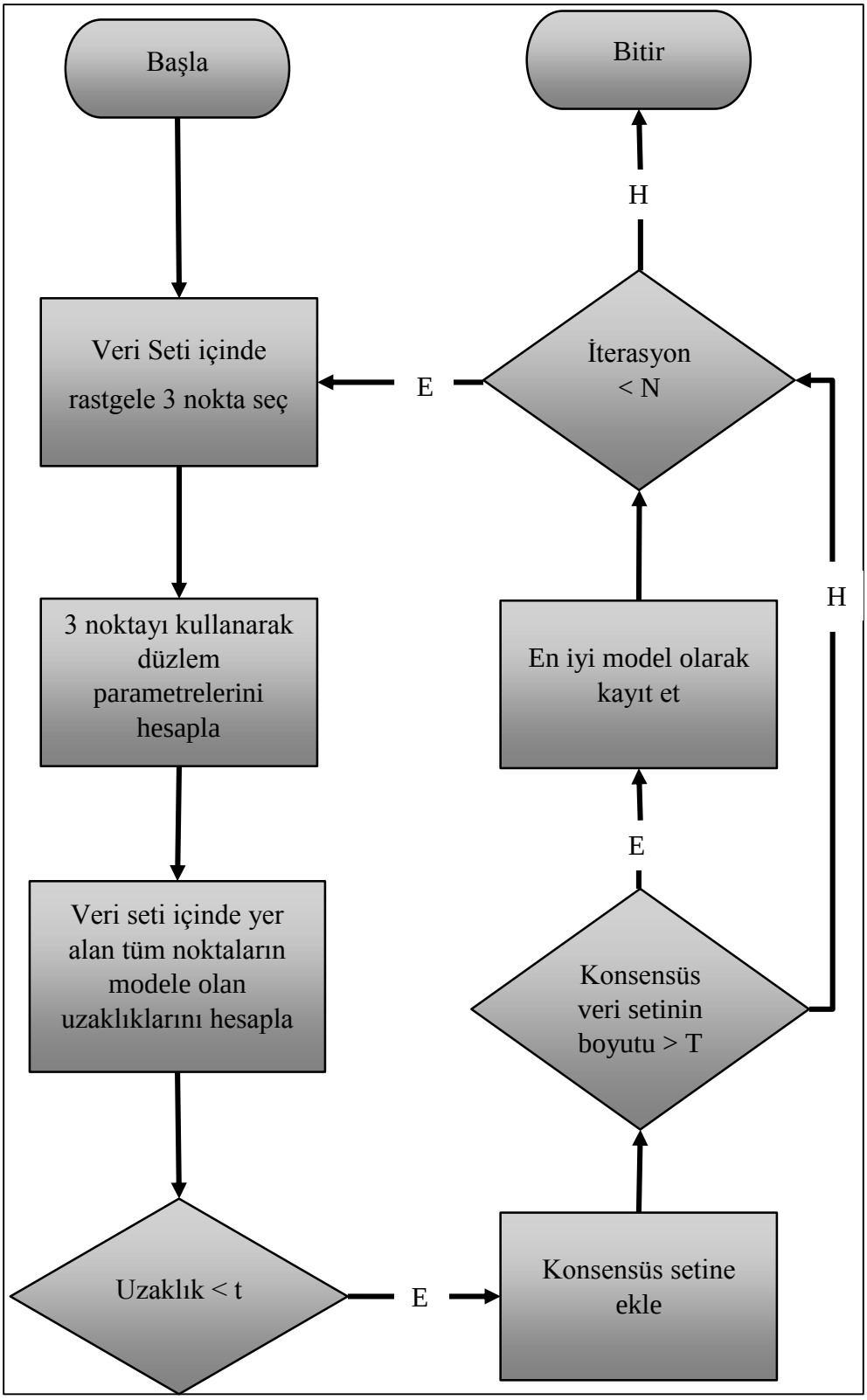
4.2.3.2. Düzlem Yüzey Çıkarma

RANSAC algoritması ile yüzey normali dikkate alınarak ve yüzey normali dikkate alınmadan iki farklı şekilde düzlem yüzey çıkarım işlemi gerçekleştirilmiş ve aralarındaki farklar incelenmiştir.

Yüzey Normalinden Bağımsız Düzlem Yüzey Çıkarma

RANSAC algoritması ile nokta bulutu içinden otomatik düzlem yüzeylerin çıkarılması için rastgele seçilen 3 nokta ile düzlem parametreleri hesaplanır. Düzlem parametre (a, b, c, d) değerlerinden ilk üçü düzleminin normal vektörünü ($a^2 + b^2 + c^2 = 1$) tanımlamaktadır. Dördüncü parametre (d) ise düzlemin orijine olan uzaklığını göstermektedir (Boulaassal ve ark., 2007). Düzlem parametreleri belirlendikten sonra, düzleme belirlenen bir eşik değer mesafesi (t) içinde kalan noktalar düzleme atanır. Noktaların verilen eşik değer içinde kalıp kalmadığını belirmek için noktalar P (x, y, z) ile düzlem PL (a, b, c, d) arasındaki öklid mesafesi hesaplanır (Eşitlik 4.2) (Boulaassal, vd. 2007). RANSAC algoritması ile tek bir düzlemin çıkarılmasında izlenen iş akış diyagramı aşağıdaki gibidir (Şekil 4.11).

$$d_{öklid} = a'x + b'y + c'z + d' = 0 \qquad (4.2)$$



Şekil 4.11. RANSAC algoritması ile tek bir düzlem çıkarılmasında kullanılan iş akış diyagramı

Çalışmada kullanılan nokta bulutu verileri birden fazla düzlem yüzey içerdiğinden RANSAC algoritması veri setlerine ardışık olarak uygulanmıştır. Algoritmanın veri setlerine ardışık olarak uygulanmasındaki amaç nokta bulutları içindeki tüm düzlemleri çıkarmak ve bir noktanın birden fazla düzlem içine girmesini önlemektir. Algoritma veri setlerine her uygulandığında belirlenen bir t eşik değeri mesafesine göre en fazla noktayı içeren düzlem nokta bulutu içinden çıkarılmaktadır. Geriye kalan nokta bulutuna tekrardan algoritma uygulanmakta ve bu şekilde veri seti içinde yer alan düzlem yüzeyler çıkarılmaktadır.

RANSAC algoritması ile düzlem çıkarmada belirlemesi gereken en önemli değerlerden birisi düzleme atanacak noktaların, düzleme en fazla ne kadar uzakta olması gerektiğini gösteren eşik değeri mesafesinin (t) belirlenmesidir. Bu çalışmada t eşik değeri her bir veri seti için deneysel olarak belirlenmiştir.

PCL kütüphanesini kullanarak çıkarılan düzlem yüzeye ait modelin hesaplanabilmesi için belirli değerlerin kullanıcı tarafında girilmesi gerekmektedir. Bu değerler aşağıdaki gibidir.

setModelType (): Nokta bulutu içinde çıkarılacak model tipi tanımlanır.

setMethodType (): Model çıkarmada kullanılacak yöntem tanımlanır.

setMaxIterations (): Uygun modelin çıkarılması için yapılacak maksimum iterasyon sayısı belirlenir.

setDistanceThreshold (): Modele atanacak noktalar için gerekli olan t eşik değeri mesafesi tanımlanır.

PCL kütüphanesi kullanılarak RANSAC algoritması ile düzlem çıkarma işleminde ihtiyaç duyulan model tanımlama parametreleri, Şekil 4.12'de örnek olarak tanımlanmıştır.

```
seg.setModelType (pcl::SACMODEL_PLANE);
seg.setMethodType (pcl::SAC_RANSAC);
seg.setMaxIterations (1000);
seg.setDistanceThreshold (0.01);
```

Şekil 4.12. PCL kütüphanesinde düzlem yüzey çıkarma için gerekli girdi parametreler

Yüzey Normaline Bağlı Düzlem Yüzey Çıkarma

Nokta bulutu içinden yüzey normaline bağlı olarak düzlem çıkarma işleminde ilk önce nokta bulutu içindeki her bir noktanın yüzey normali hesaplanmaktadır. Yüzey normalleri hesaplanan nokta bulutu verisine daha sonra RANSAC algoritması uygulanmaktadır. Veri setine RANSAC algoritmasının uygulanması yüzey normalinden bağımsız düzlem yüzey çıkarma işleminde olduğu gibidir. Aradaki tek fark bir noktanın düzleme atanıp atanmayacağı kararı verilirken düzlem ile nokta arasında mesafenin hesaplanmasında, düzlem normali ile nokta normali arasındaki açısal mesafesininde kullanılmasıdır. Açısal mesafeye bağlı olarak nokta ile düzlem arasındaki mesafenin hesaplanması için aşağıdaki işlemler yapılır.

- İlk olarak nokta ile düzlem normali arasındaki öklid mesafesi ($d_{öklid}$) eşitlik 4.3'e göre hesaplanır.
- İkinci olarak nokta normali ile düzlem normali arasındaki açısal mesafe farkı hesaplanır. 3 boyutlu uzayda iki normal vektör (düzlem normal vektörü ve normal vektör) arasındaki açı iki vektörün skaler çarpımı kullanılarak elde edilmektedir.

  N_d düzlem normal vektörü, N_p nokta normal vektörü olmak üzeri iki vektörün skaler çarpımı;

  $$N_d * N_p = |N_d| * |N_p| * \cos\alpha \quad (4.3)$$

  Buradan α eşitlik 4.4'e göre radyan biriminden hesaplanır.

  $$\alpha = \cos^{-1}((N_d * N_p)/(|N_d| * |N_p|)) \quad (4.4)$$

  Birim küre için d_normal mesafesi eşitlik 4.5'deki gibi tanımlanır.

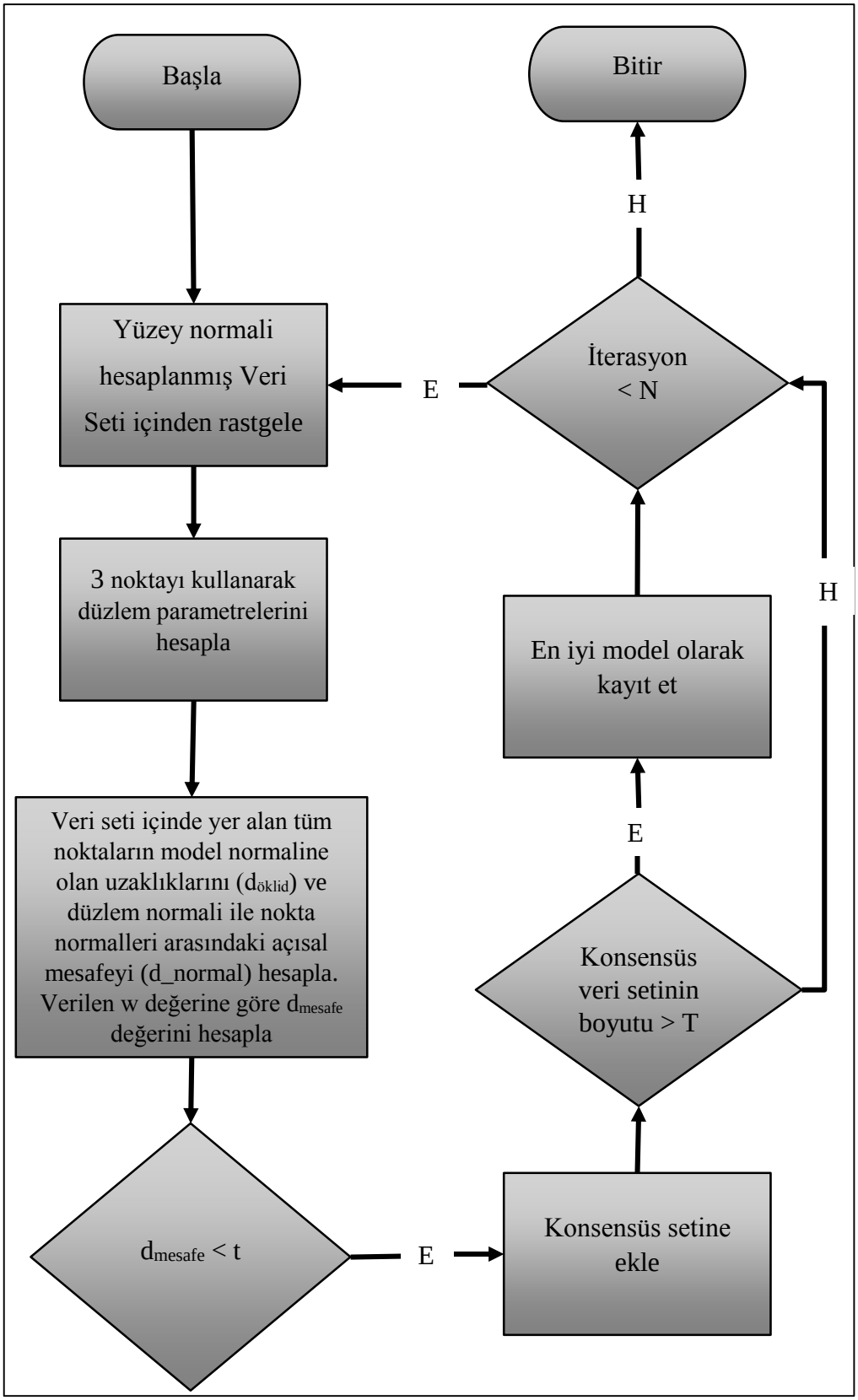
  $$d_{normal} = 1 * \alpha \quad (4.5)$$

- Üçüncü olarak ise verilen bir w ağırlık değerine göre nokta ile düzlem arasındaki mesafe d_{mesafe} Eşitlik 4.6'ya göre hesaplanır. w ağırlık değeri nokta normali ile düzlem normali arasında hesaplanan açısal mesafenin d_{mesafe} değerine etkisini göstermektedir.

  $$d_{mesafe} = (d_{normal} * w) + (1 - w) * d_{öklid} \quad (4.6)$$

Nokta ile düzlem arasında hesaplanan d_{mesafe} değerinin verilen t eşik mesafesi altında olup olmama durumuna göre noktalar düzleme atanmaktadır. Yüzey normali ile birlikte düzlem yüzey çıkarma işleminde nokta normalleri hesaplanan noktalara uygulanan RANSAC algoritmasına ait iş akış diyagramı Şekil 4.13'de gösterilmiştir.

Yüzey normaline bağlı olarak yapılan düzlem yüzey çıkarma işleminde nokta ile model arasında hesaplanan d_{mesafe} değerinin hesaplanmasında kullanılan w ağırlık değeri 0-1 arasında değişen değerler almaktadır. Bu değer t eşik mesafesinde olduğu gibi kullanıcı tarafından tanımlanması gereken bir değerdir ve deneysel olarak belirlenmektedir. Eğer w değeri 0 verilirse mesafe hesaplamada yüzey normali dikkate alınmadan işlem yapılmaktadır.



Şekil 4.13. RANSAC algoritması ile Nokta normali ile birlikte düzlem yüzey çıkarma iş akış şeması

PCL kütüphanesi kullanılarak nokta bulutunun yüzey normali hesaplanırken, aşağıdaki fonksiyonlar kullanılmaktadır.

setSearchMethod (): Nokta bulutu içinde yer alan noktaların k boyutlu uzayda organize edilmesi için kullanılacak yöntem tanımlanır. PCL kütüphanesinde KdTree olarak adlandırılmaktadır.

setInputCloud (): Yüzey normali hesaplanacak nokta bulutu tanımlanır.

setKSearch (): Yüzey normali hesaplanacak nokta için komşu noktaları arama yönteminden birisidir. Kaç tane nokta aranacağı belirlenir.

setRadiusSearch (): Yüzey normali hesaplanacak nokta için komşu noktaların hangi yarıçap içinde aranacağı tanımlanır.

compute (): Belirlenen yöntemlere göre nokta bulutu içinde yer alan tüm noktaların yüzey normali hesaplanır.

Yüzey normali hesaplanacak noktanın komşu noktaları belirlenirken KSearch veya RadiusSearch yöntemlerinden sadece birisi kullanılır. Şekil 4.14'de C++ ortamında PCL kütüphanesi kullanılarak nokta normali hesaplama örneği sunulmuştur.

```
//Nokta normalini hesapla
ne.setSearchMethod (tree);
ne.setInputCloud (cloud);
ne.setRadiusSearch(0.03);
ne.compute (*cloud_normal);
```

Şekil 4.14. PCL kütüphanesinde nokta normali hesaplama

Hesaplanan nokta normallerine göre PCL Kütüphanesi ile düzlem yüzey çıkarmak için gerekli model, yöntem ve eşik değeri tanımlama fonksiyonları yüzey normalinden bağımsız düzlem yüzey çıkarmada olduğu gibidir. Bu fonksiyonlara ek olarak w ağırlık değerini tanımlamak için setNormalDistanceWeight () fonksiyonu kullanılır.

setNormalDistanceWeight (): Model normali ile nokta normali arasındaki maksimum açısal mesafenin, nokta ile düzlem arasındaki mesafenin hesaplanmasına olan etki değeri belirlenir. Bu değer 0 ile 1 arasında değişen bir ağırlık değeridir.

Şekil 4.15'de C++ ortamında PCL kütüphanesi kullanılarak yüzey normaline bağlı olarak düzlem çıkarmak için örnek bir tanımlama yapılmıştır.

```
seg.setModelType (pcl::SACMODEL_NORMAL_PLANE);
seg.setMethodType (pcl::SAC_RANSAC);
seg.setNormalDistanceWeight (0.1);
seg.setMaxIterations (1000);
seg.setDistanceThreshold (0.01);
```

Şekil 4.15. Yüzey normaline bağlı olarak düzlem yüzey çıkarma model, yöntem, iterasyon ve eşik değeri tanımlama

4.2.3.3. Silindir Yüzey Çıkarma

RANSAC algoritması ile silindir bir yüzey çıkarmak için yedi adet parametrenin hesaplanması gerekmektedir. Bu parametreler silindir ekseninde yer alan bir P_1 noktasının X, Y, Z koordinatları, silindir ekseninin doğrultusunu gösteren bir P_2 noktasının X, Y, Z koordinatları ve silindirin yarıçapı r'dir.

Çalışma kapsamındı veri setleri içinde yer alan silindir yüzeyler noktalara ait yüzey normalleri dikkate alınarak çıkarılmıştır. PCL kütüphanesini kullanarak RANSAC algoritması ile silindir yüzey çıkarma işleminde rastgele 2 nokta seçilmekte ve seçilen bu noktalar ile çıkarılacak silindir modelin parametreleri hesaplanmaktadır. Hesaplanan parametrelere göre bir silindir model oluşturulmaktadır. Model oluşturma aşamasında algoritmanın uygulandığı veri seti içinden aranan modelin belirlenebilmesi için bazı parametrelerin kullanıcı tarafından belirlenmesi gerekmektedir. Bu parametreler silindirin yarıçap sınırları, silindir modelin aranacağı eksen doğrultusu ve epsilon açısıdır.

Yarıçap Sınırları: Veri seti içinde kullanıcı bulmak istediği silindir modelin yarıçapının alabileceği bir maksimum ve minimum değer aralığı tanımlaması gerekir. Bu şekilde veri seti içinden seçilen iki nokta ile silindirin yedi parametresinden birisi olan yarıçap (r) değerinin alabileceği değerler sınırlandırılmış olur.

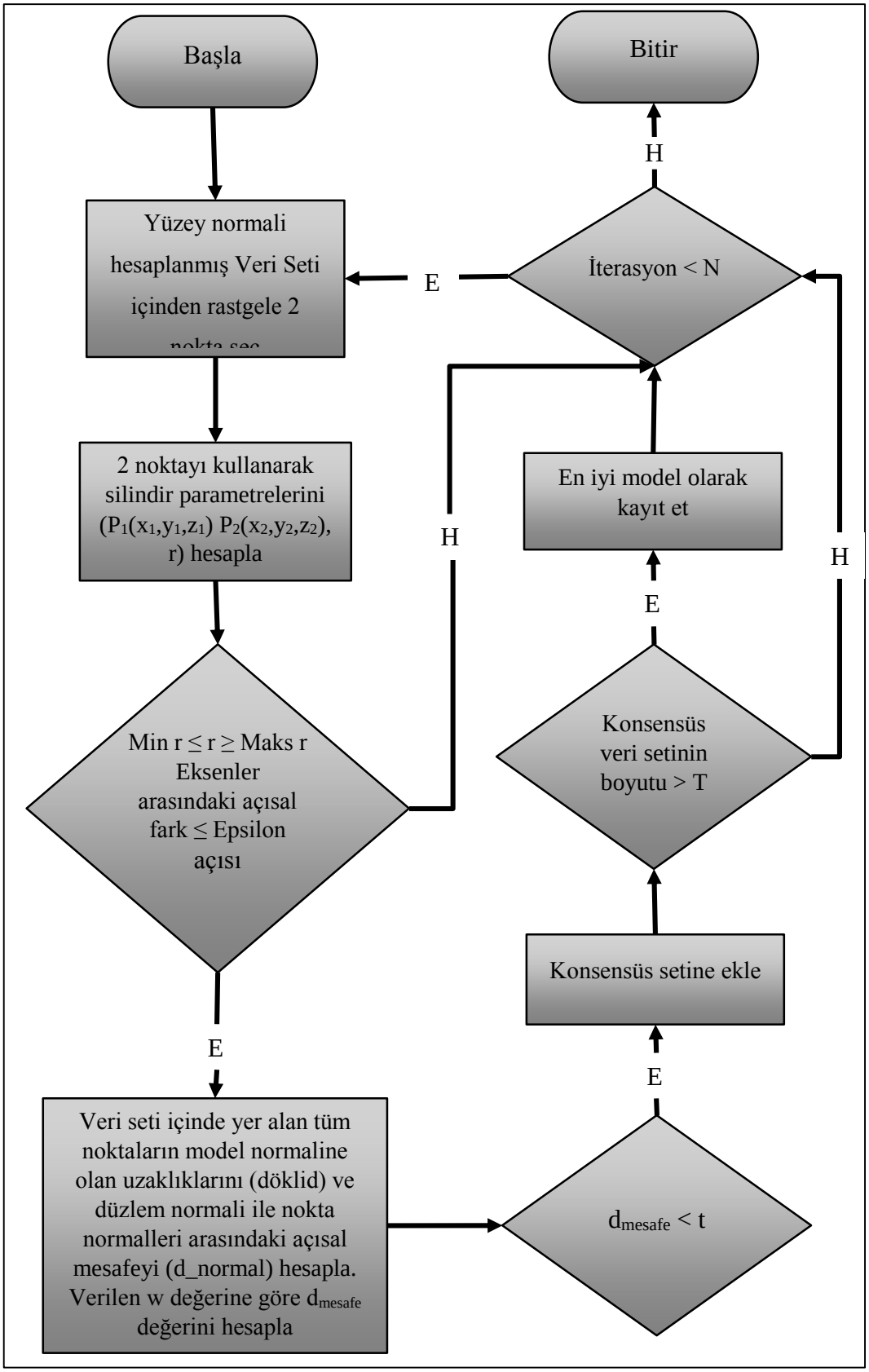
Eksen Doğrultusu: 3 boyutlu koordinat sisteminde silindirin modelin hangi eksen veya eksenler (X, Y, Z eksenleri) yönünde aranacağı belirten doğrultudur.

Epsilon Açısı: Veri seti içinden rastgele seçilen iki nokta ile elde edilen silindir modelin eksen doğrultusunun, silindirin aranacağı eksenden ne kadar sapabileceğini gösteren açıdır.

Yukarıda bahsedilen parametrelerde dikkate alınarak belirlenen silindir modele, atanacak noktaları belirlemek için gerekli olan maksimum t eşik değeri mesafesi ve yüzey normalinin mesafe hesaplamaya etkisini gösteren w ağırlık değeri kullanıcı tarafından deneysel olarak belirlenir. Belirlenen bu değerlere göre silindir model üzerine düşen noktalar nokta bulutu içinden çıkarılır.

Silindir modele atanacak noktalar ile model arasındaki mesafesinin belirlenmesi için ilk önce, nokta ile silindir eksini arasındaki mesafe hesaplanır. Daha sonra ise hesaplanan mesafe değerinden silindir yarıçapı çıkarılarak model ile nokta arasındaki mesafe belirlenir.

Nokta bulutu içinden bir adet silindir yüzeyin çıkarılmasına ilişkin iş akış diyagramı Şekil 4.16'da gösterilmiştir. Silindir yüzey çıkarma işleminde nokta bulutuna RANSAC algoritması uygulanmadan önce her bir noktanın yüzey normalleri hesaplanmaktadır. PCL kütüphanesi kullanılarak noktaların yüzey normallerinin hesaplanması nokta normaline bağlı düzlem yüzey çıkarma bölümünde anlatılmıştır.



Şekil 4.16. RANSAC algoritması ile tek bir silindir çıkarmak için iş akış diyagramı

PCL kütüphanesini kullanarak RANSAC algoritması ile silindir yüzey çıkarmada parametre tanımlama fonksiyonları yüzey normaline bağlı düzlem çıkarmada kullanılan fonksiyonlarla benzerdir. Bu fonksiyonlara ek olarak silindir yüzey çıkarmada aşağıdaki tanımlama fonksiyonları kullanılır.

setRadiusLimits (): Çıkarılacak silindir modelin alabileceği maksimum ve minimum değerler tanımlanır. Birimi metredir.

setAxis(Eigen::Vector3f ()): Silindir modelin hangi eksen veya eksenler doğrultusunda aranacağı tanımlanır.

setEpsAngle (): Silindir ekseni ile modelin belirleneceği eksen (X, Y, Z eksenlerinden herhangi biri) arasında izin verilen maksimum açı tanımlanır. Birimi radyandır.

Aşağıda şekil 4.17'de C++ ortamında PCL kütüphanesini kullanarak nokta bulutu içinden silindir model çıkarmak için tanımlanan parametreler gösterilmiştir.

```
seg2.setModelType (pcl::SACMODEL_CYLINDER);
seg2.setMethodType (pcl::SAC_RANSAC);
seg2.setNormalDistanceWeight (0.1);
seg2.setDistanceThreshold (0.01);
seg2.setMaxIterations (1000);
seg2.setRadiusLimits (0, 0.25);
seg2.setEpsAngle(0.2);
seg2.setAxis(Eigen::Vector3f (0,0,1));
```

Şekil 4.17. Silindir yüzey çıkarmak için kullanıcı tarafından tanımlanan parametreler

4.2.3.4. Küre Yüzey Çıkarma

RANSAC algoritması ile nokta bulutu içinden nokta yüzey normallerini de hesaplayarak küre yüzey çıkarmak için kürenin dört adet parametrenin hesaplanması gerekmektedir. Bu parametreler küre merkezin normalize edilmiş X,Y, Z koordinatları ve kürenin yarıçapı (r) değerleridir.

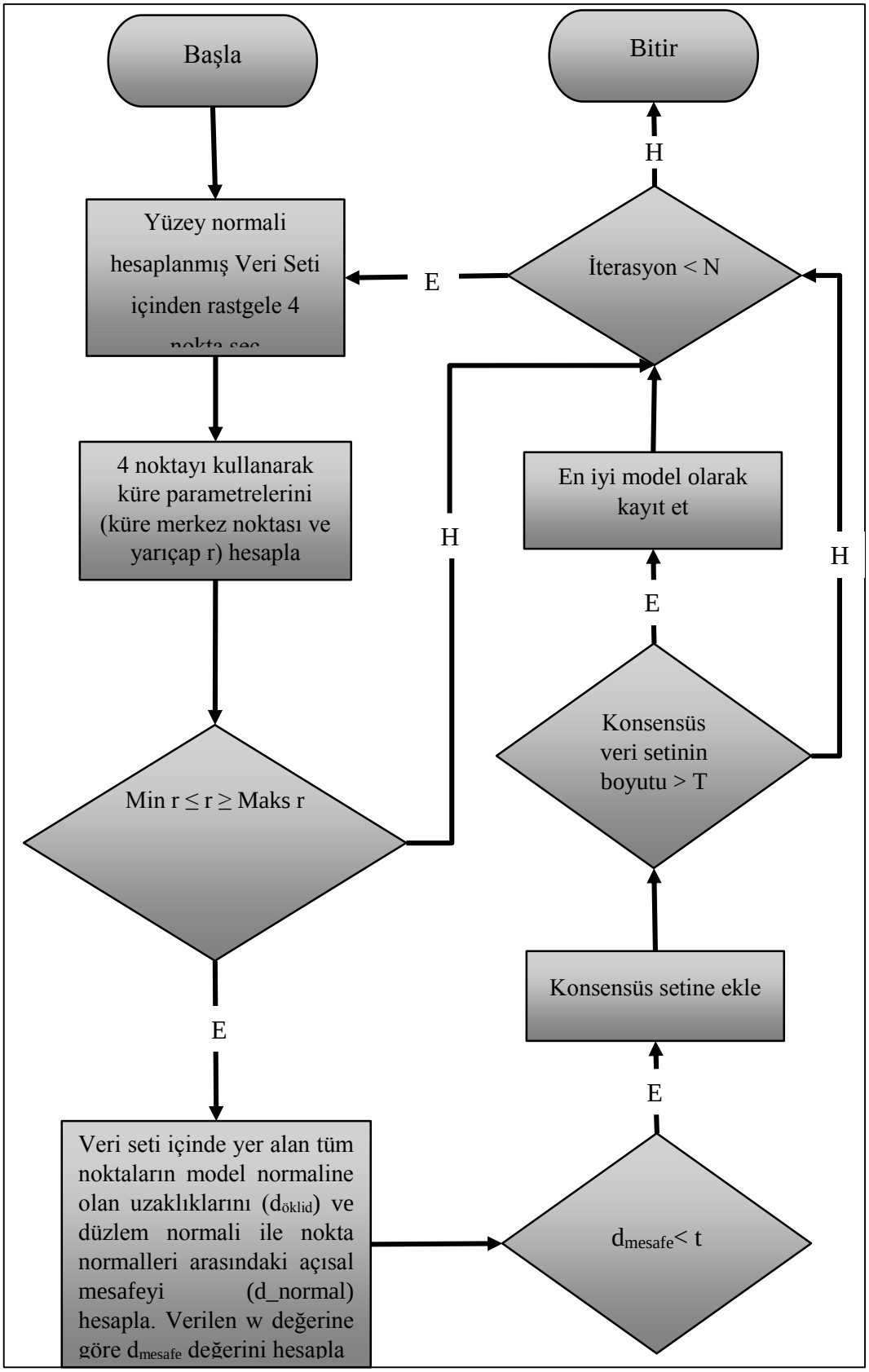
PCL kütüphanesini kullanarak RANSAC algoritması ile küre yüzey çıkarma işleminde rastgele 4 nokta seçilmekte ve seçilen bu noktalar ile çıkarılacak küre modelin parametreleri hesaplanmaktadır. Hesaplanan parametrelere göre bir küre model oluşturulmaktadır. Model oluşturma aşamasında algoritmanın uygulandığı veri seti içinden aranan küre modelin belirlenebilmesi için küre için izin verilen

maksimum ve minimum yarıçap sınır değerlerinin kullanıcı tarafından tanımlanması gerekmektedir. Yarıçap sınır değerlerinin ne olduğu ile ilgili bilgi silindir yüzey çıkarma bölümünde anlatılmıştır.

Seçilen 4 nokta ve girilen minimum maksimum yarıçap aralıklarında küre model belirlenmektedir. Nokta bulutu içinde belirlenen modele düşen noktaları belirlemek için model ile noktalar arasındaki izin verilen maksimum t eşik mesafesi ve model ile nokta arasında hesaplanacak mesafe için normal etkisini belirleyen w değerleri deneysel olarak belirlenir.

Küre modele atanacak noktalar ile model arasındaki mesafesinin belirlenmesi için ilk önce, nokta ile küre merkezi arasındaki mesafe hesaplanır. Daha sonra ise hesaplanan mesafeden küre yarıçapı çıkarılarak model ile noktalar arasındaki mesafe elde edilir.

Nokta bulutu içinden bir adet küre yüzeyin çıkarılmasına ilişkin iş akış diyagramı Şekil 4.18'de gösterilmiştir. Küre yüzey çıkarma işleminde nokta bulutuna RANSAC algoritması uygulanmadan önce her bir noktanın yüzey normalleri hesaplanmaktadır. PCL kütüphanesi kullanılarak noktaların yüzey normallerinin hesaplanması yüzey normaline bağlı düzlem çıkarma bölümünde anlatılmıştır.



Şekil 4.18. RANSAC algoritması ile tek bir yüzey çıkarmak için izlenen iş akış diyagramı

PCL kütüphanesini kullanarak RANSAC algoritması ile küre yüzey çıkarmada parametre tanımlama fonksiyonları silindir ve düzlem yüzey çıkarma bölümlerinde anlatılmıştır. Şekil 4.19'da C++ ortamında PCL kütüphanesini kullanarak yüzey normaline bağlı olarak küre yüzey çıkarma işlemi için model ve kullanıcı parametre tanımlama örneği gösterilmiştir.

```
seg.setModelType (pcl::SACMODEL_NORMAL_SPHERE);
seg.setMethodType (pcl::SAC_RANSAC);
seg.setNormalDistanceWeight (0.1);
seg.setMaxIterations (10000);
seg.setDistanceThreshold (0.02);
seg.setRadiusLimits (0, 0.2);
```

Şekil 4.19. Küre yüzey çıkarmak için kullanıcı tarafından tanımlanan parametreler

4.2.3.5. Koni Yüzey Çıkarma

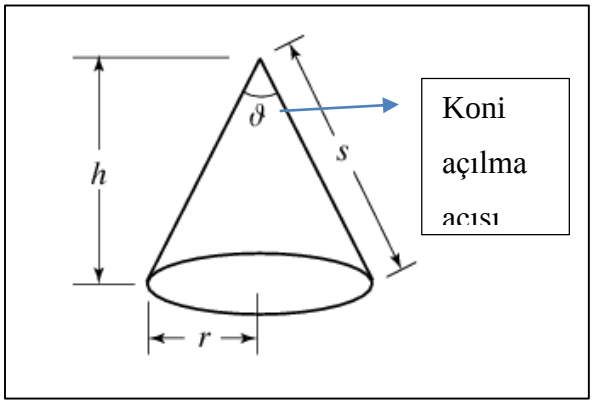
RANSAC algoritması ile nokta bulutu içinden yüzey normallerini de hesaplayarak koni yüzey çıkarmak için koni modelin oluşturulmasında kullanılacak yedi adet parametrenin hesaplanması gerekmektedir. Bu parametreler koninin tepe noktasına ait üç adet (X_T, Y_T, Z_T) koordinatları, koninin eksen yönünü belirlemek için kullanılacak, eksen üzerinde yer alan bir P_1 noktasının 3 adet (X_1, Y_1, Z_1) koordinatları ve koninin açılma açısı (θ) değerleridir.

Koni parametrelerini hesaplamak için nokta bulutu içinden rastgele 3 adet nokta seçilmekte ve koni parametreleri hesaplanmaktadır. Hesaplanan parametrelere göre yüzey çıkarımında kullanılacak bir koni model oluşturulmaktadır. Oluşturulan koni modelin nokta bulutu içinden yüzey çıkarımı için kullanılabilmesi için kullanıcı tarafından tanımlanacak bazı parametreleri karşılaması gerekmektedir. Bu parametreler koninin maksimumu ve minimum açılma açısı, koni modelin aranacağı eksen doğrultusu ve epsilon açısıdır. Eksen doğrultusu ve epsilon açısının tanımlamaları silindir yüzey çıkarma bölümünde yapılmıştır. Bu bölümde sadece koni açılma açısının tanımı yapılacaktır.

Koni Açılma Açısı: Koni açılma açısı koninin tepesinden itibaren tabana doğru genişlemesini sağlayan tepe açısıdır (Şekil 4.20). RANSAC algoritması ile koni yüzey çıkarmada koni açılma açısı rastgele seçilen 3 nokta kullanılarak hesaplanmaktadır. Açılma açısının işlem yapılan nokta bulutu içinde yer alan koni

yüzeyler için alabileceği maksimumu ve minimum değerler kullanıcı tarafından tanımlanması gerekmektedir.

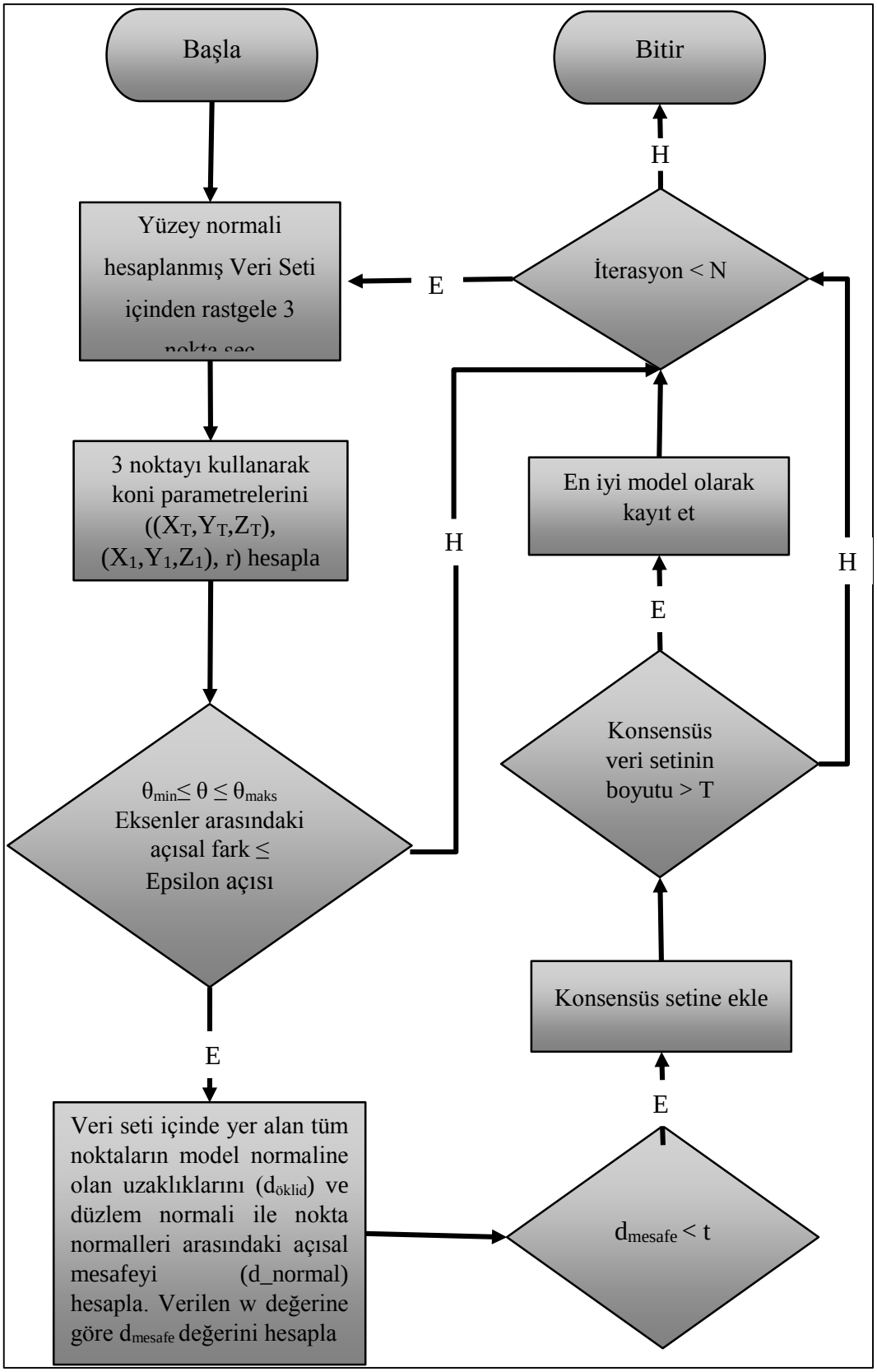
Koni modele, atanacak noktaların belirlenmesinde kullanılacak mesafe değeri için, ilk önce noktalar ile koni ekseni arasındaki mesafe hesaplanır. Mesafe hesaplama işlemi w normalin etki ağırlığıda dikkate alınarak yapılır. Daha sonra ise hesaplanan mesafeden noktanın bulunduğu konumdaki koninin yarıçap değeri çıkarılarak nokta ile model arasındaki mesafe hesaplanır.



Şekil 4.20. Koni açılma açısı

Yukarıda bahsi geçen parametreler dikkate alınarak hesaplanan koni model üzerine düşen noktalar kullanıcı tarafında belirlenen t eşik değeri mesafesi ve w normal uzunluk ağırlığına göre modele atanmaktadır.

Nokta bulutu içinden bir adet koni yüzeyin çıkarılmasına ilişkin iş akış diyagramı Şekil 4.21'de gösterilmiştir. Koni yüzey çıkarma işleminde nokta bulutuna RANSAC algoritması uygulanmadan önce her bir noktanın yüzey normalleri hesaplanmaktadır. PCL kütüphanesi kullanılarak noktaların yüzey normallerinin hesaplanması düzlem yüzey çıkarma bölümünde anlatılmıştır. Bu çalışmada RANSAC algoritması koni yüzey çıkarımı için nokta bulutuna ardışık olarak uygulanmıştır. Uygulanan algoritma en fazla noktaya sahip koni yüzeyi nokta bulutu içinden çıkarılmıştır. Algoritma, Geriye kalan nokta bulutuna, mevcut tüm konileri bulana kadar ardışık olarak uygulanmıştır.



Şekil 4.21. RANSAC algoritması ile tek bir koni yüzey çıkarmak için izlenen iş akışı

PCL kütüphanesini kullanarak RANSAC algoritması ile koni yüzey çıkarmada kullanıcı parametresi tanımlama fonksiyonlarının büyük çoğunluğu silindir, küre ve düzlem çıkarmada kullanılan fonksiyonlarla aynıdır. Koni yüzey çıkarmada diğer geometrik şekillerden farklı olarak maksimum ve minimum koni açılma açısının tanımlandığı setMinMaxOpeningAngle() fonksiyonu kullanılmaktadır. Bu fonksiyonda koni açılma açısı radyan biriminden tanımlanmaktadır. Şekil 4.22'de koni yüzey çıkarma için, C++ ortamında PCL kütüphanesini kullanarak örnek tanımlamalar yapılmıştır.

```
seg.setModelType (pcl::SACMODEL_CONE);
seg.setNormalDistanceWeight (0.1);
seg.setMinMaxOpeningAngle(5.0 / 180.0  * M_PI, 85.0/ 180.0 * M_PI);
seg.setAxis(Eigen::Vector3f (0, 0, 1));
seg.setEpsAngle(0.1);
seg.setMethodType (pcl::SAC_RANSAC);
seg.setMaxIterations (10000);
seg.setDistanceThreshold (0.03);
```

Şekil 4.22. Koni yüzey çıkarma için kullanıcı tarafında tanımlanan parametreler

5. ARAŞTIRMA BULGULARI

5.1. Test Verisi İçinden Geometrik Yüzeylerin Çıkarılması

Tez çalışması kapsamında geometrik olarak tanımlanabilen silindir, koni, düzlem ve küre gibi nesnelerin lazer tarama veresi içinde otomatik olarak çıkarılması işlemi gerçekleştirilmiştir. Bu işlem için laboratuvar ortamında geometrik şekilleri temsil eden cisimler kullanılarak bir veri düzeneği hazırlanmıştır. Hazırlanan düzenek yersel lazer tarayıcı ile taranarak örnek bir test veri seti elde edilmiştir (Şekil 5.1).

Şekil 5.1. Laboratuvar ortamında hazırlanan tarama düzeneği

Test düzeneğinde, düzlem yüzey için çalışma masası, silindir yüzey için kâğıt rulo, koni yüzey için trafik konisi ve küre yüzey için plastik top kullanılmıştır (Şekil 5.2).

Çalışma masası (Düzlem)

Trafik Konisi (Koni)

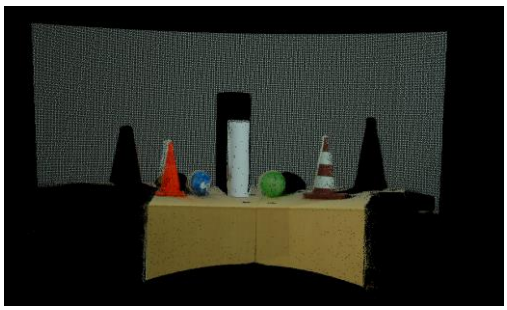
Kâğıt Rulo (Silindir)

Plastik Top (Küre)

Şekil 5.2. Çalışma kapsamında kullanılan geometrik nesneler

Tarama işlemi Riegl LMS 390i lazer tarayıcı ile yapılmıştır. Bu tarayıcı ile 0.08° açısal çözünürlükte tarama işlemi gerçekleştirilmiştir. Tarama işlemi tek bir noktadan yapılmış ve elde edilen noktaları görsel olarak sunmak için tarama işlemi sonunda fotoğraf çekilerek nokta bulutu renklendirilmiştir. Elde edilen nokta bulutu verisi içinden test nesnelerini içeren alan RiScan Pro yazılımında çıkartılarak ayrı bir nokta verisi olarak kayıt edilmiştir. Ayrıştırılan veriler ASCII veri formatında X, Y, Z değerlerini içerecek şekilde dışa aktarılmıştır. Şekil 5.3a'de ham nokta bulutu, Şekil 5.3b'de ise test için ham nokta bulutundan ayrıştırılmış nokta bulutu görülmektedir.

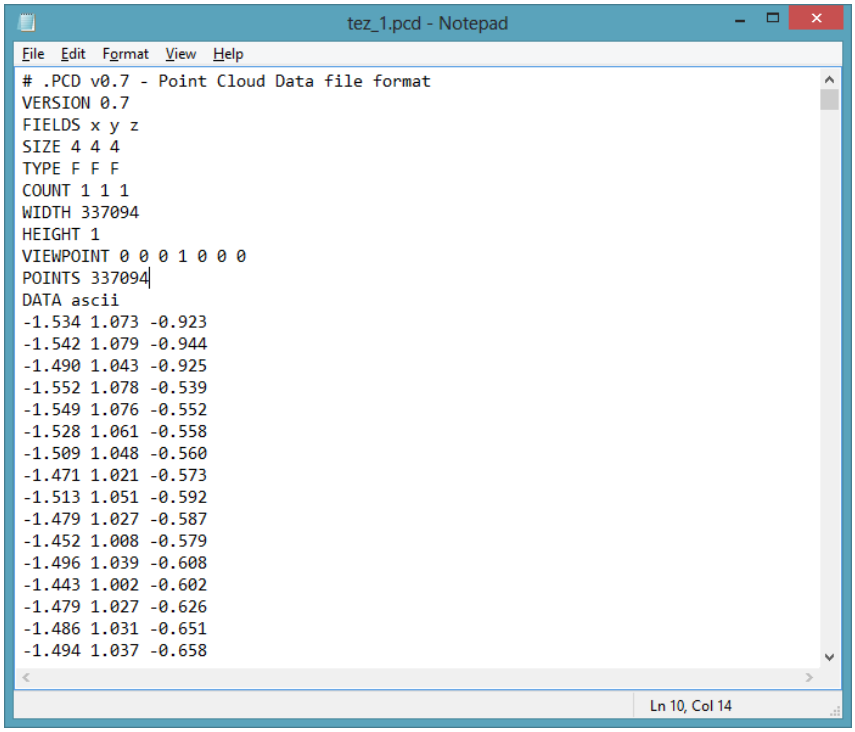
a



b

Şekil 5.3. a:Tarama sonucu elde edilen renklendirilmiş ham nokta bulutu; b: test için ayrılan nokta bulutu

ASCII formatında dışarı aktarılan noktalar PCL kütüphanesinde kullanılan PCD (Point Cloud Data file format: Nokta bulutu verisi veri formatı) nokta dosyası formatına dönüştürülmüştür. Şekil 5.4'de örnek bir PCD dosya formatı gösterilmiştir. PCD dosyasında nokta bulutu içinde yer alan noktaların sadece X, Y ve Z koordinatları yer almaktadır. EK-2'de PCD dosya formatına ilişkin detaylı bilgi yer almaktadır.



Şekil 5.4. PCD nokta formatı ve test verisinde yer alan koordinat değerleri

PCD formatında bulunan test verisinde toplamda 337094 adet nokta bulunmaktadır. Bu veri seti içinde yer alan düzlem, silindir, koni ve küre yüzeyleri RANSAC algoritması uygulanarak çıkarılmıştır. Algoritma ilk önce PCL kütüphanesini kullanılarak her bir geometrik yüzey için ayrı ayrı C++ ortamında programlanmıştır. Daha sonra tüm yüzeyler tek bir programla çıkarılacak şekilde RANSAC algoritması kodlanmıştır.

5.1.1. Düzlem Yüzey Çıkarma

5.1.1.1.Yüzey Normalinden Bağımsız Düzlem Yüzey Çıkarma

Test veri seti üzerine ilk önce RANSAC algoritması ile düzlem çıkarma işlemi uygulanmıştır. Düzlem çıkarma işlemi, nokta bulutu içinde yer alan noktaların yüzey normallerinden bağımsız olarak sadece t eşik değeri mesafesine göre gerçekleştirilmiştir. Veri seti içinde birden fazla düzlem olduğu için algoritma veri setine ardışık olarak uygulanmıştır. EK-3'de PCL kütüphanesini kullanarak C++ ortamında yazılan yüzey normalinden bağımsız düzlem yüzey çıkarma kodları sunulmuştur.

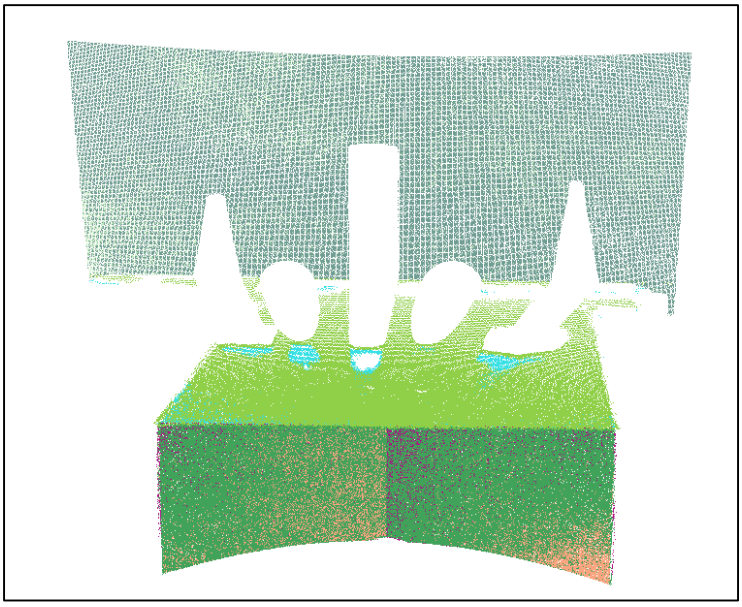
RANSAC algoritması kullanarak nokta bulutu içinden yüzey çıkarımından belirlenmesi gereken en önemli parametre t eşik değeri mesafesidir. Yersel lazer tarayıcılar ile tarama işlemi gerçekleştirilirken gerek taranan objenin yüzey pürüzlülüğü, gerekse lazer ışınının yüzeyden saçınım özelliklerinden dolayı nokta bulutları obje yüzeyinde bir kalınlık oluşturmaktadır. Yüzey üzerindeki noktaların, çıkarılan model üzerine atanması için t eşik değeri mesafesinin doğru bir şekilde tespit edilmesi gerekmektedir. Bu çalışmada test verisi içinde düzlem yüzeyler çıkarılırken düzlem yüzeyler için uygun t değeri deneysel olarak belirlenmiştir. Bu kapsamda t değerine farklı değerler verilerek RANSAC algoritması veri setine uygulanmıştır.

Çalışma kapsamında RANSAC algoritması t eşik değeri mesafeleri t= 0.005 m, t = 0.01 m, t= 0,015 m ve t= 0.02 m seçilerek veri setine uygulanmıştır. Uygulanan algoritma ile çıkarılan düzlem yüzeylere ait bilgilere Çizelge 5.1'de gösterilmiştir. Algoritma ile çıkarılması beklenen toplamda üç adet düzlem yüzey vardır. Bu yüzeyler masanın ön yüzeyi, masanın üst yüzeyi ve karşı duvar yüzeyidir.

Çizelge 5.1. Farklı t eşik değeri mesafesine göre test verisinden çıkarılan yüzey sayısı

t= eşik değeri mesafesi (m)	Çıkarılan Düzlem Sayısı
0.005	6
0.01	3
0.015	3
0.02	3

Eşik değeri t= 0.005 metre seçildiğinde nokta bulutu içinden toplamda 6 adet düzlem yüzey çıkarılmıştır. Bu düzlem yüzeyler farklı renklerle renklendirilerek Şekil 5.5'de gösterilmiştir. Şekilden de anlaşılacağı gibi t =0.005 değeri için masanın ön cephesi 3 farklı düzlem, üst bölümü ise 2 farklı düzlem olarak çıkarılmıştır, karşı duvar ise 1 düzlem yüzey olarak çıkarılmıştır.



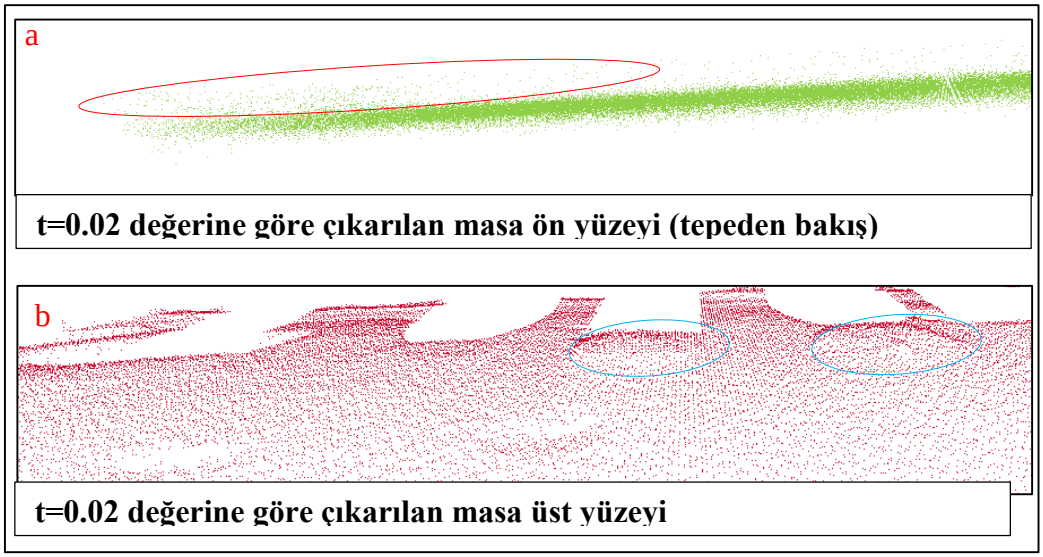
Şekil 5.5. t=0.005 metre değeri için test veri setinden çıkarılan düzlem yüzeyler

t =0.01 m, t = 0.015m, t = 0.02 değerleri için ise 3 adet düzlem çıkarılmıştır. Çıkarılan en fazla noktaya sahip birinci düzlem taranan masanın ön bölümüne ait düzlem yüzeydir. Bu alan tarayıcıya yakın olmasından dolayı fazla nokta verisi içermektedir. Çıkarılan ikinci düzlem duvara ait düzlem yüzeydir. Çıkarılan üçüncü düzlem ise masanın üstüne ait düzlem yüzeydir. Farklı t değerlerine göre çıkarılan düzlemlerin sahip oldukları nokta sayıları Çizelge 5.2'de gösterilmiştir.

Çizelge 5.2. RANSAC algoritması ile çıkarılan düzlemelere ait bilgiler

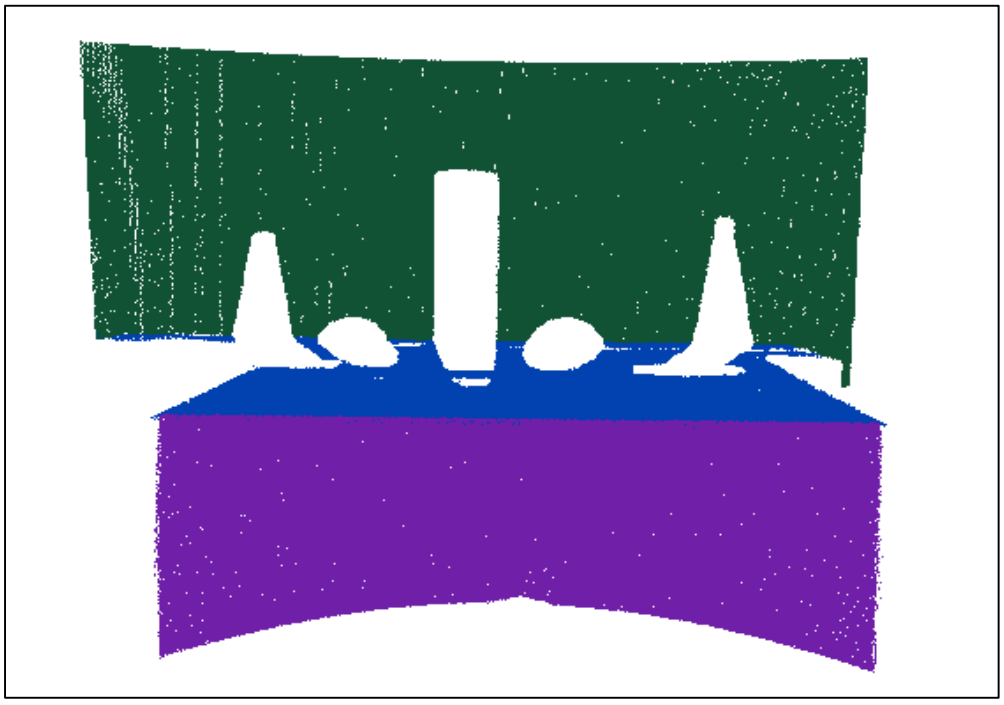
Düzlem No	Nokta Sayısı		
	t = 0.01 m	t = 0.015 m	t = 0.02 m
1. Düzlem	165294	167879	168577
2. Düzlem	82978	83347	83406
3. Düzlem	66026	66140	66100

Veri seti içinde düzlem yüzeyler çıkarılırken t eşik değeri artırıldığından düzleme fazladan noktalar atanmaktadır. Örneğin t = 0.01 ve t = 0.02 eşik değerlerine göre masanın ön yüzeyine ait birinci düzlemler arasında 3283 adet nokta farkı vardır. t = 0.02 m eşik değerine göre birinci düzlem yüzey çıkarılırken masanın üst yüzeyinde yer alan noktalardan, birinci düzleme 2 cm yakınlıkta olan noktalar da düzleme atanmaktadır (Şekil 5.6a). Kırmızı daire içindeki noktalara masanın üst yüzeyinden ön yüzeyine atanan noktaları göstermektedir. Bununla birlikte üçüncü düzlemin birinci düzlemle kesiştiği bölgelerde nokta kayıpları yaşanmaktadır. Benzer şekilde masanın üst bölgesine ait üçüncü düzlem yüzey çıkarılırken masanın üzerinde yer alan objelere ait noktalarda düzleme atanmaktadır (Şekil 5.6b). Mavi daire içinde yer alan noktalar masanın üstünden yer alan trafik konisi ve top cismine ait noktaları göstermektedir. Bundan dolayı t = 0.01 m değeri test verisi içinden düzlem yüzeylerin çıkarılmasında ideal t eşik mesafesidir.



Şekil 5.6. t = 0.02 m değeri için birinci ve üçüncü düzlem yüzeylere atanan fazlalık noktalar.

Test verisi için ideal eşik mesafesi olarak t = 0.01 m değerine göre çıkarılan düzlemler farklı renklerle renklendirilerek Şekil 5.7'de gösterilmiştir.



Şekil 5.7. Test veri seti içinde t = 0.01 değeri için çıkarılan düzlemler

Model çıkarımı için modeli tipi, yöntem, t eşik mesafesi ve iterasyon sayısı Şekil 5.8 deki gibi belirlenmiştir. İterasyon sayısı Eşitlik 3.8'e göre hesaplanmıştır. p doğru model seçme olasılığı 0.99 ve veri seti içindeki ayrıkırı değer oranını %80 - %85 arası kabul edilerek iterasyon sayısı 1000 olarak belirlenmiştir.

```
// düzlem yüzey çıkarımı için yüzey tipi, yöntem,
//iterasyon sayısı ve eşik değeri mesafesi (t) nin girilmesi
seg.setOptimizeCoefficients (true);
seg.setModelType (pcl::SACMODEL_PLANE);//model tipi
seg.setMethodType (pcl::SAC_RANSAC);//yöntem
seg.setMaxIterations (1000);//iterasyon sayısı
seg.setDistanceThreshold (0.01); //t eşik değeri mesafesi
```

Şekil 5.8. Düzlem model belirlemek için parametre tanımlama

RANSAC algoritması ile yüzey normalinden bağımsız olarak otomatik düzlem yüzey çıkarma işlemi zamansal açıdan incelenecek olursa, test verisi içindeki 3 adet düzlem yüzey 30 saniye gibi kısa bir sürede çıkarılmıştır.

5.1.1.2. Yüzey Normaline Bağımlı Düzlem Yüzey Çıkarma

RANSAC algoritması ile yüzey normaline bağlı olarak düzlem yüzey çıkarımı için ilk önce test verisi içinde yer alan tüm noktaların yüzey normallerinin hesaplanması gerekmektedir. Bu kapsamda PCL kütüphanesini kullanarak nokta bulutu içindeki noktaların yüzey normalinin hesaplandığı ve hesaplanan yüzey normallerine bağlı olarak düzlem yüzeylerin çıkarıldığı C++ kodları EK-4 sunulmuştur. RANSAC algoritması test verisine ardışık olarak uygulanmıştır.

PCL kütüphanesini kullanarak test verisi içinde yer alan noktalara ait yüzey normalleri hesaplanırken yüzey normali hesaplanacak noktanın komşularını elde etmek için yarıçapa göre arama yapılmıştır. Bu kapsamda yarıçap mesafesi 0.03 metre olarak belirlenmiştir. Yüzey normali hesaplanacak nokta merkez alınarak 3 cm daire içinde kalan noktalar belirlenmiş ve belirlenen bu noktalar kullanılarak en küçük kareler düzlem yakalama algoritmasına göre yüzey normalleri hesaplanmıştır. EK-4'de sunulan C++ kodunda yer alan, yüzey normallerinin hesaplandığı kodlar Şekil 5.9 gösterilmiştir.

```
//Nokta normallerini hesapla
ne.setSearchMethod (tree);
ne.setInputCloud (cloud);
ne.setRadiusSearch(0.03);
ne.compute (*cloud_normal);
```

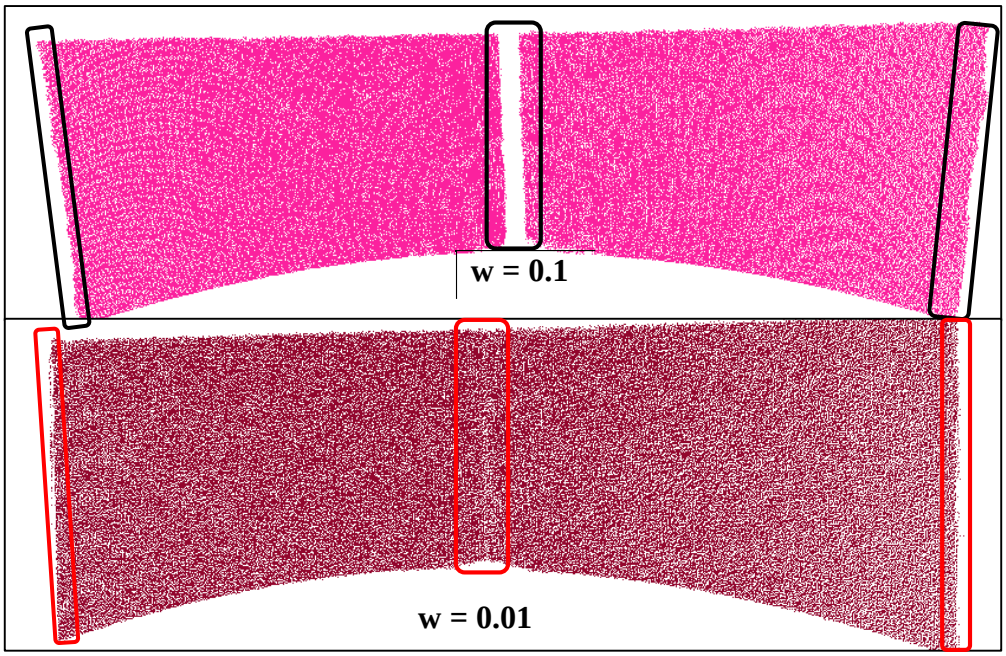
Şekil 5.9. Test verisi içinde yer alan noktaların yüzey normallerinin hesaplanması

RANSAC algoritması ile yüzey normaline bağlı olarak düzlem yüzey çıkarımı işlemi için tanımlanması gereken t eşik değeri mesafesi ve normal mesafe ağırlığı w deneysel olarak belirmiştir. t eşik değeri mesafesi normalden bağımsız düzlem çıkarma bölümünde belirlenen 0.01 m olarak alınmıştır. Test verisi için uygun w değerini belirlemek için w'ye farklı değerler vererek RANSAC algoritması nokta bulutuna uygulanmıştır. Yapılan denemeler sonucunda test verisi için uygun ağırlık değerinin 0.01 olduğu gözlemlenmiştir. Farklı w değerlerine göre çıkarılan düzlemler ve nokta sayıları Çizelge 5.3'de gösterilmiştir.

Çizelge 5. 3. Farklı w değerlerine göre test verisi içinden çıkarılan düzlemlere ait nokta sayıları

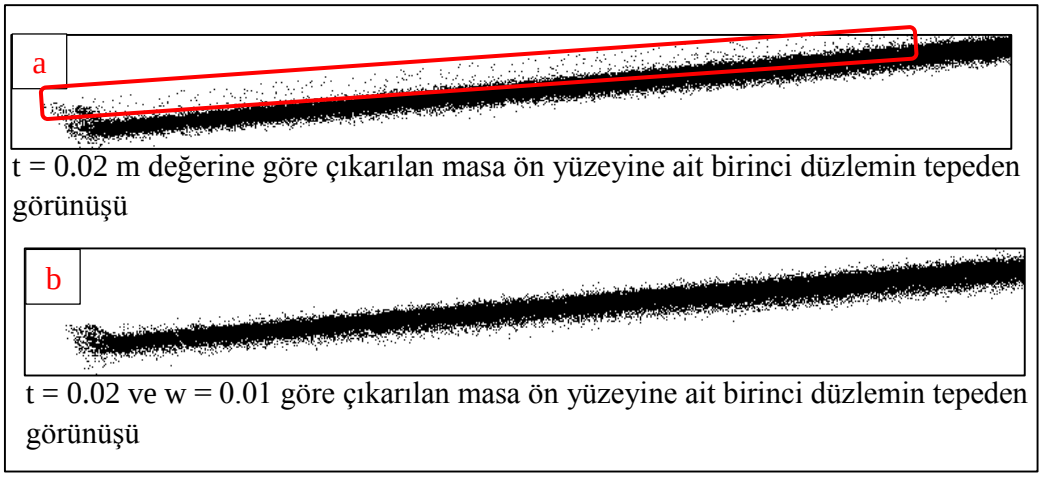
Düzlem No	**Nokta Sayısı**			
	w = 0.01	**w = 0.02**	**w= 0.05**	**w = 0.1**
1. Düzlem	163032	160542	153802	148211
2. Düzlem	82831	82380	81069	74269
3. Düzlem	65204	63481	60693	57081

w = 0.1 ve w = 0.01 değerlerine göre çıkarılan birinci düzlem karşılaştırıldığında w = 0.1 değerine göre çıkarılan birinci düzlemin kenarlarında ve düzlem ortasında çok sayıda noktanın düzleme atanmadığı görülmektedir. Test verisi içinde yer alan masa, iki farklı masanın birleşiminden oluşmaktadır. w = 0.1 değerine göre düzleme atanmayan noktaların masa birleşiminde ve düzlemin kenarlarında olduğu görülmektedir. w = 0.01 göre yapılan düzlem çıkarma işleminde de düzleme atanmayan noktalar mevcuttur ancak bu noktalar düzlemin genel yapısını bozmamaktadır. Bununla birlikte masa kenarlarında meydana gelen saçınım daha azdır (Şekil 5.10).



Şekil 5.10. w = 0.01 ve w = 0.1 değerlerine göre çıkarılan düzlemlerin karşılaştırılması

Yukarıda anlatılanlara ek olarak yüzey normalinden bağımsız yapılan düzlem çıkarım bölümünde test verisinden t = 0.02 m değerine göre düzlem yüzeyler çıkarılmıştır. t = 0.02 değerine göre çıkarılan düzlemlerden masa ön yüzeyine ait birinci düzlem incelendiğinde, masa üst yüzeyine ait noktaların ön yüzeye atandığı gözlemlenmişti (Şekil 5.6). Bu bölümde t = 0.02 m değerine ve yüzey normal ağırlığı w = 0.01 değerine göre yüzey çıkarım işlemi tekrar yapılmıştır. Yüzey normalinden bağımsız t = 0.02 m göre çıkarılan birinci düzlemle bu bölümde t = 0.02 ve w = 0.01 çıkarılan birinci düzlem karşılaştırıldığında, daha az sayıda masa üstüne ait noktaların masa ön yüzüne ait düzleme atandığı gözlemlenmiştir. Şekil 5.11a'da t = 0.02 m değerine göre çıkarılan masa ön yüzeyine ait birinci düzlemin tepeden görünüşü gösterilmiştir. Kırmızı ile işaretlenen noktalar masa üst yüzeyine ait noktalardır. Şekil 5.11b'de ise t = 0.02 ve w = 0.01 göre çıkarılan masa ön yüzeyine ait birinci düzlemin tepeden görünüşü gösterilmiştir. Şekil incelendiğinde çok az sayıda masa üst yüzeyine ait noktanın ön yüzeye atandığı görülecektir.



Şekil 5.11. a: Yüzey normalinden bağımsız çıkarılan düzlem; b: Yüzey normaline bağımlı olarak çıkarılan düzlem

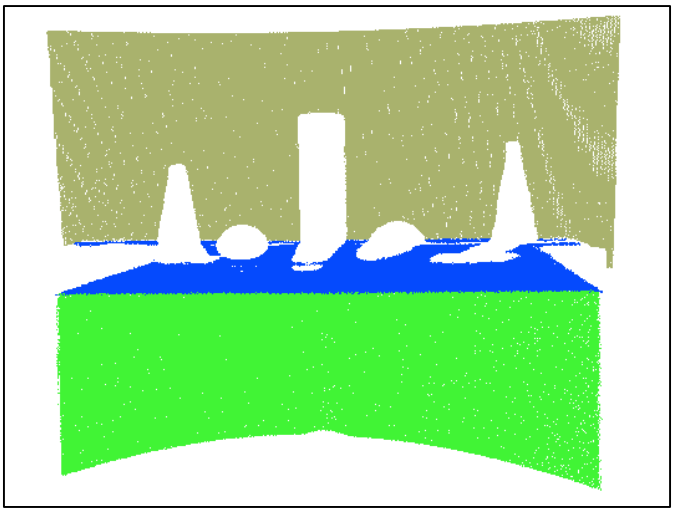
Yüzey normali ile birlikte test verisi için uygun w değeri 0.01 olarak belirlenmiştir. EK-4'de yer alan ve yüzey normaline bağlı olarak düzlem yüzey çıkarmada PCL kütüphanesine kullanarak tanımlanan model çıkarım parametreleri Şekil 5.12'de gösterilmiştir.

```
// duzlem yüzey için segmentasyon parametrelerini gir
 seg.setOptimizeCoefficients (true);
 seg.setModelType (pcl::SACMODEL_NORMAL_PLANE);
 seg.setMethodType (pcl::SAC_RANSAC);
 seg.setNormalDistanceWeight (0.01);
 seg.setMaxIterations (1000);
 seg.setDistanceThreshold (0.01);
```

Şekil 5.12. Yüzey normaline bağımlı düzlem yüzey çıkarım parametreleri

İterasyon sayısı, p doğru model seçme olasılığı 0.99 ve veri seti içindeki aykırı değer oranını %80 - %85 arası kabul edilerek iterasyon sayısı Eşitlik 3.8'e göre 1000 olarak belirlenmiştir.

Yukarıda verilen değişkenlere göre test verisinden otomatik olarak çıkarılan 3 adet düzlem yüzey Şekil 5.13'de farklı renklerle renklendirilerek gösterilmiştir. Düzlemlerin sahip olduğu noktalar Çizelge 5.3'de w = 0.01 değerine göre gösterilmiştir.



Şekil 5.13. t= 0.01 ve w =0.01 değerlerine göre çıkarılan düzlem yüzeyler

Test verisinde RANSAC algoritması ile yüzey normaline bağlı olarak çıkarılan yüzeyler zamansal olarak değerlendirildiğinde, t= 0.01 ve w =0.01 değerlerine göre düzlem yüzeyler 290 saniyede çıkarılmıştır. Bu zamanın büyük bir bölümü yüzey normallerinin hesaplanması için harcanmıştır.

5.1.2. Silindir Yüzey Çıkarma

Test veri seti içinde bir adet kâğıt rulo ile temsil edilen bir adet silindir nesnesi yer almaktadır. Bu nesneyi RANSAC algoritması uygulanarak veri seti içinden otomatik olarak çıkarılmıştır. Silindir yüzey çıkarma işleminde algoritma yüzey normaline bağlı olarak uygulanmıştır. Algoritma uygulanırken ilk önce veri seti içinde yer alan tüm noktaların yüzey normalleri hesaplattırılmış daha sonra ise belirlenen model parametrelerine göre silindir yüzey veri seti içinden çıkarılmıştır. PCL kütüphanesi kullanılarak RANSAC algoritması ile silindir yüzey çıkarma C++ kodları EK-5'de sunulmuştur.

Silindir yüzey çıkarma işleminde uygun modelin seçilebilmesi için kullanıcı tarafında epsilon açısı, silindirin minimum ve maksimum yarıçap değerleri ve silindirin aranacağı eksen bilgisinin tanımlanması gerekmektedir. Bunlara ek olarak belirlenen silindir modele hangi noktaların atanacağını belirlemek için gerekli olan t eşik değeri mesafesi ve w yüzey normal ağırlığı değerlerinin belirlenmesi gerekmektedir. Test verisi içinden kağıt ruloyu temsil eden noktaların çıkarılması için bu parametrelere farklı değerler verilerek deneysel denemeler sonucunda

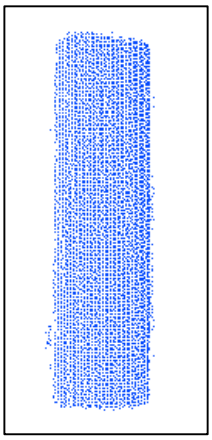
belirlenmiştir. Bu kapsamda silindir yüzey veri seti içinde Z ekseni etrafında aranacağından setAxis(Eigen::Vector3f (0,0,1)) olarak tanımlanmıştır. Burada tanımlanan 1 değeri Z eksenini göstermektedir. Silindir yüzeyin çıkarılması için kullanılan diğer parametre değerleri Şekil 5.14'de gösterilmiştir.

```
seg.setOptimizeCoefficients (true);
seg.setModelType (pcl::SACMODEL_CYLINDER);
seg.setMethodType (pcl::SAC_RANSAC);
seg.setNormalDistanceWeight (0.1);
seg.setMaxIterations (1000);
seg.setEpsAngle(0.2);
seg.setAxis(Eigen::Vector3f (0,0,1));
seg.setDistanceThreshold (0.01);
seg.setRadiusLimits (0, 0.25);
```

Şekil 5.14. Silindir yüzey çıkarmada uygulanan model parametreleri

Veri seti içinde silindir yüzey üzerine düşebilecek çok az sayıda nokta (yaklaşık 6 000) mevcuttur. Bundan dolayı iterasyon sayısı p doğru model seçme olasılığı 0.99 ve veri seti içindeki aykırı değer oranını %95 kabul edilerek iterasyon sayısı Eşitlik 3.8'e göre 2000 olarak belirlenmiştir

Belirlenen değerlere göre algoritma veri setine ardışık olarak uygulanmıştır. Ardışık olarak uygulana algoritma ile silindir kâğıt ruloyu temsil eden noktalar veri setinden çıkarılmıştır. Silindir kâğıt rulo için 5678 adet nokta ile temsil edilmektedir.



Şekil 5.15. RANSAC algoritması ile çıkarılan silindir model

Çıkarılan yüzey zamansal olarak değerlendirildiğinde, nokta bulutu içinden silindir yüzeyin çıkarılması, 5.1.1.2 bölümünde çıkarılan düzlem yüzeylerden daha fazla zaman almıştır. Silindir yüzey, veri seti içinden 522 saniyede çıkarılmıştır.

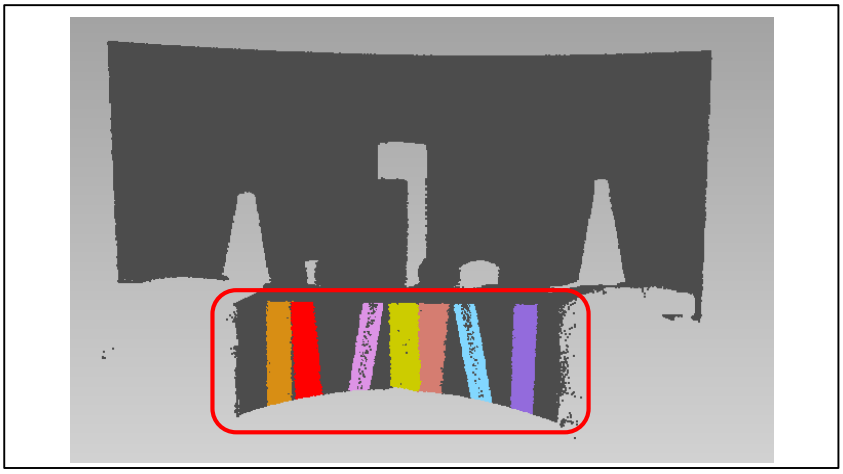
EK-5'te sunulan silindir yüzey çıkarma kodlarında nokta bulutundan ardışık silindir yüzey çıkarmak için veri seti bir while döngüsüne tabi tutulmaktadır. While döngüsünün bitmesi için koşulan şart ise girdi nokta bulutu içindeki nokta sayısının belirli bir yüzdenin altına düşmesidir. Bu yüzde değerinin belirlenmesi veri seti içinden yüzeylerin doğru bir şekilde çıkarılması için son derece önemlidir. Silindir yüzey için bu değer % 99 olarak belirlenmiştir. Bu değer, veri setine göre tahmini olarak belirlenmektedir. Örneğin bu uygulamada silindir yüzey yaklaşık 6000 noktaya sahiptir. Test verisi içinde 337094 adet nokta vardır ve silindir yüzeyin sahip olduğu noktalar test verisinin yaklaşık %1 ile %2 arasına karşılık gelmektedir. Bundan dolayı veri seti içinde algoritmanın bir defa dönüp bitmesi için bu değer %99 olarak verilmiştir. Eğer bu değer %95 gibi bir değer girilmiş olsaydı algoritma ikinci defa dönecek ve veri seti içinde silindir yüzeye sahip olmayan ancak oluşturulacak silindir modele, yüzeyin çok ince bir kesitini temsil edebilecek noktalar modele atanabilecekti.

While döngü değerinin belirlenmesi gibi model hesaplamada gerekli parametrelerinde doğru bir şekilde belirlenmesi son derece önemlidir. Parametre değerleri gerçekçi tanımlanmadığında oluşabilecek sorunları göre bilmek için RANSAC algoritması, veri setine while döngüsü orijinal verinin %80 kalana kadar dönecek şekilde aşağıdaki parametrelere göre uygulanmıştır (Şekil 5.16).

```
seg.setOptimizeCoefficients (true);
seg.setModelType (pcl::SACMODEL_CYLINDER);
seg.setMethodType (pcl::SAC_RANSAC);
seg.setNormalDistanceWeight (0.01);
seg.setMaxIterations (2000);
seg.setEpsAngle(0.2);
seg.setAxis(Eigen::Vector3f (0,0,1));
seg.setDistanceThreshold (0.015);
seg.setRadiusLimits (0, 0.25);
```

Şekil 5.16. Model parametrelerinin yanlış tanımlanması

Tanımlanan parametrelere göre veri seti içinden nokta sayıları 8000 ile 1400 arasında değişen toplam 7 adet silindir yüzey çıkarılmıştır. Bu yüzeylerin hepsi veri seti içinde yer alan masanın ön yüzeyine ait noktaların temsil ettiği silindir yüzeylerdir. Ancak gerçekte masanın ön yüzey sadece düzlem bir yüzeydir. Veri seti içinden çıkarılan hatalı silindir yüzeyler şekil 5.17'de farklı renklerle renklendirilerek gösterilmiştir.



Şekil 5.17. Yanlış tanımlanan parametreler sonucu çıkarılan hatalı silindir yüzeyler

5.1.3. Koni Yüzey Çıkarma

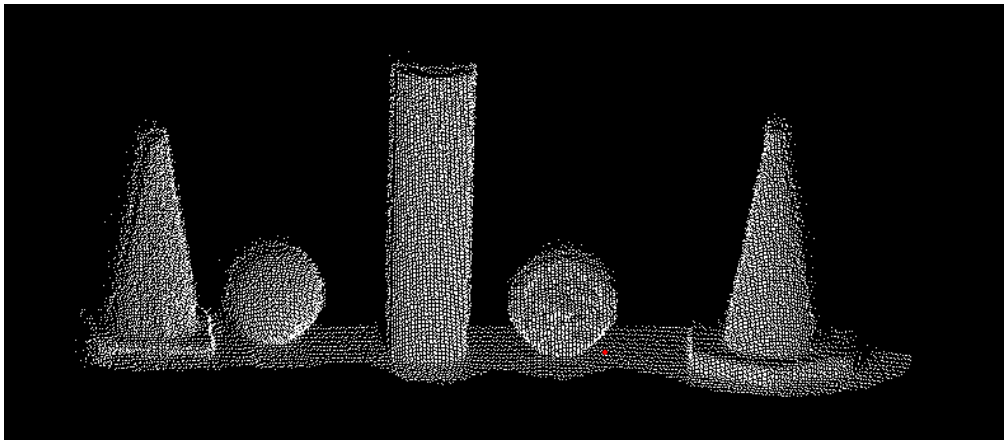
Test verisi içinde yer alan iki adet koni şekle ait noktaları RANSAC algoritması ile çıkarmak için, algoritma PCL kütüphanesini kullanarak kodlanmıştır. RANSAC algoritması ile koni yüzey çıkarılmasın ait C++ kodları EK-6 sunulmuştur. EK-6 sunulan kodlarla, koni yüzey çıkarım işlemi nokta normallerine bağlı olarak yapılmaktadır. RANSAC algoritması ile koni yüzey çıkarılmadan önce girdi olarak kullanılan veri seti içinde yer alan noktaların yüzey normalleri hesaplanmaktadır. Koni çıkarım işlemi için yüzey normali hesaplanacak noktaya komşu noktalar PCL kütüphanesinde yer alan KSearch fonksiyonu ile 50 adet olarak belirlenmiştir. Belirlenen bu noktalar k-en yakın komşuluk ilkesine göre nokta bulutu içinden bulunmaktadır. Bulunan noktalar kullanılarak en küçük kareler düzlem yakalama yöntemine bağlı olarak noktanın normali hesaplanmaktadır.

Normal hesaplama işleminden sonra koni yüzey çıkarımı için gerekli parametreler belirlenmektedir. Bu parametrelerden koninin tepe açısı 5 derece ile 85 derece arasında belirlenmiştir. Bu değerler radyan biriminde setMinMaxOpeningAngle() fonksiyona tanımlanmıştır. Koninin aranacağı eksen ise setAxis(Eigen::Vector3f (0,0,1)) olarak Z ekseni olarak tanımlanmıştır.

Test veri seti içinde koni nesneleri çok az sayıda nokta ile temsil edilmektedir. Veri seti içinde koni model için seçilecek bir verinin aykırı değer (ϵ)

yani koni üzerine düşmeme olasılığı yaklaşık %95 olarak kabul edilmiştir. Aykırı değerler % 95 olasılıkla hesaplanacak olursa bu değer 0.95* 337094 = 320240 adet noktaya denk gelmektedir. Bunun anlamı seçilen noktalarından, 320240 adetinin modele atanmama olasılığı vardır. Aykırı değer olma olasılığı $\epsilon = 0.95$ ve p doğru modelin seçilme olasılığı 0.99 kabul edilirse veri setine uygulanması gereken iterasyon sayısı Eşitlik 3.8'den $N = \log(1 - 0.99) / \log(1 - (1 - 0.95)^3)$ = 36840 olarak hesaplanmaktadır. PCL kütüphanesinde maksimum iterasyon sayısı 10000 olarak tanımlandığı için EK-6'da verilen algoritma test verisi içinden koni yüzey çıkarmada başarısız olmuştur.

Nokta bulutu içinden koni yüzey çıkarımı işlemi için Test verisi içinde sadece silindir, küre ve koni şekilleri kalacak şekilde ayrı bir test verisi oluşturulmuştur. Bu işlem için RiScan Pro yazılımı kullanılmıştır. Oluşturulan ikinci test verisi Şekil 5.18'de gösterilmiştir. Yeni test verisi 22526 adet nokta içermektedir.



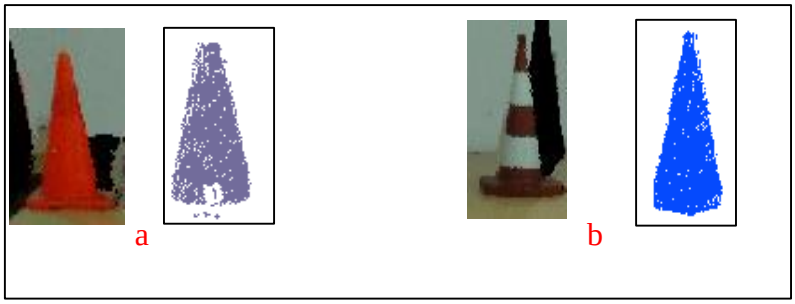
Şekil 5.18. Düzlem yüzey haricinde diğer geometrik şekilleri temsil eden veri seti

Oluşturulan veri seti için t eşik değeri mesafesi, w normal etki ağırlığı ve epsilon açısı farklı değerler verilerek veri seti için uygun değerler bulunmuştur. Koni yüzey çıkarımı için kullanılan parametreler Şekil 5.19'da gösterilmiştir. İterasyon değeri p = 0.99 ve $\epsilon = 0.85$ değeri için 1000 olarak belirlenmiştir.

```
// Koni yüzey için segmentasyon parametrelerini gir
seg.setOptimizeCoefficients (true);
seg.setModelType (pcl::SACMODEL_CONE);
seg.setMethodType (pcl::SAC_RANSAC);
seg.setNormalDistanceWeight (0.08);
seg.setMinMaxOpeningAngle(5.0 / 180.0  * M_PI, 85.0/ 180.0 * M_PI);
seg.setAxis(Eigen::Vector3f (0, 0, 1));
seg.setEpsAngle(0.1);
seg.setMaxIterations (1000);
seg.setDistanceThreshold (0.033);
```

Şekil 5.19. Koni çıkarımında tanımlanan parametreler

Şekil 5.19'da tanımlanan parametrelere göre 2 adet koni, ikinci veri seti içinden ardışık olarak çıkarılmıştır. Veri setinde sağ tarafta yer alan koni (Şekil 5.20a) için 3118 nokta sol taraftaki koni (Şekil 5.20b) için 2741 tane nokta çıkarılmıştır. Çıkarılan koniler Şekil 5.20'de gösterilmiştir.



Şekil 5.20. İkinci test verisinden çıkarılan koniler

Koni çıkarım işlemi zamansal olarak değerlendirilecek olursa 22526 adet noktaya sahip ikinci test verisinden 2 adet koni 8 saniyede çıkarılmıştır.

5.1.4. Küre Yüzey Çıkarma

Veri seti içinde küreyi temsil etmek için iki adet plastik top yer almaktadır. Bu toplara ait verileri RANSAC algoritması ile otomatik olarak çıkarmak için algoritma, PCL Kütüphanesi kullanılarak C++ ortamında kodlanmıştır. EK-7'de RANSAC algoritması ile küre yüzey çıkarmada kullanılan C++ kodları sunulmuştur.

RANSAC algoritması ile küre yüzey çıkarma işlemi, noktaların yüzey normaline bağlı olarak gerçekleştirilmiştir. EK-7'de sunulan kodlarda ilk önce girdi veri seti için yarıçap arama yöntemine (KSearch) göre normali hesaplanacak

noktanın komşuları bulunmakta daha sonra bu komşu noktalara bağlı olarak noktaların yüzey normalleri hesaplanmaktadır.

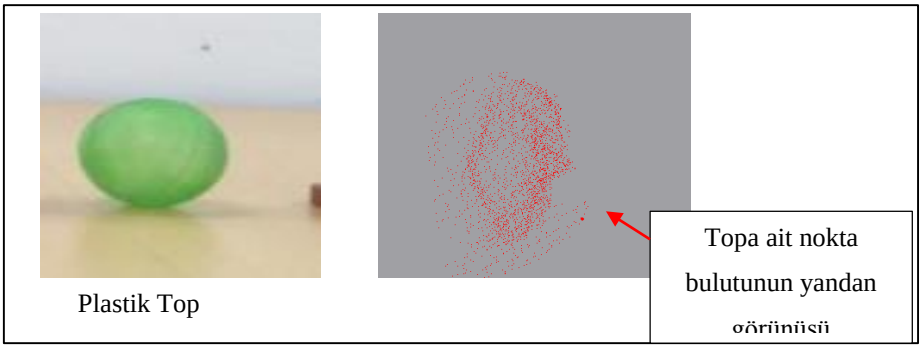
Yüzey normalleri hesaplanan noktalar küre model için gerekli parametre tanımları yapıldıktan sonra girdi veri içinden RANSAC algoritması ile çıkarılmaktadır. Çalışma kapsamında küre yüzey çıkarma algoritması ilk önce 337094 noktaya sahip olan ilk veri setine uygulanmıştır. Ancak koni yüzey çıkarma işleminde karşılaşılan problemle küre yüzey çıkarma uygulamasında da karşılaşılmıştır. Küre yüzeyleri temsil eden nesneler üzerine düşen noktaların sayısı çok azdır (yaklaşık 2000). İlk veri setinden küre yüzey çıkarmak için gerekli olan iterasyon sayısı, PCL kütüphanesinde tanımlı olan maksimum iterasyon değerinden çok fazladır. Bundan dolayı küre çıkarma algoritması 22526 adet noktaya sahip veri setine uygulanmıştır.

İkinci veri setinden küre yüzeylerin çıkarımı için gerekli minimum ve maksimum değerleri 0.10 m ve 0.15 m olarak tanımlanmıştır. p = 0.99 ve ϵ = 0.85 kabul edilerek maksimum iterasyon değeri Eşitlik 3.8'den yaklaşık 10000 olarak tanımlanmıştır. t eşik mesafesi ve w normal ağırlık değeri ise farklı değerler verilerek deneme yoluyla bulunmuştur. Şekil 5.21'de ikinci test verisinden küre yüzey çıkarmak için tanımlanan parametreler gösterilmiştir.

```
// küre yüzey için segmentasyon parametrelerini gir
seg.setOptimizeCoefficients (true);
seg.setModelType (pcl::SACMODEL_NORMAL_SPHERE);
seg.setMethodType (pcl::SAC_RANSAC);
seg.setNormalDistanceWeight (0.008);
seg.setMaxIterations (10000);
seg.setDistanceThreshold (0.01);
seg.setRadiusLimits (0.10,0.15);
```

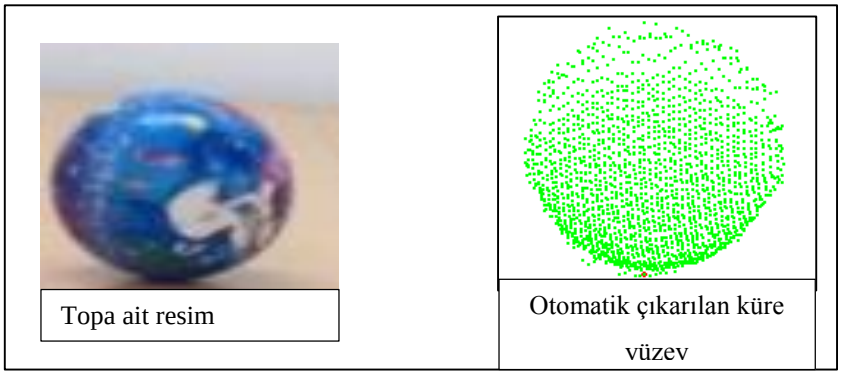
Şekil 5.21. Küre yüzey çıkarımı için tanımlanan model parametreleri

Tanımlanan parametrelere göre algoritma veri setine uygulanmıştır. Veri seti içinde sağ tarafta yer alan yeşil renkli plastik top saydamsı bir yapıya sahip olduğu için bu top üzerine gelen lazer ışınlarının büyük bölümü topun içinden yansıdığından top küre özelliğini kayıp etmiştir (Şekil 5.22). Bundan dolayı test verisi içinden çıkarılamamıştır.



Şekil 5.22. Saydamsı plastik topun yüzeyinden geri yansıyan noktalar

Sol tarafta yer alan mavi rekli top ise uygulanan RANSAC algoritması ile çıkarılmıştır. Çıkarılan topa ait küre yüzey toplamda 1952 adet nokta verisi içermektedir. Şekil 5.23'de top küre yüzeyi için veri seti içinden çıkarılan noktalar renklendirilerek gösterilmiştir. Veri seti içinden top küre yüzeyi zamansal olarak incelendiğinde 12 saniye gibi çok kısa bir sürede çıkarılmıştır.



Şekil 5.23. RANSAC algoritması ile veri seti içinden çıkarılan küre yüzey

5.1.5. Geometrik Yüzeylerin Ardışık Olarak Çıkarılması

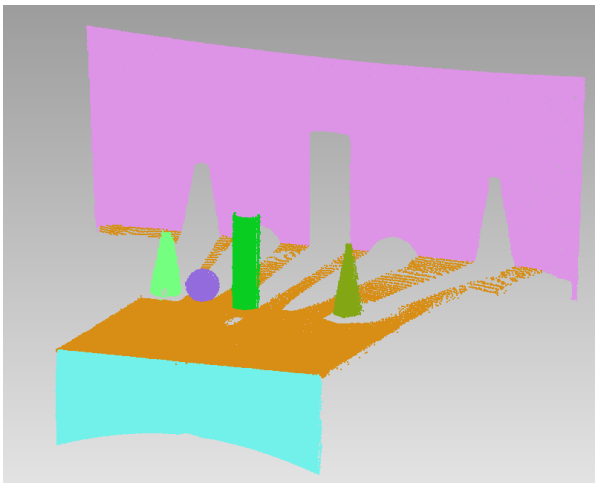
Tez çalışması kapsamında PCL Kütüphanesi kullanılarak içinde düzlem, silindir, küre ve koni gibi yüzeyleri içeren bir nokta bulutundan tüm yüzeyleri ardışık olarak tek bir programla çıkarabilen bir kod yazılmıştır. C++ ortamında yazılan kod EK-8'de sunulmuştur. EK-8'de sunulan kodlarda ilk önce nokta bulutu diskten okunmakta ve daha sonra nokta bulutu içindeki noktaların yüzey normalleri hesaplanmaktadır. Noktaların yüzey normallerinin hesaplanmasında kullanılacak uygun komşu nokta sayısı yapılan denemeler sonucunda 50 olarak belirlenmiştir. Bundan dolayı KSearch (50) olarak tanımlanmıştır.

Yüzey normalleri hesaplanan nokta bulutu verisine sırasıyla, RANSAC düzlem, silindir, küre ve koni çıkarma algoritmaları uygulanmıştır. Her bir geometrik yüzeyin çıkarılmasında kullanılacak modellerin hesaplanması için gerekli olan parametreler deneysel olarak tekrarlı işlemlerle belirlenmiştir. Belirlenen parametreler EK-8'de sunulan kodlarda yer aldığı şekildedir. Parametre değerlerine göre nokta bulutu içindeki yüzeyler ardışık olarak çıkarılmıştır. Çıkarılan her bir yüzeye ait noktalara veri seti içinde filtrelenerek bilgisayara kayıt edilmiştir. Kalan nokta bulutu verisine sıradaki algoritma uygulanmıştır. Bu şekilde özellikle küre ve koni çıkarım işleminde sorun yaşanmamıştır. Çıkarılan her bir yüzeye ait nokta sayısı, çıkarılan yüzey türü ve sayısı Çizelge 5.4'de verilmiştir.

Çizelge 5.4. RANSAC algoritması ile veri seti içinden tek bir programla çıkarılan geometrik yüzeylere ait bilgiler

Çıkarılan Yüzey		Nokta Sayısı
Düzlem	Düzlem 1	165846
	Düzlem 2	83260
	Düzlem 3	67058
Silindir	Silindir 1	6142
Küre	Küre 1	1866
Koni	Koni 1	3173
	Koni 2	2769

337094 adet nokta içeren veri setine her bir yüzey çıkarım algoritması ayrı ayrı uygulandığında veri seti içinden küre ve koni yüzeylerin çıkarılamadığı görülmüştü. Aynı veri setine tek bir program içinde tüm yüzey çıkarım algoritmaları uygulandığında tüm yüzeylerin başarılı bir şekilde çıkarıldığı görülmüştür. Çıkarılan tüm geometrik yüzeyler Şekil 5.24'de farklı renklerle renklendirilerek gösterilmiştir.



Şekil 5.24. Tek bir program ile geometrik yüzeylerin nokta bulutundan ayrıştırılması

Test verisi içinde yer alan tüm yüzeylerin tek bir algoritma ile çıkarım işlemini zamansal olarak algoritmaların tek tek veri setine uygulanması ile karşılaştırıldığında zamansal olarak daha hızlı olduğu gözlemlenmiştir. Tüm şekilleri temsil eden yüzeyler veri seti içinden 150 saniyede çıkarılmıştır.

Veri seti içinde yer alan geometrik yüzeyler bu bölümde anlatıldığı gibi tek bir programla çıkarılabileceği her bir yüzey çıkarım algoritması ayrı ayrı uygulanarak da çıkarılabilir. Örneğin veri setine ilk önce düzlem yüzey çıkarma algoritması uygulanıp tüm düzlemleri içeren noktalar veri seti içinde çıkarılabilir. Daha sonra kalan noktalara silidir yüzey çıkarma algoritması, küre yüzey çıkarma algoritması ve koni yüzey çıkarma algoritmaları uygulanabilir.

5.2. Arazi Verilerinden Geometrik Yüzeylerin Otomatik Olarak Çıkarılması

Tez çalışmasının bu bölümünde, laboratuvar ortamında elde edilen test verilerine uygulanan RANSAC yüzey çıkarım algoritmaları araziden yersel lazer tarayıcılar ile elde edilen veri setleri üzerine uygulanmıştır. Gerçek hayata birçok yapı düzlem yüzeye sahip olduğu için bu çalışmada daha çok düzlem yüzey çıkarma işlemi gerçekleştirilmiştir.

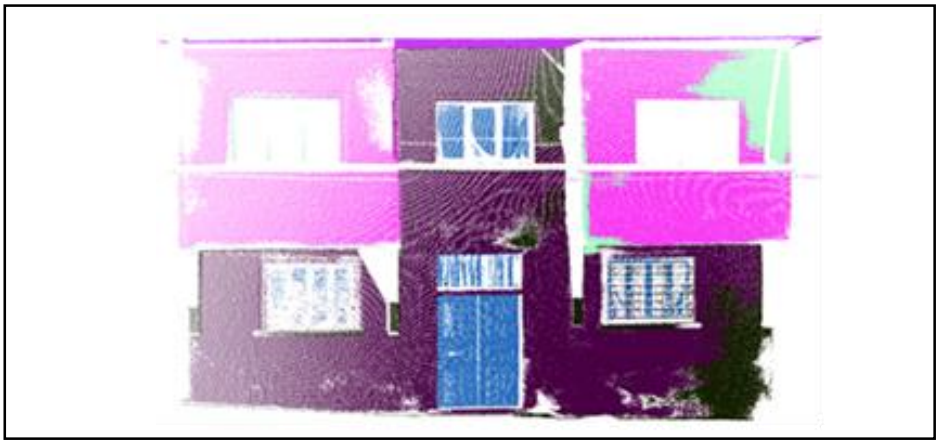
5.2.1. Bina Düzlem Cephelerinin Yüzey Normalinden Bağımsız Çıkarılması

Bina düzlem yüzeylerini RANSAC algoritmasının kullanılabilirliğini araştırmak için iki adet bina cephesi kullanılmıştır. Cephelere ait nokta bulutu verileri Eskişehir Sivrihisar ilçesinde yapılan sokak sağlıklaştırması projesi kapsamında elde edilen veriler içinden seçilmiştir. İlk veri seti iki katlı bir yapının ön cephesine ait nokta bulutu verisidir (Şekil 5.25a). Bu nokta bulutu verisi tek istasyondan elde edilen 549285 adet nokta içermektedir. İkinci veri seti ise yine iki katlı bir konuta ait cephe verisidir (Şekil 5.25b). Bu nokta bulutu verisi 582118 noktadan oluşmakta ve içinde 2 ayrı istasyondan elde edilmiş noktalar yer almaktadır.

Şekil 5.25. a: Birinci veri seti; b: ikinci veri seti

Örneklem olarak kullanılan veri setlerine EK-3'de sunulan RANSAC yüzey normalinden bağımsız düzlem yüzey çıkarma algoritması uygulanmıştır. Düzlem çıkarmada hangi noktaların düzleme atanabileceği kararında ihtiyaç duyulan t eşik değeri mesafesini bulmak için, deneysel olarak denemeler yapılmıştır. İlk örneklem nokta bulutu verisinden düzlem yüzeylerini çıkarılmasında, uygun t eşik mesafesini belirlemek için Çizelge 5.5'de gösterilen t eşik mesafesi değenlerine göre RANSAC algoritması nokta bulutu verisine uygulanmıştır. Veri seti için t değeri ilk önce 2 cm olarak verilmiştir. 2 cm'lik t değerine göre otomatik olarak 26 adet düzlem çıkarılmıştır. Çıkarılan düzlemler incelendiğinde tek bir düzlem olarak çıkarılması gereken yüzeyler birden fazla düzlem olarak çıkarıldığı gözlemlenmiştir. Şekil

5.26'da 2 cm t değerine göre çıkarılan ilk altı düzlem gösterilmiştir. Her bir renk farklı düzlemi ifade etmektedir.



Şekil 5.26. t = 2 cm değeri için veri setinden çıkarılan ilk altı düzlem

Veri seti içinde yer alan tüm düzlemlerin yüzey normalinden bağımsız olarak RANSAC algoritması ile çıkarılması toplamda 18 dakikada gerçekleştirilmiştir. Veri seti için t eşik değeri için farklı uzunluklar seçilerek algoritma veri setine uygulanmıştır. Çizelge 5.5'de seçilen t eşik mesafesi ve bu eşik mesafesine bağlı olarak veri setinden çıkarılan düzlem sayısı gösterilmiştir. t eşik mesafesi veri seti için 6 cm'ye kadar arttırılmış ve konuta ait nokta bulutu için uygun değerin 5 olduğu sonucuna ulaşılmıştır. t eşik değeri 5 cm'den küçük alındığın düzlem sayısı arttığı 5 cm'den büyük alındığında ise düzlem sayısının azaldığı görülmüştür.

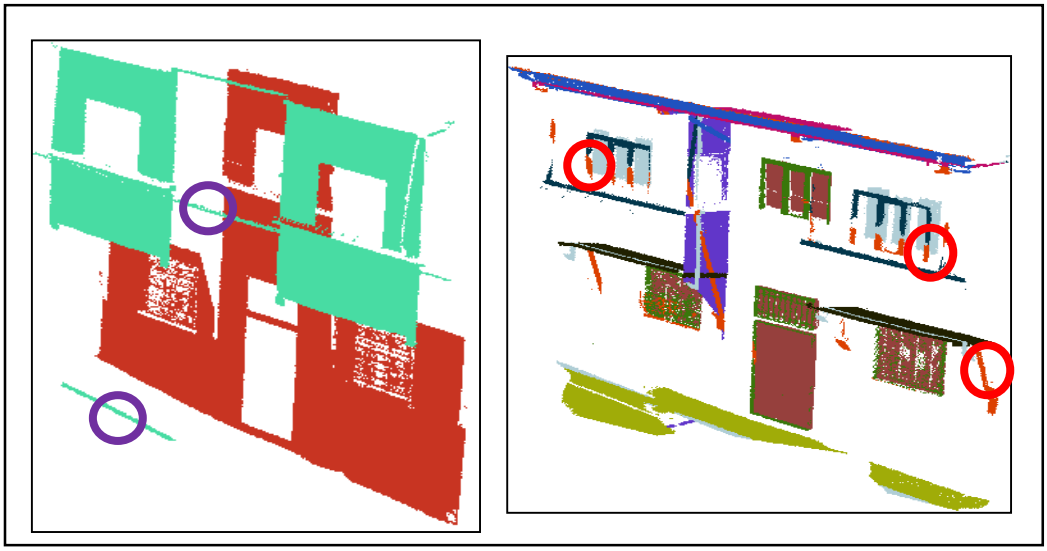
Çizelge 5.5. Farklı t eşik değeri mesafelerine göre veri seti

t eşik değeri mesafesi (cm)	Çıkarılan düzlem yüzey sayısı
2.0	26
3.0	18
3,5	16
4.0	14
4.5	13
5.0	12
5.5	12
6.0	11

t = 5 cm eşik mesafesine göre çıkarılan düzlemler Şekil 5.27'de gösterilmiştir. Her renk ayrı düzlemi temsil etmektedir.

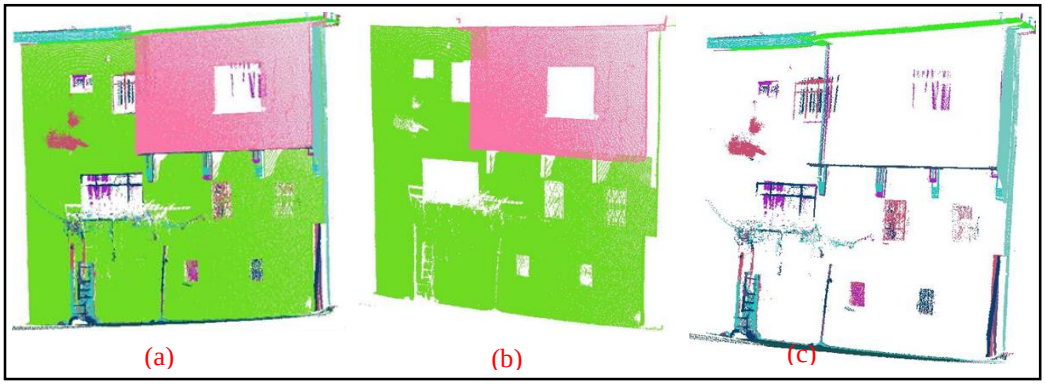
Şekil 5.27. t = 5 cm değerine göre çıkarılan düzlemler

Şekil 5.28'de ise t = 5 göre çıkarılan düzlemler ana iki cephe ve diğer 10 cephe olmak üzere iki görüntü olarak sunulmuştur. Çıkarılan düzlemler şu şekilde sıralanmaktadır. Pencere ve kapılar için 4, ön cepheler için 2, yan cephe için 1, zemin için 1, çıkma kat altı için 1, çatı düz tavanı ve eğik tavanı için 2 adet düzlem çıkarılmıştır. Son düzlem ise şekil 5.28'de daire içinde gösterilmiş ikinci katta bulunan çıkmayı destekleyen ağacın sahip olduğu eğik düzlemdir. Çıkarılan düzlemler incelediğinden yapının ikinci kat çıkması için çıkarılan düzleme zeminde yer alan noktaların (mor daire) ve iki çıkma arasında yer alan kabloya ait noktalarında (mor daire) atandığı gözlemlenmiştir. Bununla birlikte ikinci kat destek ağacı için çıkarılan düzlem yüzeye, ikinci katta yer alan pencere noktalarının (kırmızı daire) da atandığı görülmüştür.



Şekil 5.28. t = 5 değerine göre çıkarılan düzlemlerin ayrı ayrı gösterimi

İkinci veri seti için gerekli olan t değerini belirlemek için ilk örneklem verisinde olduğu gibi farklı eşik değerleri verilerek EK-3'de sunulan RANSAC yüzey normalinden bağımsız düzlem yüzey çıkarma algoritması veri setine uygulanmıştır. Uygulama sonucunda ikinci veri seti için uygun t değeri 4.5 cm olduğu gözlemlenmiştir. Şekil 5.29a'da t = 4.5 cm eşik değeri uzunluğuna göre ikinci veri setinden çıkarılan düzlemler bütünleşik ve ayrı ayrı gösterilmiştir. Otomatik olarak çıkarılan ilk iki düzlem bina cephesinin ana hatlarını göstermektedir (Şekil 5.29b). Şekil 5.29c'de ise ana hatlar dışında pencere perdeleri, zemin ve yan cephe çıkıntısı gibi alanlara ait düzlemler görülmektedir.

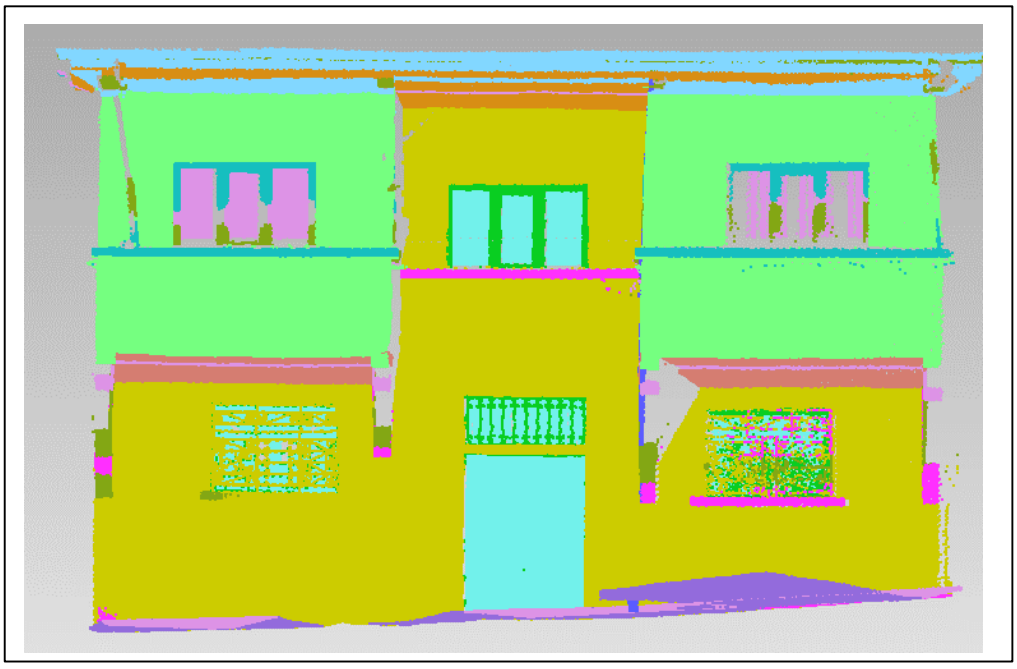


Şekil 5.29. t = 4.5 değerine göre ikinci veri setinden çıkarılan düzlem yüzeyler

5.2.2. Bina Düzlem Cephelerinin Yüzey Normaline Bağımlı Olarak Çıkarılması

Bu bölümde 549285 adet noktaya sahip birinci konuta EK-4'de sunulan RANSAC yüzey normaline bağımlı olarak düzlem yüzey çıkarma algoritması uygulanmıştır. Algoritma ile ilk önce nokta bulutu içinde yer alan noktaların yüzey normalleri hesaplanmıştır. Noktaların yüzey normalinin hesaplanması için gerekli olan komşu nokta seçim işlemi için yarıçap arama (RadiusSearch) ile belirlenmiştir. Komşu noktaların belirlenmesi için gerekli yarıçap değeri 3 cm olarak tanımlanmıştır.

Veri setinde yer alan düzlemlerin çıkarılması için gerekli olan t eşik mesafesi ve w normal ağırlık etkisi farklı değerler verilerek veri seti için uygun t ve w değerleri belirlenmiştir. Konut verisi için bu değerler t = 0.05 m ve w = 0.1 değerleridir. Belirlenen değerlere göre nokta bulutu içinde toplamda 11 adet düzlem yüzey çıkarılmıştır. Çıkarılan düzlemler Şekil 5.30'da gösterilmiştir. Düzlemler incelendiğinde, yapının çıkma cephesine ait düzlem yüzey çıkarım işleminde yüzey normalinden bağımsız yapılan işlemde zeminde bulunan noktaların da düzleme atandığı gözlemlenmişti. Düzlem çıkarma işleminde yüzey normali dikkate alındığında w = 0.1 değerine göre zemin noktalarının, kabloya ve üst çatıya ait noktaların cepheye atanmadığı görülmüştür.



Şekil 5.30. t = 0.05 ve w = 0.1 değerlerine göre veri setinden çıkarılan düzlemler

Binaya ait nokta bulutu içinden düzlem yüzeylerin çıkarılmasında, yüzey normali hesap katıldığında düzlem çıkarma işleminin zamanı artmaktadır. Bu

kapsamda veri seti içinde yer alan tüm düzlemlerin yüzey normaline bağımlı olarak çıkarılması toplamda 38 dakikada gerçekleşmiştir.

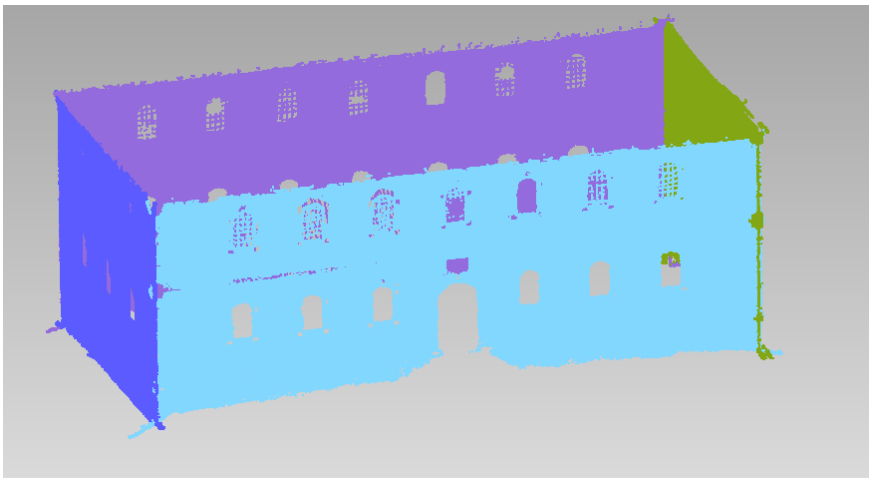
5.2.3. Tarihi Bir Yapının Düzlem Yüzeye Sahip Cephelerinin Çıkarılması

Bu bölümde RANSAC algoritması kullanılarak tarihi bir yapının düzlem yüzeyleri çıkarılmıştır. Yapı Eskişehir'in Sivrihisar ilçesinde yer alan tarihi bir askerlik şubesidir. Bina, çatısı yıkılmış taş bir binadır (Şekil 5.31). Yapının tamamına ait yersel lazer tarayıcı ile elde edilmiş nokta bulutu verisi vardır. Bu veri içinde binanın ana gövdesi RiScan Pro yazılımında elle çıkartılarak dışarı aktarılmıştır. Dışarı aktarılan veri PCD formatına dönüştürülerek, veri setine RANSAC algoritması uygulanmıştır. Binanın dış cephesine ait nokta bulutu verisi toplamda yedi farklı istasyondan elde edilmiş ve 2684238 adet nokta verisi içermektedir.

Şekil 5.31. Tarihi Askerlik Şubesine ait nokta bulutu verisi

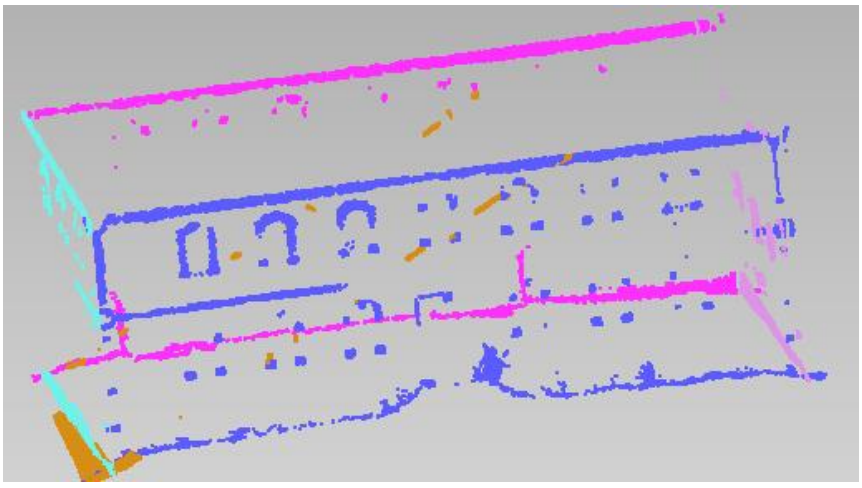
Veri seti taş bir yapı olduğundan, duvarı oluşturan taşlar arasında yüzey normali farklılıkları çok olacaktır. Bundan dolayı veri setine EK-3'de sunulan RANSAC yüzey normalinden bağımsız düzlem yüzey çıkarma algoritması uygulanmıştır. Düzlem yüzeylerin çıkarılması için gerekli olan t eşik değeri mesafesin farklı değerler verilerek deneysel olarak belirlenmiştir. Askerlik şubesine

ait nokta bulutu için uygun t eşik değeri mesafesinin 0.05 m olduğu gözlemlenmiştir. t =0.05 m değerine göre nokta bulutu içinden toplamda 9 adet düzlem yüzey otomatik olarak çıkarılmıştır. Çıkarılan bu düzlemlerden ilk dört tanesi yapının dört ana cephesine ait düzlem yüzeylerdir. Bu düzlemler farklı renklerle renklendirilerek Şekil 5.32'da gösterilmiştir.



Şekil 5.32. t = 0.05 m değerine göre askerlik şubesinin dört ana cephesine ait düzlemler

Veri seti içinde çıkarılan diğer 5 adet düzlem ise zemine, yapının pencere çıkıntıları ve çatı çıkıntılarına ait düzlem yüzeylerdir. Bu düzlem yüzeylerde farklı renklerle renklendirilip Şekil 5.33'de gösterilmiştir.

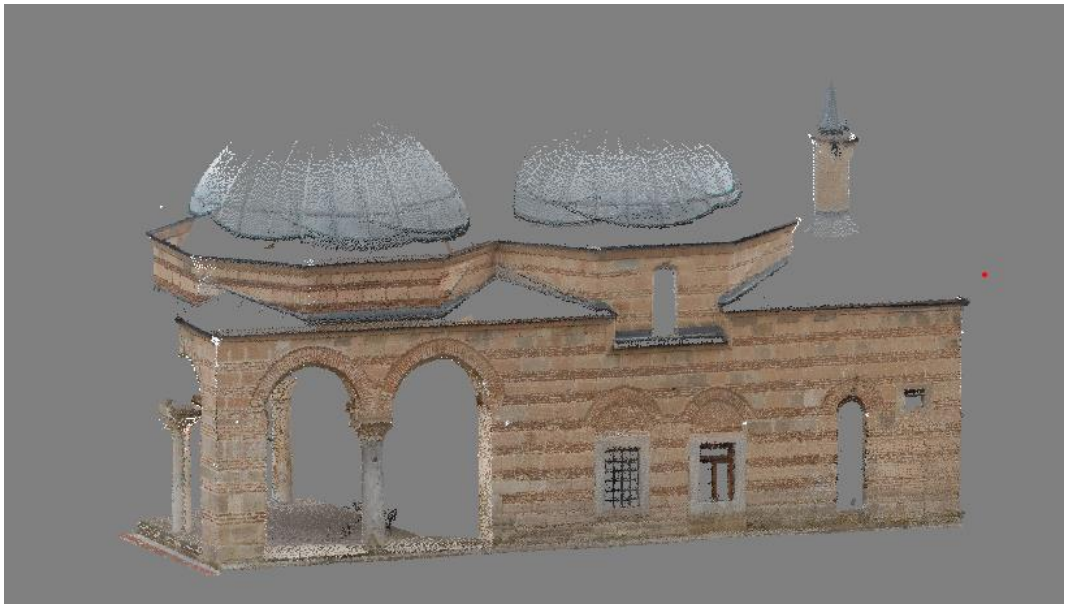


Şekil 5.33. t = 0.05 m değerine göre askerlik şubesine ait diğer düzlem yüzeyler

Tarihi yapıya ait nokta bulutu içinden mevcut tüm düzlemlerin çıkarılması işlemi zamansal olarak değerlendirildiğinde, veri seti içindeki tüm düzlemler 55 dakikada çıkarılmıştır.

5.2.4. Farklı Geometrik Yüzeylere Sahip Tarihi Bir Yapının Yüzeylerinin Çıkarılması

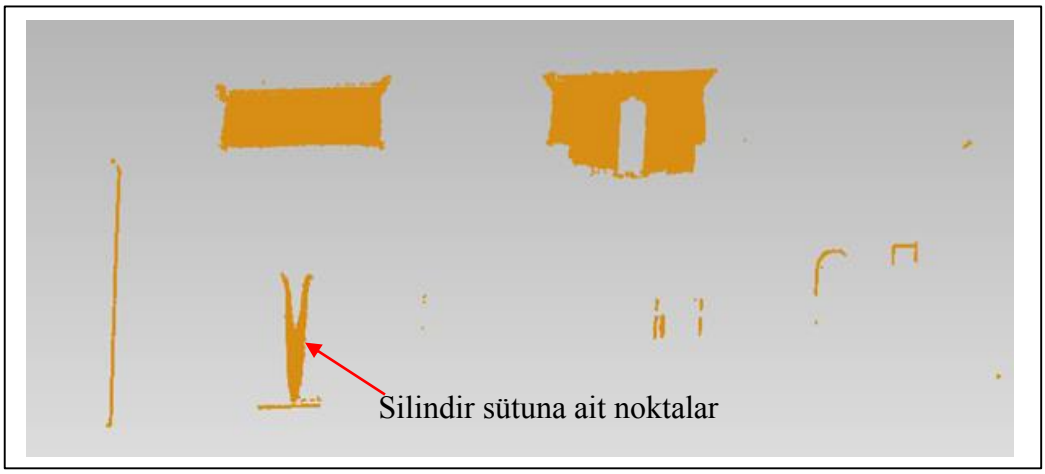
Araziden elde edilen nokta bulutu içinden farklı geometrik yüzeye sahip verilerin RANSAC algoritması ile çıkarılması için Eskişehir'de yer alan Kurşunlu Külliyesinin okuma salonuna ait lazer tarama verisi kullanılmıştır. Çalışma kapsamında kullanılacak veri seti RiScan Pro ortamında külliyeye ait birleştirilmiş nokta bulutu içinden ayrıştırılıp alınmıştır. Şekil 5.34'de veri seti olarak kullanılan Kurşunlu külliyesine ait nokta bulutu verisi yer almaktadır. Veri seti içinde 3 adet silindir sütun 2 adet küre kubbe ve duvarlara ait düzlem yüzeyler yer almaktadır. Bununla birlikte minarenin tepesinde koni yüzey olarak kabul edilebilir. Veri seti 1909390 adet nokta verisi içermektedir.



Şekil 5.34. Kurşunlu Külliyesine ait nokta bulutu verisi

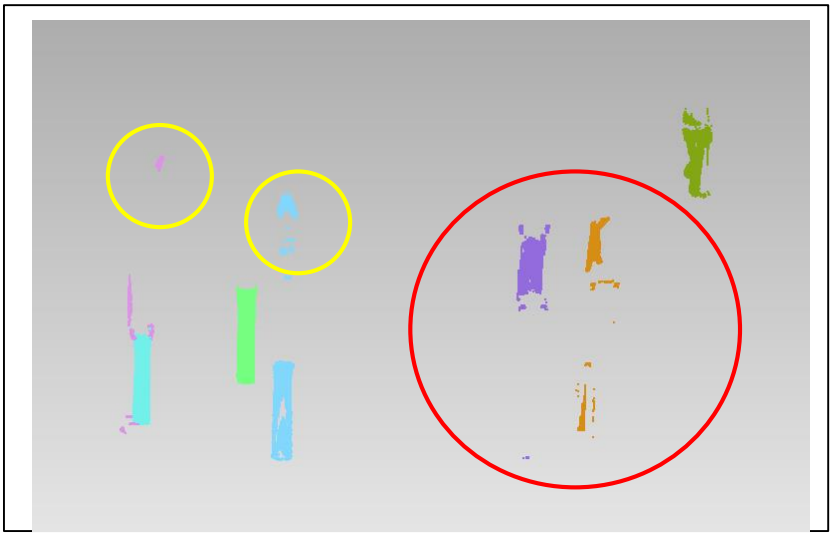
Veri setine ilk önce yüzey normalinden bağımsız olarak düzlem çıkarma algoritması uygulanmıştır. Uygulanan algoritma ile veri seti içinde yer alan en büyük düzlem yüzeyler çıkarılmıştır. Düzlem yüzeylerin çıkarılmasında t eşik mesafesi yapının sahip olduğu en büyük düzlem yüzey olan ön cephesine göre belirlenmeye çalışılmıştır. Bu yüzeye göre t eşik mesafesi 3.5 cm olarak deneysel olarak tespit edilmiştir. Belirlenen bu değere göre uygulanan algoritma ile düzlem yüzeyler çıkarılırken bir takım sorunlarla karşılaşılmıştır. Örneğin yapının ön cephesinin hemen üstünde yer alan düzlem yüzeyler çıkarılırken, yapıdaki silindir

sütunlara ait noktaların bu düzleme atandığı gözlemlenmiştir (Şekil 5.35). Ayrıca aynı cephe yüzeyin t = 3.5 cm değerine göre birden fazla düzlem ile çıkarıldığı görülmüştür. t değeri artırıldığında bu cephenin tek bir düzlem ile çıkarıldığı gözlemlenmiştir. Ancak artırılan t değeri ile birlikte ön cepheye ait olmayan noktaların ön cepheye atandığı, silindir sütunun birçok noktasının düzlem yüzey olarak çıkarıldığı görülmüştür.



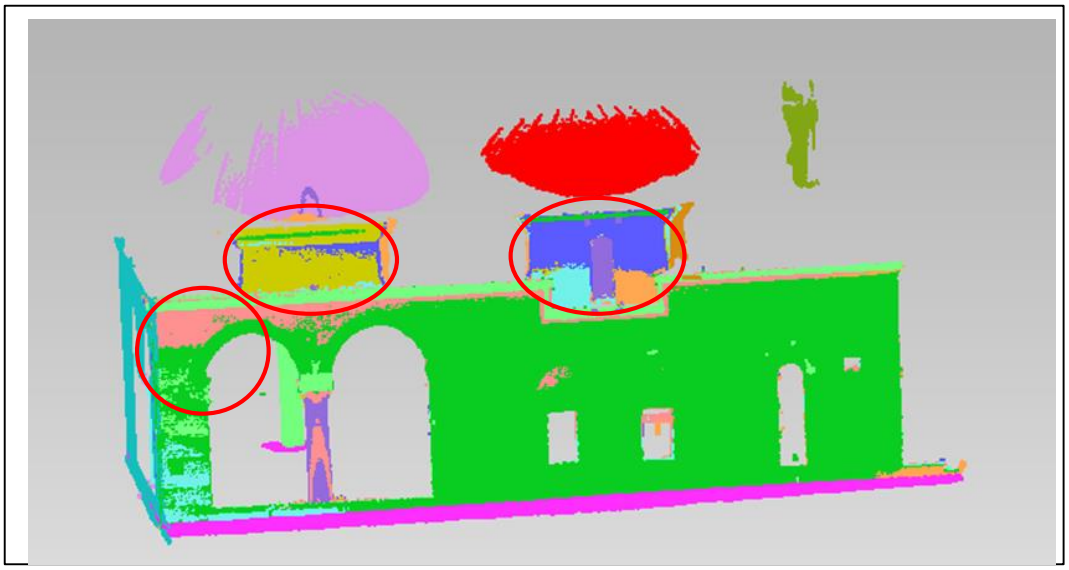
Şekil 5.35. t = 3.5 cm değerine göre düzlem yüzey atanan silindir noktaları

Veri setine düzlem yüzey çıkarma algoritması uygulandıktan sonra kalan nokta bulutuna silindir yüzey çıkarma algoritması uygulanmıştır. Silindir yüzey çıkarma algoritması için yarıçap sınırları 0.15 m ile 0.50 m arasında belirlenmiştir. Silindir yüzey çıkarma işlemi için uygun t eşik mesafesi ise 4.5 olarak deneysel olarak tespit edilmiştir. Veri setine uygulanan silindir yüzey çıkarma işlemi sonucunda elde edilen silindir yüzeyler Şekil 5.36'da gösterilmiştir. Çıkarılan silindir yüzeyler incelendiğinde yapının pencerelerine ait bazı nokta kümlerinin (kırmızı daire) silindir yüzey olarak çıkarıldığı görülmüştür. Bununla birlikte yapıda yer alan 3 adet sütun çıkarılırken yapının kubbesine ait noktalarında (sarı daireler) bu yüzeylere atandığı gözlemlenmiştir.



Şekil 5. 36. t = 4.5 cm'ye göre çıkarılan silindir yüzeyler

Veri seti içinde yer alan silinidir yüzeylerde çıkarıldıktan sonra kalan nokta bulutuna küre yüzey çıkarma algoritması uygulanmıştır. Küre yüzey çıkarma işlemi için t = 5 cm ve yarıçap sınırları 1 m ile 3 m olarak belirlenmiştir. Belirlenen bu değerlere göre küre yüzeyler çıkarılmıştır. Veri seti içinden çıkarılan tüm yüzeyle Şekil 5.35'de gösterilmiştir. Şekil 5.37'de kırmızı daire içinde yer alan cepheler birden fazla düzlem ile çıkarılmıştır.



Şekil 5. 37. Kurşunlu Külliyesine ait veri seti içinden çıkarılan tüm yüzeyler

Çıkarılan tüm yüzeyler zamansal olarak değerlendirildiğinde yüzeyler veri seti içinden 2 saatlik bir süre içinde çıkarılmıştır.

6. ARAŞTIRMA SONUÇLARI

RANSAC algoritması ile nokta bulutu verilerinden otomatik olarak çıkarılan yüzeylerin doğruluklarını araştırmak için çalışmada kullanılan veri setleri içinde yer alan düzlem, silindir, küre ve koni nesneleri veri setinden RiScan Pro yazılımında elle de çıkarılmıştır. Otomatik çıkarılan yüzeylerin doğruluklarını araştırmak için elle çıkarılan yüzeyler doğru kabul edilmiştir. Yüzeylerin karşılaştırma işleminde ortak olarak çıkarılan noktaların belirlenmesi için MATLAB ortamında kısa bir kod yazılmıştır. İlgili kod EK-9'da sunulmuştur. Ayrıca yüzeyler çıkarılan yüzeyler Geomagic Studio 2012 ortamında çakıştırılarak görsel olarak farklılıklar izlenmeye çalışılmıştır.

6.1. Elde Edilen Düzlem Yüzeylerin Değerlendirilmesi

Test verisi içinde yer alan masanın düzlem ön yüzeyine ait nokta bulutu verisi RiScan Pro yazılımında elle de çıkarılmıştır. RANSAC algoritması ile hem yüzey normaline bağımlı hem de yüzey normalinden bağımsız olarak çıkarılan bu yüzey en fazla noktaya sahip olan düzlem yüzeydir. Düzlem yüzeylerin değerlendirilmesine yönelik olan bu bölümde ilk önce elle çıkarılan düzlem yüzey, RANSAC algoritması ile yüzey normalinden bağımsız olarak çıkarılan düzlem yüzey ile karşılaştırılacaktır. İkinci olarak elle çıkarılan düzlem yüzey, yüzey normaline bağımlı olarak çıkarılan düzlemle karşılaştırılacaktır. Ardından ise algoritma ile çıkarılan bu iki düzlem bir biri ile karşılaştırılmıştır.

6.1.1. Yüzey Normalinden Bağımsız Çıkarılan Düzlemlerin Değerlendirilmesi

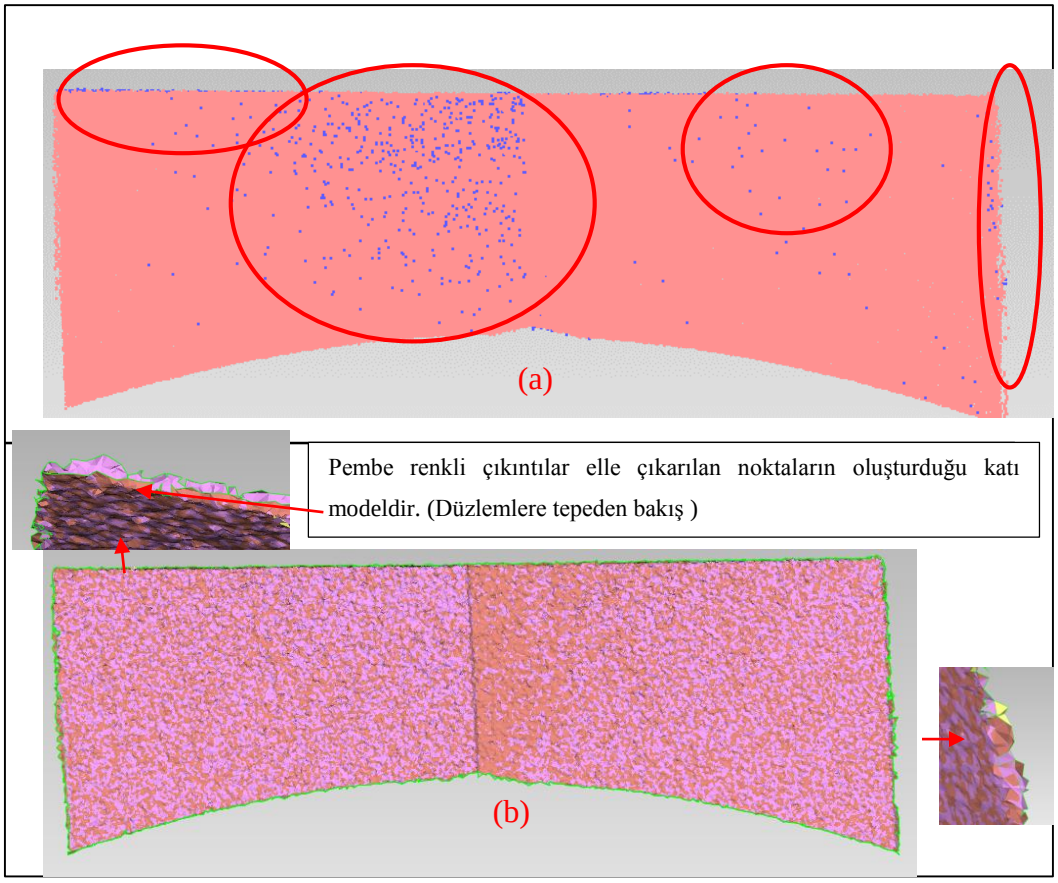
RANSAC algoritması test verisine yüzey normalinden bağımsız olarak uygulandığında masa ön cephesine ait toplam 165294 adet nokta çıkarılmıştır. Aynı yüzeyin elle çıkarılan nokta adedi ise 167109 dur. Çıkarılan iki düzlem karşılaştırıldığında her iki düzlemde ortak olarak çıkarılan 164944 adet nokta olduğu görülmektedir. Ortak olarak çıkarılan noktalar elle çıkarılan noktaların % 98.70'ine karşılık gelmektedir. Otomatik olarak çıkarılan düzlemde sadece 350

nokta elle çıkarılan düzlemden farklıyken, Elle çıkarılan düzlemde 2165 adet nokta otomatik olarak çıkarılan düzlemden farklı olarak çıkarılmıştır. Çizelge 6.1'de düzlemlere ait nokta bilgisi sunulmuştur.

Çizelge 6. 1. RANSAC algoritması ile yüzey normalinden bağımsız çıkarılan düzlemin elle çıkarılan düzlemle karşılaştırılması

Düzlem Adı	Nokta Sayısı
Otomatik (O)	165294
Elle (E)	167109
Eşit Nokta Sayısı (O ∩ E)	164944
O\E	350
E\O	2165

Her iki düzlem görsel olarak görüntülendiğinde iki düzlemin hemen hemen örtüştüğü gözlemlenmiştir. Her iki düzleme ait nokta bulutları Geomagic- Studio 2012 yazılımında açılarak üst üste çakıştırılmıştır (Şekil 6.1). Şekil 6.1a'da çakıştırılan nokta bulutları gösterilmiştir. Nokta bulutunda mavi renkli gösterilen noktalar elle çıkarılan düzleme ait noktalardır. Şekil 6.1a'da kırmızı daire içinde olan yerlerde elle çıkarılan noktaların, otomatik çıkarılan noktalara göre fazlalık olduğu alanladır. Şekil 6.1b'de ise çıkarılan nokta bulutları görsel olarak değerlendirilmesi için yüzey ağı modelleme ile katı model haline getirilmiştir. Pembe renkli model elle çıkarılan nokta bulutunu, diğer renk ise otomatik çıkarılan yüzeyi temsil etmektedir. Çakıştırılan katı modellere tepeden bakıldığında elle çıkarılan noktaların oluşturduğu çıkıntılar görülmektedir.



Şekil 6.1. Elle ve otomatik olarak çıkarılan noktaların görsel olarak karşılaştırılması

Masa ön cephesinin ön yüzeyinde, elle çıkarılan noktalarda fazlalıkların olması ve bu noktaların otomatik çıkarılan yüzeye atanmamasının en büyük sebebi bu bölge lazer ışınının saçınımının fazla olmasıdır.

Elle ve otomatik olarak çıkarılan düzlemler yorumlanacak olursa, otomatik olarak çıkarılan düzlemlerde, verilen t eşik değerine göre sınır bölgelerinde, farklı düzleme atanması gereken noktalar, ait olmadığı düzleme atanabilmektedir. Örneğin bu uygulamada t = 1 cm mesafesinde masa üstüne ait noktalar birinci düzleme atandığı gözlemlenmiştir. Ancak yine de çıkarılan noktaların tutarlılığı RANSAC algoritmasının düzlem çıkarımı işlemi uygun bir yöntem olduğunu göstermiştir. Elle çıkarılan düzlemlerde ise özellikle iki düzlem arasındaki geçiş bölgelerinde çıkarılmak istenilen düzleme ait olmayan noktalar, düzleme ait olarak çıkarılabilmektedir. Elle yüzey çıkarım işleminin başarısı, yüzeyi çıkaran operatörün başarısına bağlı olduğundan operatörden kaynaklı hatalar ortaya çıkabilmektedir.

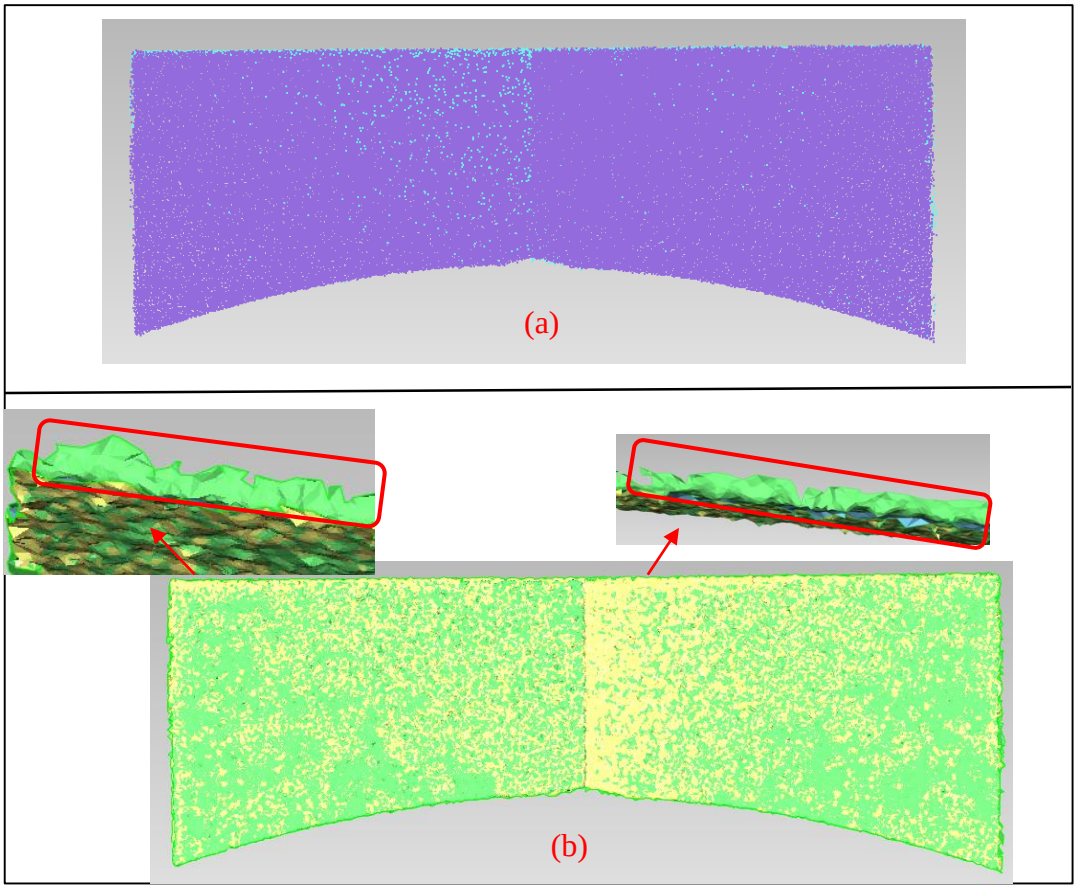
6.1.2. Yüzey Normaline Bağımlı Çıkarılan Düzlemlerin Değerlendirilmesi

Test verisi içinden RANSAC algoritması ile yüzey normaline bağlı olarak çıkarılan masa ön yüzeyine ait düzlem 163032 adet nokta verisi içermektedir. Bu düzlem ile elle çıkarılan düzlem verisi karşılaştırıldığında her iki düzlemde ortak olarak çıkarılan 162924 adet nokta yer almaktadır. Ortak olarak çıkarılan noktalar elle çıkarılan noktaların % 97.50'sine karşılık gelmektedir. Otomatik olarak çıkarılan düzleme ait noktalardan 108 tanesi elle çıkarılan düzlemden farklıdır. Elle çıkarılan düzleme ait noktalardan ise 4185 adet nokta otomatik çıkarılan düzlemden farklıdır (Çizelge 6.2).

Çizelge 6. 2. RANSAC Algoritması İle Yüzey Normaline Bağımlı Çıkarılan Düzlem ve Elle çıkarılan düzleme ait nokta bilgilerinin karşılaştırılması

Düzlem Adı	Nokta Sayısı
Otomatik (O)	163032
Elle (E)	167109
Eşit Nokta Sayısı (O ∩ E)	162924
O\E	108
E\O	4185

Elde edilen düzlemler görsel olarak karşılaştırıldığında elle çıkarılan düzleme ait noktaların özellikle düzlemin üst bölümünde ve düzlem yüzeyinde otomatik olarak çıkarılan düzlemden farklılıklar gösterdiği görülmüştür. Şekil 6.2a'da üste yer alan görüntüde iki nokta bulutunun çakıştırıldığı görülmektedir. Şekilde mavi renkli noktalar elle çıkarılan noktaların, otomatik olarak çıkarılan düzleme ait noktalardan farklı olduğu yerleri göstermektedir. Şekil 2b'de yer alan şekilde ise otomatik ve elle çıkarılan nokta bulutu verilerinden yüzey ağı modelleme ile oluşturulan katı modellerin çakıştırılmış hali görülmektedir. Katı yüzeyler incelendiğinde özellik düzlemin üst bölümünde elle çıkarılan nokta bulutu içinde masanın üst yüzeyinde yer alması gereken noktalar ön yüzey noktası olarak çıkarıldığı için burada üst yüzeye doğru düzlem dik şekilde kırılmaktadır. Şekil 6.2b'de yeşil renkli olarak kırmızı dikdörgen içinde gösterilen çıkıntılar bu kırılma alanlarıdır.



Şekil 6.2. Elle çıkarılan düzlem ve RANSAC algoritması ile yüzey normaline bağımlı otomatik olarak çıkarılan düzlemin karşılaştırılması.

RANSAC algoritması ile yüzey normaline göre otomatik olarak çıkarılan düzlemler verilen t eşik değerinin yanında nokta normali ve düzlem normali arasındaki açısal fark da dikkate alındığı için sınır bölgelerinde noktaların düzleme atanması daha başarılı bir şekilde olmuştur. Örneğin bu uygulamada t = 0.01 m ve normal etki değeri w = 0.01 değerine göre masa üstüne ait noktalar birinci düzleme atanmadığı gözlemlenmiştir.

6.1.3. Yüzey Normaline Bağımlı ve Yüzey Normalinden Bağımsız Çıkarılan Düzlemlerin Değerlendirilmesi

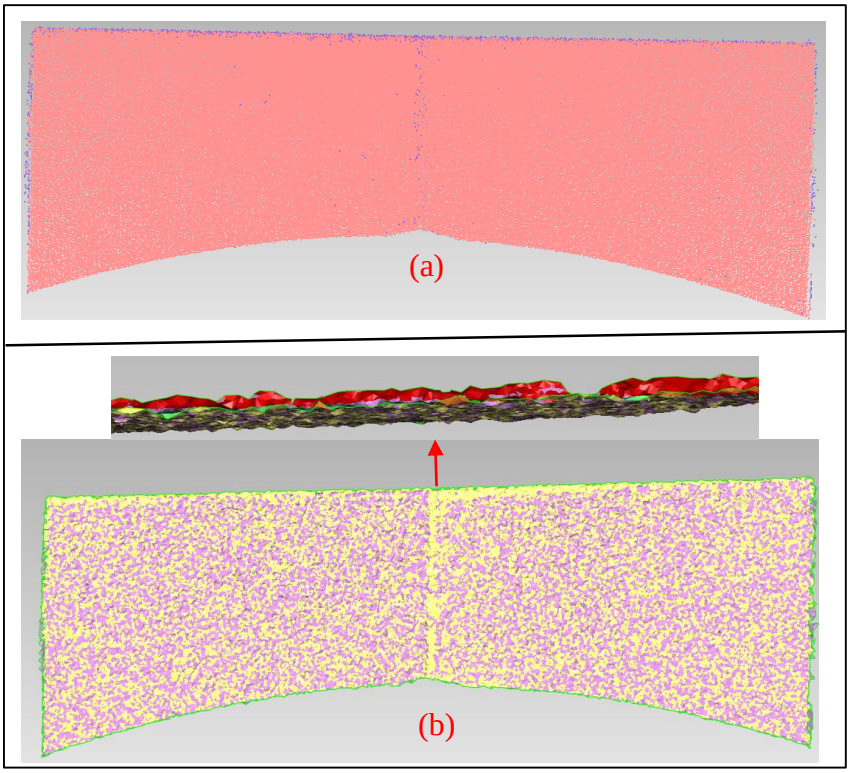
Bu bölümde RANSAC algoritması ile yüzey normaline bağlı ve yüzey normalinden bağımsız olarak çıkarılan düzlemler arasında farkı görmek için test verisi içinden çıkarılan masanın ön yüzeyine ait düzlemeler karşılaştırılmıştır.

Yapılan karşılaştırmaya ait nokta sayısı bilgisi Çizelge 6.3'de verilmiştir. Çizelge incelendiğinde her iki düzlemde ortak olarak 163002 ortak nokta mevcuttur. Yüzey normaline bağlı olarak çıkarılan düzlemden 30 nokta yüzey normalinden bağımsız olarak çıkarılan düzlemden farklıdır. Yüzey normalinden bağımsız çıkarılan düzlemde ise 2292 adet nokta diğer düzlemden farklıdır.

Çizelge 6.3. Yüzey normaline bağımlı ve yüzey normalinden bağımsız olarak çıkarılan düzlem noktalarının karşılaştırılması

Düzlem Adı	Nokta Sayısı
Otomatik (Yüzey normaline bağlı (O_1))	163032
Otomatik (Yüzey normalinden bağımsız (O_2))	165294
Eşit Nokta Sayısı ($O_1 \cap O_2$)	163002
$O_1 \setminus O_2$	30
$O_2 \setminus O_1$	2292

Çıkarılan düzlemler görsel olarak karşılaştırıldığında yüzey normalinden bağımsız çıkarılan düzlemdeki fazlalık noktaların düzlem kenarlarında ve düzlemin ortasında, masa ayrımındaki küçük sekmede olduğu görülmektedir. Şekil 6.3a' da yer alan görüntüde mor renkli noktalar yüzey normalinden bağımsız çıkarılan düzleme ait noktaların, yüzey normaline bağımlı olarak çıkarılan düzlemden farklı olduğu noktalardır. Nokta bulutları yüzey ağı modelleme ile katı model haline getirildiğinde yüzey normalinden bağımsız çıkarılan düzlemde, masanın üst yüzeyine ait noktalar mevcuttur. Bundan dolayı sınır bölgelerde bu noktalardan kaynaklı kırılmalar mevcuttur. Masa üst yüzeyine doğru meydana gelen bu kırılmalar Şekil 6.3b'de kırmızı renkli olarak gösterilmiştir.



Şekil 6. 3. RANSAC algoritması ile yüzey normaline bağımlı ve yüzey normalinden bağımsız olarak çıkarılan düzlemlerin karşılaştırılması

6.1.4. Arazi Verilerinden Çıkarılan Düzlemlerin Değerlendirilmesi

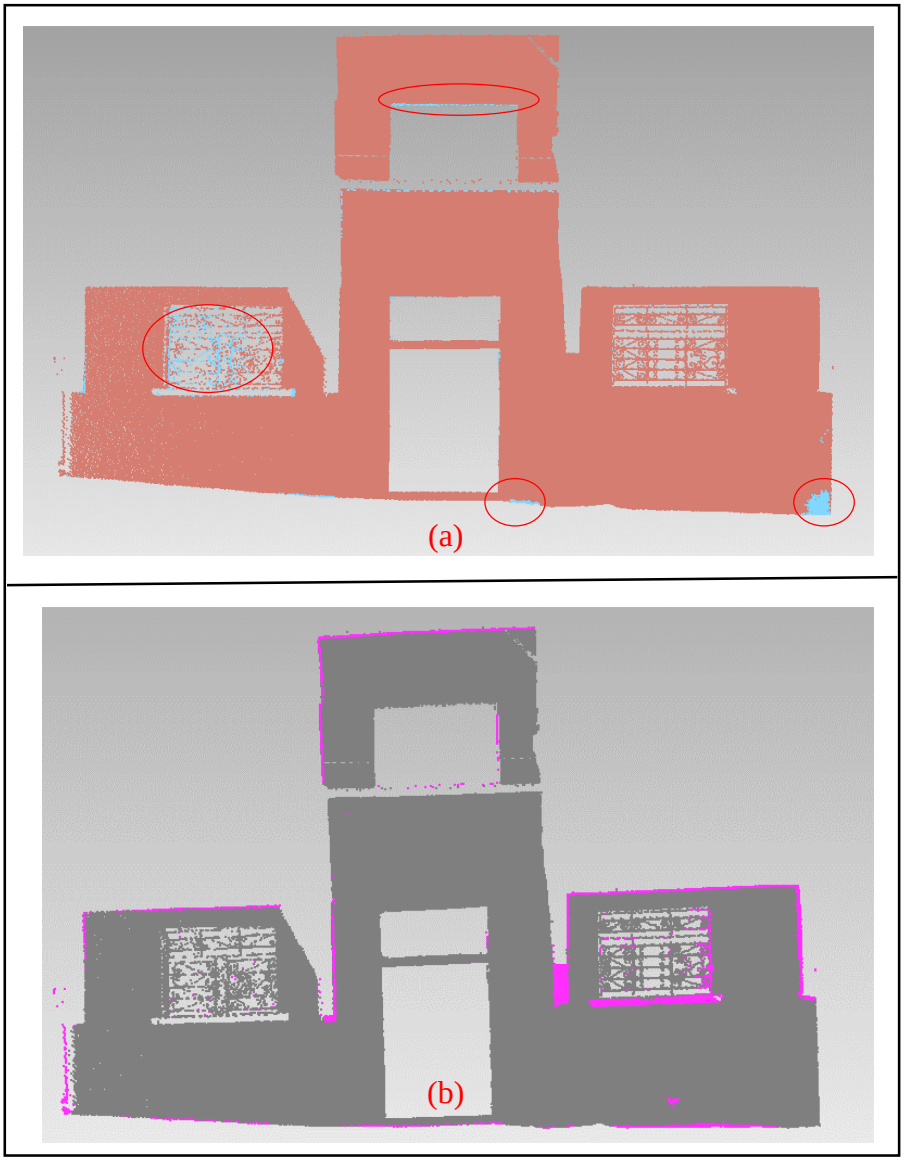
RANSAC algoritmasının arazi verilerine uygulanması ile elde edilen düzlemlerin doğruluklarının araştırılması için 5.2.1 bölümünde kullanılan verilere ait bir adet düzlem, veri seti içinden elle çıkartılmıştır.

Elle çıkartılan düzlem 5.2.1 bölümünde kullanılan ilk veri setine ait düzlemdir. Bu düzlem veri setinden RANSAC algoritması ile çıkarılan en büyük düzlemidir. Çizelge 6.4'de bu düzlem için otomatik (O) ve elle (E) çıkartılan nokta sayıları verilmiştir. Bu düzlem için otomatik olarak 213672 adet nokta, elle ise 210872 adet nokta çıkarılmıştır. Otomatik ve elle çıkarılan noktalar içinde 208377 nokta ortak olarak çıkarılmıştır. Ortak olarak çıkarılan noktalar elle çıkarılan noktaların %98.79'unu denk gelmektedir. Otomatik çıkartılan düzlemde yer alan 2552 adet nokta elle çıkarılan düzlemden farklıdır. Elle çıkartılan düzlemde ise 5352 adet nokta otomatik olarak çıkarılan düzlemden farklıdır.

Çizelge 6.4. Araziden elde edilen düzlemlerin doğruluklarının değerlendirilmesi

Düzlem Adı	**Nokta Sayısı**
Otomatik (O)	213672
Elle (E)	210872
Eşit Nokta Sayısı (O ∩ E)	208320
O\E	2552
E\O	5352

Şekil 6.4'de da otomatik ve elle çıkarılan düzlemlere ait noktalar üst üste çakıştırılmıştır. Çakıştırılan nokta bulutları incelendiğinde Şekil 6.4a'da yer alan görüntüde mavi renkle gösterilen noktalar elle çıkarılıp otomatik olarak çıkarılamayan yüzeyleri göstermektedir. Şekil 6.4b'de yer alan görüntüde ise pembe renkli noktalar elle çıkarılamayıp otomatik olarak çıkarılan noktaları göstermektedir. Genel olarak yüzeyler elle ve otomatik olarak çıkarılan yüzeylerin örtüştüğü gözlemlenmiştir.



Şekil 6.4. Arazi verilerinden otomatik ve elle çıkarılan düzlemlerin karşılaştırılması

6.2. Çıkarılan Silindir Yüzeylerin Değerlendirilmesi

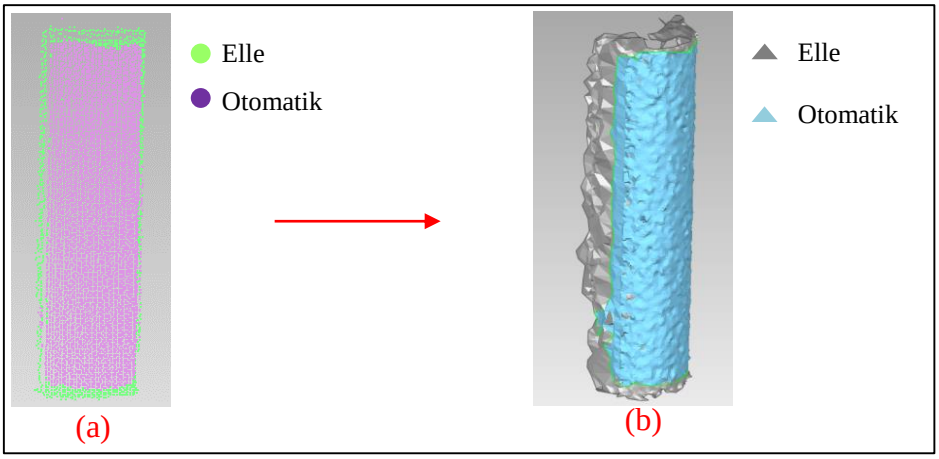
Çalışma kapsamında test verisi içinden RANSAC algoritması ile otomatik olarak çıkarılan silindir yüzey, veri seti içinden elle de çıkartılmıştır. Elle ve otomatik çıkarılan silindir yüzeye ait noktaların karşılaştırılması Çizelge 6.5'de gösterilmiştir. Test verisi içinden silindir yüzey için otomatik olarak 6579 adet, elle 5690 adet nokta çıkarılmıştır. Otomatik ve elle çıkarılan düzlemlerde toplamda 5689 adet nokta ortak olarak çıkarılmıştır. Ortak olarak çıkarılan noktaların elle çıkarılan noktalara oranı % 86.47 olarak hesaplanmıştır. Otomatik çıkarılan silindir

yüzeyden sadece 1 adet nokta elle çıkartılan noktadan farklı olarak çıkarılmıştır. Elle çıkarılan silindir yüzeyde ise 890 adet nokta otomatik olarak çıkarılan silindir yüzeye ait nokta bulutundan farklıdır.

Çizelge 6.5. Otomatik ve elle çıkarılan silindir yüzey noktalarının karşılaştırılması

Düzlem Adı	**Nokta Sayısı**
Otomatik (O)	5690
Elle (E)	6579
Eşit Nokta Sayısı (O ∩ E)	5689
O\E	1
E\O	890

Otomatik ve elle çıkarılan silindir yüzeyler görsel olarak karşılaştırıldığında elle çıkarılan silindir yüzeyde saçınıma uğrayan noktalar silindir yüzeye ait olarak çıkartılmıştır. Şekil 6.5a'da yeşil renkli görülen noktalar elle çıkarılan yüzeye ait noktaları temsil etmekte olup otomatik olarak çıkarılan yüzeyden farklı olan noktalardır. Mor renkli noktalar ise otomatik olarak çıkarılan yüzeyi göstermektedir. Her iki yüzey Geomagic yazılımında yüzey ağı modelleme ile katı modele dönüştürüldüğünde elle çıkarılan silindirin kenarlarında bozulmalar olduğu görülmüştür. Şekil 6.5b'de sağ tarafta çıkarılan yüzeylerin 3 boyutlu katı model halleri görülmektedir. Katı modellerden gri renkle gösterilen elle çıkarılan noktalardan oluşturulan katı modeli, mavi renkli katı model ise otomatik olarak çıkarılan noktalardan oluşan katı modeli göstermektedir. Katı modeller incelediğinde elle çıkarılan silindir yüzeyin kenarlarında oluşan üçgenlerde bozulmalarında olduğu gözlemlenmektedir. Otomatik olarak çıkarılan silindir noktalarında ise, sınırlarda böyle bir problem olmadığı görülmektedir. Elle çıkarılan silindir yüzeyin kenar noktlarında meydana gelen bu bozulmaların en büyük nedeni, silindiri temsil eden noktalar, nokta bulutu içinden çıkarılırken, operatör tarafından kenarda kalan noktaların silindire düşüp düşmediğinin anlaşılmasının zor olmasıdır.



Şekil 6.5. Otomatik ve elle çıkarılan silindir yüzeylerin görsel olarak karşılaştırılması

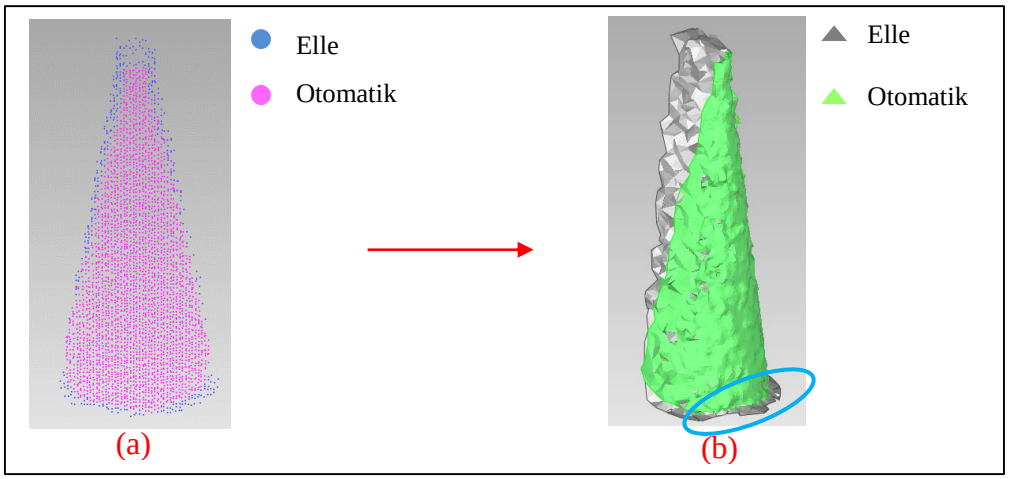
6.3. Çıkarılan Koni Yüzeylerin Değerlendirilmesi

Test verisi içinde iki adet koni şekli yer almaktadır. Test verisi içinde sağ tarafta yer alan ve RANSAC algoritması ile otomatik olarak çıkarılan koni yüzey bu bölümde veri seti içinden elle çıkarılan yüzey ile karşılaştırılmıştır. Otomatik ve elle çıkarılan koni yüzeylerin sahip olduğu noktaların karşılaştırılması Çizelge 6.6 gösterilmiştir. Çizelge incelendiğinde otomatik olarak çıkarılan koni yüzeyin 3118 elle çıkarılan koni yüzeyin ise 3512 adet noktaya sahip olduğu görülmektedir. Bu iki veri seti karşılaştırıldığında her iki veri setinden 3118 adet ortak nokta olduğu görülmektedir. Bunun anlamı otomatik olarak çıkarılan tüm noktalar elle çıkarılan yüzeyi temsil eden nokta bulutu içinde yer almaktadır. Elle çıkarılan koni yüzeyine ait nokta bulutu otomatik olarak çıkarılan yüzeye ait nokta bulutundan 394 adet farklı nokta içermektedir. Ortak olarak çıkarılan noktaların elle çıkarılan noktalara oranı ise % 88.78 olarak hesaplanmıştır.

Çizelge 6.6. Otomatik ve elle çıkarılan koni yüzey noktalarının karşılaştırılması

Düzlem Adı	Nokta Sayısı
Otomatik (O)	3118
Elle (E)	3512
Eşit Nokta Sayısı (O ∩ E)	3118
O\E	0
E\O	394

Koni yüzeyler görsel olarak karşılaştırıldıklarında, silindir yüzeyde olduğu gibi koni kenarlarında saçınıma uğrayan noktalar ve koninin alt düzlem tablasına ait noktalar elle çıkarım işleminde, koni yüzeyi olarak çıkarılmıştır. Şekil 6.6a'da otomatik ve elle çıkarılan koni yüzeylerine ait nokta bulutu verisi yer almaktadır. Altta yer alan mavi renkli noktalar elle çıkarılan noktaları üste yer alan mor renkli noktalar otomatik olarak çıkarılan koni yüzeyini temsil etmektedir. Şekil 6.6b'de ise nokta bulutlarından üretilen 3 boyutlu katı modeller görülmektedir. Katı modeller incelendiğinde altta gri renkli görülen, elle çıkarılan koni yüzeyin kenarlarında oluşan üçgenlerin girintili çıkıntılı olduğu görülmektedir. Bununla birlikte koni alt bölümünde (mavi daire) katı model düzleşmektedir. Otomatik olarak çıkarılan ve yeşil renkli katı model olarak görülen koni yüzeyi ise daha pürüzsüz bir görüntü sunmaktadır. Gri renkli koni yüzeyin kenarlarında meydana gelen bozulmaların nedeni, elle nokta çıkarımında, kenarlarda saçınıma uğrayan noktaların koniye ait olup olmadığının belirlenmesinin zor olmasıdır.



Şekil 6.6. Otomatik ve elle çıkarılan koni yüzeylerin görsel olarak karşılaştırılması

6.4. Çıkarılan Küre Yüzeylerin Değerlendirilmesi

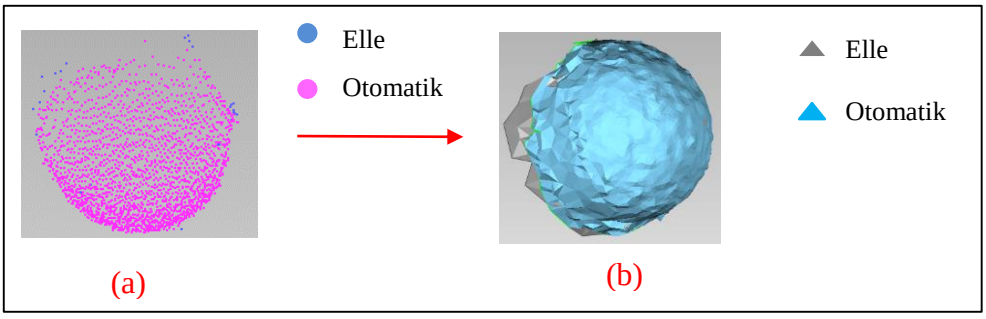
Test verisi içinde yer alan ve küreyi temsil etmek için kullanılan plastik top verisine ait nokta bulutu verisi test verisi içinden elle çıkarılmıştır. Elle çıkarılan bu küre yüzey verisi RANSAC algoritması ile otomatik olarak elde edilen küre yüzey nokta bulutu verisi ile karşılaştırılarak değerlendirilmiştir. Otomatik ve elle çıkarılan küre yüzeylerin sahip olduğu noktaların karşılaştırılması Çizelge 6.7

gösterilmiştir. Çizelge incelendiğinde otomatik olarak çıkarılan küre yüzeyin 1952 elle çıkarılan küre yüzeyin ise 1967 adet noktaya sahip olduğu görülmektedir. Bu iki veri seti karşılaştırıldığında her iki veri setinden 1940 adet ortak nokta olduğu görülmektedir. Ortak çıkarılan noktalar, elle çıkan noktalara oranlandığında verilerin % 98.63 tutarlı olduğu görülmektedir. Otomatik çıkarılan küre yüzeyden 12 adet nokta elle çıkartılan nokta bulutundan farklı olarak çıkarılmıştır. Elle çıkarılan küre yüzeyde ise 27 adet nokta otomatik olarak çıkarılan küreye ait nokta bulutundan farklıdır.

Çizelge 6. 7. Otomatik ve elle çıkarılan küre yüzey noktalarının karşılaştırılması

Düzlem Adı	Nokta Sayısı
Otomatik (O)	1952
Elle (E)	1967
Eşit Nokta Sayısı (O ∩ E)	1940
O\E	12
E\O	27

Otomatik ve elle çıkarılan küre yüzeyler görsel olarak karşılaştırıldığında iki veri setinin çok büyük oranda örtüştüğü görülmektedir. Her iki veri setinde de dış yüzeylerde birkaç adet nokta farklı olarak görülmektedir. Şekil 6.7a'da çıkarılan iki nokta bulutu çakıştırıldığında mor renkli görülen noktalar otomatik olarak çıkarılan noktalardır. Kenarlarda mavi renkli görülen noktalar ise elle çıkarılan ve otomatik olarak çıkarılan küre yüzeyine ait noktalardan farklı noktaları temsil etmektedir. Bu iki nokta bulutu katı model haline getirildiğinde elle çıkarılan ve sınır bölgelerden kalan alanlarda (gri renkli) bozuk üçgenler olduğu görülmektedir.



Şekil 6.7. Otomatik ve elle çıkarılan küre yüzeylerin görsel olarak karşılaştırılması

7. TARTIŞMA VE ÖNERİLER

Yersel lazer tarayıcılara ile elde edilen nokta bulutu verileri çok sayıda nokta verisi içermektedir. Bu veri seti içinden elle yapılan nesne çıkarma, modelleme çalışmaları çok fazla zaman alan ve hassasiyeti kullanıcıya bağlı olan bir yöntemdir. Ayrıca yersel lazer tarayıcı verileri bilgisayar ortamında çok fazla yer kaplamaktadır. Günümüzde yersel lazer tarayıcı verilerinde otomatik model oluşturma ve otomatik nesne çıkarma yöntemleri araştırma konuları arasında yer almaktadır.

Yersel lazer tarayıcı verilerinde yer alan geometrik nesnelerin otomatik olarak çıkarılması ve modellenmesi, zaman kaybının önüne geçilmesi, verinin boyutunun küçültülmesi ve farklı amaçlar için kullanılması açısından son derece önemlidir.

Günümüzde özellikle kentsel alanda 3 boyutlu veri elde etmede kullanılan cihazlardan yersel lazer tarayıcılar son derece önemli bir yer tutmaya başlamıştır. Kentsel alandaki birçok bina düzlem geometrik bir yapıya sahip olduğundan yersel lazer tarayıcılardan elde edilen veriler üzerinde yapılan geometrik temelli modelleme çalışmaları son derece önem arz etmektedir. Geometrik temelli model oluşturmanın en önemli işlem adımlarından bir tanesi veri seti içindeki noktaların sınıflandırıldığı, kümelere ayrıldığı segmentasyon aşamasıdır.

Bu tez çalışmasında yersel lazer tarayıcı verilerinden elde edilen nokta bulutu verisine RANSAC algoritması uygulanarak nokta bulutu içinde yer alan düzlem, küre, silindir, koni gibi basit geometrik şekle sahip yüzeylerin otomatik olarak çıkarılması sağlanmıştır. Yapılan bu işlem geometrik temelli modelleme yönteminin segmentasyon aşamasını temsil etmektedir.

Çalışma kapsamında laboratuvar ortamında, düzlem, silindir, koni ve küre şekillerini temsil eden objeler kullanılarak, yersel lazer tarayıcı ile deneysel bir test verisi elde edilmiştir. Elde edilen test verisine PCL Kütüphanesi kullanılarak C++ programlama dilinde derlenen RANSAC algoritması uygulanmıştır. Algoritma her bir geometrik yüzey için hem ayrı ayrı derlenmiş hem de tüm geometrik yüzeylerin tek bir programla çıkarılabilmesi için tek olarak derlenmiştir.

RANSAC algoritması ilk olarak test verisi içinden düzlem yüzeylerin çıkarılması için uygulanmıştır. Düzlem yüzeylerin RANSAC algoritması ile çıkarılması işleminde iki farklı yöntem izlenmiştir. Bunlardan birincisi nokta bulutu içinde yer alan noktaların yüzey normallerinden bağımsız düzlem yüzey çıkarma, ikincisi ise yüzey normallerine bağlı olarak düzlem yüzey çıkarma işlemidir.

Yüzey normalinden bağımsız yapılan düzlem çıkarma t eşik değeri mesafesinin çok iyi bir şekilde tespit edilmesi gerekmektedir. Bu değer genellikle deneysel olarak belirlenmektedir. t eşik değeri mesafesi veri seti için olması gerekenden düşük olarak seçildiğinde çıkarılacak düzlem yüzeye eksik nokta verisi atanmakta, büyük seçildiğinde ise düzleme atanmaması gereken noktalar düzleme atanmaktadır. Ancak her koşulda düzlem çıkarma işlemi sadece t eşik değerine göre yapıldığında mutlaka komşu yüzeylerden, çıkartılacak düzlem yüzey üzerine fazlalık veriler atanmaktadır. Bu çalışmada test veri seti için uygun t eşik değeri 1 cm olarak belirlenmiştir.

RANSAC algoritması ile yüzey normaline bağımlı olarak yapılan düzlem çıkarma işleminde t eşik mesafesinin yanında düzlem normali ile nokta normali arasındaki açısal farkı da hesaba katan w normal etki değerinin tanımlanması gerekmektedir. w normal etki değeri, t eşik değeri mesafesi gibi deneysel olarak belirlenmektedir. Test verisi üzerine uygulan yüzey normaline bağımlı RANSAC algoritmasında, w değeri kullanıldığında komşu düzlemlere ait noktaların çıkarılacak düzleme atanmadığı gözlemlenmiştir. Bu sonuçdüzlem yüzey çıkarma işleminde yüzey normalinin dikkate alınmasının daha doğru sonuçlar verdiğini göstermiştir.

Düzlem yüzey çıkarma işleminde uygulanan iki farklı RANSAC algoritmasını veri işleme hızı bakımından değerlendirilecek olursa, yüzey normali hesaplanarak yapılan işlemler, yüzey normali hesaplanmadan yapılan düzlem çıkarma işlemine göre 9-10 kat daha fazla zaman almaktadır. Örneğin test verisine uygulanan bu iki algoritmandan yüzey normalinden bağımsız yapılan işlem 30 saniye sürerken, yüzey normaline bağımlı yapılan düzlem çıkarma işlemi 290 saniye sürmüştür.

Düzlem yüzey çıkarma algoritmaları gerçek arazi verilerine uygulandığında w değerinde fazla bir değişiklik olmamasına rağmen t eşik değeri mesafesinin test

verisine uygulanan değerin 4-5 kat üstüne çıktığı görülmüştür. Araziden elde edilen veriler için t eşik değeri mesafesinin 3.5-5 cm arasında değiştiği görülmüştür. Bu değişimin en önemli nedeni fiziksel yeryüzünde yer alan bina ve objelerin yüzey pürüzlülüğünün fazla olması, tarayıcı ile taranan nesne arasındaki mesafenin değişmesi, nesnelerin birden fazla istasyondan elde edilmesi olarak sıralanabilir.

Arazi verilerine uygulanan RANSAC düzlem yüzey çıkarma algoritmalarını zamansal olarak değerlendirilecek olursa, test verisinde olduğu gibi noktalara ait yüzey normalleri hesaplandığında düzlem yüzey çıkarma işlemi daha fazla zaman almaktadır. Ancak bu zaman nokta bulutunun yoğunluğuna göre değişebilmektedir.

RANSAC algoritması ile silindir, küre ve koni yüzeylerin çıkarım işlemlerinde yüzey normalleri hesaplanarak yüzey çıkarım işlemi gerçekleştirilmiştir.

Test verisi içinde koni ve küre yüzeyler az sayıda nokta verisi içerdiği için çıkarılamamıştır. Bundan dolayı veri setindeki nokta sayısı azaltıp ikinci bir test veri seti oluşturulmuştur. Daha sonra ikinci veri setine algoritmalar uygulanarak bu yüzeylerin otomatik olarak veri seti içinden çıkarılması sağlanmıştır.

Koni, silindir ve küre yüzey çıkarma işleminde gerekli olan modelin oluşturulabilmesi için belirli parametrelerin kullanıcı tarafından tanımlanması gerekmektedir. Bu parametreler küre için maksimum minimum yarıçap uzunluğu, silindir yüzeyler için maksimum minimum yarıçap, epsilon açısı ve silindir modelin aranacağı eksen doğrultusudur. Koni yüzey içinse silindir yüzeyden farklı olarak koni tepe açısının belirlenmesi gerekmektedir. Bu parametrelerin algoritmanın uygulanacağı veri setine göre gerçekçi ve tutarlı bir şekilde belirlenmesi gerekmektedir aksi takdirde çok farklı sonuçlar elde edilebilmektedir. Örneğin test verisi içinden silindir yüzey çıkarılması işleminde, tanımlanması gereken parametre değerleri gerçek dışı tanımlandığında düzlem yüzeye ait noktaların silindir yüzey gibi çıkarıldığı görülmüştür.

Çalışma kapsamında RANSAC algoritması ile otomatik olarak çıkarılan yüzeylerin doğrulukları elle çıkarılan yüzeylerle karşılaştırılarak değerlendirilmiştir. Yapılan değerlendirme sonucunda otomatik ve elle çıkarılan yüzeylerin sahip oldukları ortak noktalar, elle çıkarılan yüzeylere oranlandığında büyük ölçüde çıkarılan yüzeylerin tutarlı olduğu görülmüştür.

Çalışma kapsamında otomatik çıkarılan düzlemlerin görsel olarak değerlendirilmesi için, elle çıkarılan noktalar ile otomatik çıkarılan noktalar geomagic yazılımında yüzey ağı modelleme ile katı model haline getirilmiştir. Katı model haline getirilen yüzeyler çakıştırıldığında, elle çıkarılan noktaların oluşturduğu katı modellerin kenar bölümlerindeki alanların, bozuk olduğu görülmüştür. Bunun en büyük nedeni elle nokta çıkarma işleminin başarısının tamamen operatöre bağlı olmasıdır. Özellikle silindir, koni ve küre gibi yüzeylerin çıkarılmasında, kenarlarda saçınıma uğrayan noktaların modele düşüp düşmediğini belirlemekte zorluk yaşanmaktadır. Buna karşılık otomatik olarak yapılan işlemlerde iset eşik mesafesi, w normal etkisi, yarıçap değerleri gibi parametreler doğru tanımlandığında noktanın modele düşüp düşmediği daha başarılı olarak belirlenebilmektedir.

Yapılan karşılaştırma ve değerlendirmeler sonucunda, RANSAC algoritmasının yersel lazer tarayıcılar ile elde edilen nokta bulutu verisi içinden otomatik yüzey çıkarım işleminde oldukça etkin bir araç olduğu görülmüştür. Özellikle düzlem yüzey çıkarım hızı ve başarısı algoritmanın düzlem yüzeye sahip nesnelerin çıkarımında kullanılabileceğini göstermiştir.

Kentsel alanlarda yapılacak geometrik temelli 3 boyutlu modelleme çalışmalarının segmentasyon aşamasında, RANSAC algoritması etkin bir şekilde kullanılabilecek bir yöntemdir. Ancak RANSAC algoritması ile çıkarılacak düzlemlerin binaya ait olup olmadığının belirlenmesi konusunda farklı karar algoritmalarından yararlanılması gerekmektedir. Örneğin RANSAC algoritması ile bir nokta bulutu içinden tüm düzlem yüzeyler otomatik olarak çıkarılabilirken algoritma ile bu düzlemin zemine mi yoksa cepheye mi ait olup olmadığı belirlenememektedir. Bunu belirleyebilmek için veri setine RANSAC algoritması uygulanmadan önce zemin noktalarını belirleyen bir filtre uygulanabilir ve daha sonra düzlem çıkarım işlemi gerçekleştirilebilir.

Çalışma kapsamında tek bir yersel lazer tarayıcıdan elde edilen nokta bulutu verileri kullanılarak t eşik mesafesine göre geometrik yüzeyler çıkarılmıştır. Ancak bu t eşik mesafesi kullanılan veriye göre değişebilmektedir. Bundan sonraki çalışmalarda t eşik mesafesini etkileyen parametreler detaylı olarak araştırılmalıdır. Örneğin lazer tarayıcı ile taranan nesne arasındaki mesafeye göre bu değerin nasıl

değiştiği araştırılabilir. Benzer şekilde taranan yüzeyin özelliklerinin, t eşik mesafesine etkisinin neler olduğu ortaya konulabilir. Ayrıca farklı lazer tarayıcılardan elde edilen verilerde bu değerin nasıl değiştiğide araştırılabilir.

RANSAC algoritması ile bina cephelerine ait çıkarılan düzlemlere kenar çizgi çıkarım algoritmaları uygulanarak bina cepheleri, pencereler otomatik olarak çizilebilir. Bu çizimler ise binalara ait rölöve çıkarım işlemlerinde altlık veri olarak kullanılabilir.

Kentsel modelleme çalışmalarında, farklı veri kaynaklarından (İHA, LIDAR vb) elde edilecek veriler ile yersel lazer tarayıcılardan elde edilecek veriler birleştirilebilir. Birleştirilen bu verilere RANSAC algoritması uygulanarak, veri seti içinden düzlem çatı ve cephe yüzeylerin çıkarılması sağlanabilir. Farklı veri kaynaklarından elde edilen verilere uygulanan RANSAC algoritmasının başarısı araştırılabilir.

8. KAYNAKÇA

Abdelhafiz, A. (2009), Integrating Digital Photogrammetry and Terrestrial Laser Scanning, Phd Thesis, Institute of Geodesy and Photogrametry Technische Universität Braunschweig, Germany.

Akça D., (2003). Full Automatic Registration of Laser Scanner Point Clouds. Optical 3-D Measurement Techniques VI, Zurich, Switzerland, September 22-25, 2003, vol.I, pp. 330-337.

Al-Manasir, K., And Fraser, C. S. "Registration of Terrestrial Laser Scanner Data Using Imagery", Photogrammetric Record, 2006, vol. 21, issue. 115, pages. 255-268.

Alshawa, M., Boulaassal, H., Landes, T. and Grussenmeyer, P. (2009). "Acquisition and automatic extraction of facade elements on large sites from a low cost laser mobile mapping system." ISPRS/3DARCH09.

Altuntaş, C., Yıldız, F. (2008) "Yersel Lazer Tarayıcı Ölçme Prensipleri ve Nokta Bulutlarının Birleştirilmesi", Jeodezi, Jeoinformasyon ve Arazi Yönetimi Dergisi, 2008/1 Sayı:98 s.20-27.

Arachchige, N. H., Perera, S. N. and Maas, H. G. (2012). "Automatıc Processıng Of Mobıle Laser Scanner Poınt Clouds For Buıldıng Façade Detectıon." Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. XXXIX-B5: 187-192.

Barber, D., Mills, J. and Bryan, P. G., (2001), Laser Scanning and Photogrammetry: 21stcenturymetrology. Proceedings of 18th International Symposium CIPA 2001. Potsdam, Germany, September 18 – 21, pp. 360 – 366.

Besl, P.J. and McKay, N.D., (1992), A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2), pp. 239-256.

Boehler, W. and Marbs A. (2002a), 3D Scanning Instruments. In Proceedings ofInternational Workshop on Scanning for Cultural Heritage Recording Complementing or Replacing Photogrammetry. Corfu, Greece, September, 1 – 2.

Boehler, W., Heinz, G., Marbs, A., Siebold, M. (2002b), 3d scanning software: an introduction. In Proceedings ofInternational Workshop on Scanning for

Cultural Heritage Recording Complementing or Replacing Photogrammetry. Corfu, Greece, September, 1 – 2.

Boulaassal, H., Chevrier, C. and Landes, T. (2010). "From laser data to parametric models: towards an automatic method for building façade modelling." Digital Heritage, Lecture Notes in Computer Science Volume 6436, pp 42-55 Springer.

Boulaassal, H., Landes, T. and Grussenmeyer, P. (2009). " Automatic extraction of planar clusters and their contours on building façades recorded by terrestrial laser scanner." International Journal of Architectural Computing 7(1): 1-20.

Boulaassal, H., Landes, T. and Grussenmeyer, P.(2011). "3D modelling of facade features on large sites acquired by vehicle based laser scanning." Archives of Photogrammetry, Cartography and Remote Sensing edited by Polish Society for Photogrammetry and Remote Sensing 22: 215-226.

Boulaassal, H., Landes, T., Grussenmeyer, P. and Tarsha-Kurdi, F., (2007) "A8_Automatic segmentation of building facades using terrestrial laser data." International Archives of Photogrammetry, Remote Sensing and Spatial Information Systems, page:65-70.

Campbell, R.J. and Flynn, P.J., (2001), A survey of free-form object representation and recognition techniques. Computer Vision and Image Understanding, 81(2), pp. 166-210.

Chen, Y. and Medioni, G., (1992), Object modelling by registration of multiple range images. Image and Vision Computing, 10(3), pp. 145-155.

Çömert, R., Avdan U., Tün, M., Ersoy, M., (2012). "Mimari Belgelemede Yersel Lazer Tarama Yönteminin Uygulanması (Seyitgazi Askerlik Şubesi Örneği)." Harita Teknolojileri Elektronik Dergisi 4(1): 1-18.

Çömert R. ve Avdan U. (2013). " RANSAC Algoritması İle Yersel Lazer Tarayıcı Verilerinden Bina Cephelerinin Otomatik Olarak Çıkarılması." Türkiye Ulusal Fotogrametri ve Uzaktan Algılama Birliği VII. Teknik Sempozyumu (TUFUAB'2013), 23-25 Mayıs, Trabzon.

Demir, N. (2005), Yersel Lazer Tarama ve Fotogrametrinin Birlikte Kullanımı" Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Jeodezi ve Fotogrametri Mühendisliği Anabilim Dalı, İstanbul.

Dorninger, P. and Pfeifer, N. (2008), "A comprehensive automated 3D approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds." Sensors, 8 (11): 7323-7343.

Elkharachy, I., 2008. Towards an automatic registration for terrestrial laser scanner data. PhD Thesis, Institute of geodesy and photogrammetry, Technical university Braunschweig, Germany.

Ergün, B. (2011). Terrestrial Laser Scanning Data Integration in Surveying Engineering, Laser Scanning, Theory and Applications, Prof. Chau-Chang Wang (Ed.), ISBN: 978-953-307-205-0, InTech.

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.

Fröhlich, C. and M. Mettenleiter (2004). "Terrestrial laser scanning—new perspectives in 3D surveying." International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36 (Part 8): W2.

Gümüş, K. (2008) "Yersel Lazer Tarayıcılar ve Konum Doğruluklarının Araştırılması, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Jeodezi ve Fotogrametri Mühendisliği Anabilim Dalı Geomatik Programı, İstanbul.

Gruen, A. and Akca, D., 2005. Least squares 3D surface and curve matching. ISPRS Journal of Photogrammetry and Remote Sensing, 59(3), pp. 151-174.

Hartley, R. and Zisserman A., (2003). Multiple View Geometry in Computer Vision, Second Edition Ambridge, UK, New York: Cambridge University Press, 2003.

Karslı, F. and Preifer, N. (2012) "RANSAC Algortiması İle LIDAR Verilerinden Otomatik Detay Çıkarımı." IV. Uzaktan Algılama ve Coğrafi Bilgi Sistemleri Sempozyumu (UZAL-CBS 2012), 16-19 Ekim, Zonguldak.

Lerma Garcı´a, J.L., Van Genechten, B., Heine, E., Santana Quintero, M., (2008), 3D RiskMapping. Theory and Practice on Terrestrial Laser Scanning. Training Material Based on Practical Applications. Universidad Polite´cnica de Valencia, 261 pp.

Masuda, T. and Yokoya, N., (1995) A robust method for registration and segmentation of multiple range images. Computer Vision and Image Understanding, 61(3), pp. 295-307.

Núñez, A., Buill, F., Regot, J. and Mesa, A. (2012), " Use of robust methods to determine quadratic surfaces: Application to heritage." Journal of Archaeological Science, 40 (2013), 1289-1294.

Overby, J., Bodum, L., Kjems, E. and Lisoe, P (2004). " Automatic 3D building reconstruction from airborne laser scanning and cadastral data using Hough transform." International Archives of Photogrammetry and Remote Sensing 35(B3): 296-301.

Park, S.Y. and Subbarao, M., (2003), A fast point-to-tangent plane technique for multi-view registration. IEEE International Conference on 3-D Digital Imaging and Modeling, Banff, October 6-10, pp. 276-283.

Pu, S. (2008), "Automatic building modeling from terrestrial laser scanning." Advances In 3d Geoinformation Systems, Part II, pp. 147-160, Publisher: Springer Berlin Heidelberg.

Pu, S. and Vosselman, G.(2006) "Automatic extraction of building features from terrestrial laser scanning." International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36(5): 25-27.

Pu, S. and Vosselman, G. (2007), "Extracting windows from terrestrial laser scanning." ISPRS Workshop on Laser Scanning ve SilviLaser, Espoo, September 12-14, 2007, Finland.

Pu, S. and Vosselman, G. (2009a). " Building facade reconstruction by fusing terrestrial laser points and images." Sensors 9(6): 4525-4542.

Pu, S. and Vosselman, G. (2009b), " Knowledge based reconstruction of building models from terrestrial laser scanning data." ISPRS Journal of Photogrammetry and Remote Sensing 64(6): 575-584.

Rabbani, T., van Den Heuvel, F. A., Vosselmann, G., (2006). "Segmentation of point clouds using smoothness constraint." International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36(5): 248-253.

Reshetyuk, Y. (2009). Self-calibration and direct georeferencing in terrestrial laser scanning, Doctoral thesis in Infrastructure, Geodesy, Royal Institute of Technology (KTH) Department of Transport and Economics Division of Geodesy, Stockholm, Sweden.

Riveiro, B., Morer, P., Arias, P., De Arteaga, I. "Terrestrial Laser Scanning and Limit Analysis of Masonry Arch Bridges", Construction and Building Materials, 2011 vol:25 issue (4): pages 1726-1735.

Rusu, R. B., (2008), " Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments." Phd. Thesis, Institute for Informatic, Technische Universität München, Germany.

Rusu, R. B., and Cousins, S. (2011). 3D is HERE: Point Cloud Library (PCL). In Robotics and Automation (ICRA), 2011 IEEE International Conference on (pp. 1-4). IEEE.

Rutzinger, M., Elberink, S.O., Pu, S. and Vosselman, G. (2009). "Automatic extraction of vertical walls from mobile and airborne laser scanning data." The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 38(Part 3): W8.

Sapkota, P.P., (2008). "Segmentation of coloured point cloud data." Master Thesis, International Institute for Geo-Information Science and Earth Observation, Specialisation: Geoinformatics, Enschede, Netherlands.

Schnabel, R., Wahl, R. and Klein, R. (2007) "Efficient RANSAC for Point-Cloud Shape Detection. " Computer Graphics Forum, Wiley Online Library.

Stephan, A., Heinz, I., Mettenleiter, M., Hartl, F. and Frohlich, C. (2002). "Interactive modelling of 3D-environments." International Workshop on Robot and Human Interactive Communication, Berlin, Germany, Sept. 25-27, Proceedings of the IEEE.

Tarsha-Kurdi, F., Landes, T., and Grussenmeyer, P. (2007). Hough-Transform and Extended Ransac Algorithms for Automatic Detection of 3d Building Roof Planes From Lidar Data. International Archives of Photogrammetry, Remote Sensing and Spatial Information Systems, 36, 407-412.

Temiz, M.S., Külür, S. (2011), Video Görüntülerinin Kaçış Noktaları Yardımıyla Rektifikasyonu., TMMOB Harita ve Kadastro Mühendisleri Odası 13. Türkiye Harita Bilimsel ve Teknik Kurultayı 18-22 Nisan, Ankara.

Tovari, T. (2006). "Segmentation Based Classification of Airborne Laser Scanner Data." Universitat Karlshure, Karlshure, Gearmany.

Turhan, Ç. ve Serçe, F. C, (2012), "C++ Dersi: Nesne Tabanlı Programlama", Yazarın Kendi Yayını.

Vieira, M. and Shimada K. (2005) " Surface Extraction from Point-Sampled Data through Region Growing." International Journal of CAD/CAM 5(1).

Vosselman, G. and Dijkman S. (2001). "3D building model reconstruction from point clouds and ground plans." International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences 34 (3/W4): 37-44.

Vosselman, G., Gorte, B., Sithole, G. and Rabbani, T. (2004). "Recognising structure in laser scanner point clouds." International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 46(8): 33-38.

Wang, R., Bach, J. Ferrie, F. P. (2011), Window detection from mobile LiDAR data. Applications of Computer Vision (WACV), IEEE Workshop on, 5-7 Jan. 2011. Kona Hawaii.

Woo H., Kang, R., Wang, S., and Lee, K.H., (2002), "A new segmentation method for point cloud data." International Journal of Machine Tools and Manufacture 42(2): 167-178.

Zhang, Z., (1994), Iterative point matching for registration of free-form curves and surfaces. International Journal of Computer Vision, 13(2), p.p.119-152.

Zogg, H. M. (2008). Investigations of High Precision Terrestrial Laser Scanning with Emphasis on the Development of a Robust Close-Range 3D-Laser Scanning System. Phd Thesis, Institute of Geodesy and Photogrammetry. ETH Zurich.

9. EKLER

EK-1. Günümüzde Kullanılan Yersel Lazer Tarayıcılarıların Teknik Özellikleri

(Ergun 2011)

Firma	Çeşit	Ölçüm Menzili m	Ölçülen Nokta	Tarama Açısı	Mesafe hassasiyeti	Konum hassasiyeti	Işık demeti
Leica	Scan Station	300 m (90% Yansıma)	4000 nokta/sn	$270^0(D)$ $360^0(Y)$	4 mm @ 50 m	6 mm @ 50 m	4 mm @ 50 m
	HDS3000	300 m (90% Yansıma)	4000 nokta/sn	$270^0(D)$ $360^0(Y)$	4 mm @ 50 m	6 mm @ 50 m	4 mm @ 50 m
	HDS4500	53.5 m	500000 nokta/sn	$310^0(D)$ $360^0(Y)$	5mm+120p p m	13.7mm @ 25 m	8.5 mm @ 25 m
	HDS6000	79 m (80% Yansıma)	500000 nokta/sn	$310^0(D)$ $360^0(Y)$	5 mm @ 50 m	10 mm @ 50 m	3 mm + 0.22mrad
Optech	Ilris 3D	3m–1500m (80% Yansıma)	2500 nokta/sn	$310^0(D)$ $360^0(Y)$	7 mm @ 100	8 mm @ 100 m	0.00974^0
Riegl	LMS-Z420	2m-1000m (80% Yansıma)	12000 nokta/sn	$0-80^0(D)$ $0-360^0(Y)$	8 mm @ 50 m	10 mm @ 50 m	0.25 mrad
	LMS-Z390	1m-300m (80% Yansıma)	11000 nokta/sn	$0-80^0(D)$ $0-360^0(Y)$	4 mm @ 50 m	6 mm @ 50 m	0.25 mrad
	LMS-Z210	4m-650m (80% Yansıma)	12000 nokta/sn	$0-80^0(D)$ $0-360^0(Y)$	10 mm @ 50 m	15 mm @ 50 m	2.7 mrad
Z – F	Imager 5006	1m – 79 m	500000 nokta/sn	$310^0(D)$ $360^0(Y)$	m m @ 50	1 mm @ 50 m	0.22 mrad
Faro	LS 880	1 m – 80 m	12000 nokta/sn	$320^0(D)$ $360^0(Y)$	3 mm	5 mm	0.01^0

EK-2. PCD Dosya Formatı

Bu doküman PCL kütüphanesinde kullanılan PCD (Point Cloud Data) dosya formatını tanıtmak için hazırlanmıştır. PCD dosya formatı içinde tanımlanması gereken parametreler aşağıdaki gibidir. (http://pointclouds.org/documentation/tutorials/pcd_file_format.php).

VERSION: PCD dosya versiyonunu tanımlamaktadır.

FIELDS: Dosya içinde yer alacak her bir alanının/boyutun isimlerini tanımlamaktadır.

Örnek: FIELDS x y z # XYZ verisi

FIELDS x y z rgb # XYZ + renk

SIZE: Dosyadaki her bir alanın/boyutunu genişliğini tanımlamaktadır.

Örnek: *unsigned char*/*char* 1 Bit genişliğe sahiptir.

unsigned short/*short* 2 Bit genişliğe sahiptir.

unsigned int/*int*/*float* 4 Bit genişliğe sahiptir.

double 8 Bit genişliğe sahiptir.

TYPE: Her bir alanın veri tipini tanımlar.

I: pozitif ve negatif olarak sunulan tam sayıları tanımlar int8 (*char*), int16 (*short*), int32(*int*)

U: pozitif olarak sunulan tam sayıları tanımlar uint8 (*unsigned char*), uint16 (*unsigned short*), uint32 (*unsigned int*)

F: Ondalık sayıları tanımlar.

COUNT: Her bir alanın kaç elemandan oluştuğunu tanımlar

WIDTH: Nokta bulunun genişliğini tanımlar. WIDHT iki anlamı ifade eder.

- Eğer veri seti organize edilmemiş bir nokta bulutu (genellikle yersel lazer tarayıcılardan elde edilen nokta bulutları organize edilmemiş nokta bulutlarıdır.) ise WIDTH nokta bulutu içindeki toplam nokta sayısı olarak tanımlanır.
- Eğer nokta bulutu organize edilmiş bir nokta bulutu (Fotoğraflardan üretilen nokta bulutları gibi) ise WIDHT veri içindeki satır satısını tanımlar.

HEIGHT: nokta bulutunun yüksekliğini tanımlar. Bu değer organize edilmemiş bir nokta bulutu için 1, organize edilmiş bir nokta bulutu için ise sutun sayısını ifade eder.

VİEWPOINT: Veri seti içindeki noktalar için bir görüş açısı tanımlanır.

DATA: Nokta bulutunun kayıt edildiği veri tipini tanımlar. PCD dosya formatında ASCII ve binari dosya formatları desteklenmektedir.

PCD dosyası için örnek bir tanımlama aşağıdaki gibidir.

```
# .PCD v.7 - Point Cloud Data file format
VERSION .7
FIELDS x y z rgb
SIZE 4 4 4 4
TYPE F F F F
COUNT 1 1 1 1
WIDTH 213
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 213
DATA ascii
0.93773 0.33763 0 4.2108e+06
0.90805 0.35641 0 4.2108e+06
0.81915 0.32 0 4.2108e+06
0.97192 0.278 0 4.2108e+06
0.944 0.29474 0 4.2108e+06
0.98111 0.24247 0 4.2108e+06
0.93655 0.26143 0 4.2108e+06
0.91631 0.27442 0 4.2108e+06
0.81921 0.29315 0 4.2108e+06
0.90701 0.24109 0 4.2108e+06
0.83239 0.23398 0 4.2108e+06
0.99185 0.2116 0 4.2108e+06
0.89264 0.21174 0 4.2108e+06
0.85082 0.21212 0 4.2108e+06
0.81044 0.32222 0 4.2108e+06
0.74459 0.32192 0 4.2108e+06
```

EK-3. Yüzey Normalinden Bağımsız Düzlem Çıkarma C++ Kodları

```
#include <iostream>
#include <pcl/ModelCoefficients.h>
#include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/filters/extract_indices.h>
#include <pcl/features/normal_3d.h>
#include <pcl/sample_consensus/method_types.h>
#include <pcl/sample_consensus/model_types.h>
#include <pcl/segmentation/sac_segmentation.h>

//Bu program PCL Kütüphanesi kullanılarak nokta bulutu içinden yüzey normalinden bağımsız düzlem yüzey çıkarılmasına ait kodları içermektedir.
typedef pcl::PointXYZ PointT;

int
main (int argc, char** argv)
{
  // Tüm verilerin ihtiyaç duyduğu fonksiyonların tanımlanması
  pcl::PCDReader reader; //nokta bulutu okuyucu
  pcl::SACSegmentation<PointT> seg;//segmentasyon fonksiyonu
  pcl::PCDWriter writer;//diske yazıcı
  pcl::ExtractIndices<PointT> extract;//nokta bulutu içinden model noktalarını çıkarma

  // kullanılacak veri setleri
  pcl::PointCloud<PointT>::Ptr nokta_bulutu (new pcl::PointCloud<PointT>);
  pcl::PointCloud<PointT>::Ptr nokta_bulutu_cikan (new pcl::PointCloud <PointT>);
  pcl::PointCloud<PointT>::Ptr nokta_bulutu_kalan (new pcl::PointCloud<PointT>);
  pcl::ModelCoefficients::Ptr duzlem_parametreleri (new pcl::ModelCoefficients);
  pcl::PointIndices::Ptr duzlem_noktalari (new pcl::PointIndices);
  // Nokta bulutu diskten okunur
  reader.read ("D:\\TEZ PROGRAM\\tez_1.pcd", *nokta_bulutu);
```

```
 std::cerr << "Nokta bulutu: " << nokta_bulutu->points.size ()  << " noktaya sahiptir."  <<
std::endl;

 // düzlem yüzey için segmentasyon nesnelerin tanımlanması ve
  //iterasyon sayısı ve eşik değeri mesafesi (t) nin girilmesi
  seg.setOptimizeCoefficients (true);
  seg.setModelType (pcl::SACMODEL_PLANE);  //model tipi
  seg.setMethodType (pcl::SAC_RANSAC);  //yöntem
  seg.setMaxIterations (1000);    //iterasyon sayısı
  seg.setDistanceThreshold (0.01); //t eşik değeri mesafesi

   int i = 0, nr_points = (int) nokta_bulutu->points.size ();
  // orijinal noktanın %10'i kalana kadar algoritma dönsün
  while (nokta_bulutu->points.size () > 0.10 * nr_points)
  {

 // En büyük düzlemin nokta bulutu içinden segmentasyonu
    seg.setInputCloud (nokta_bulutu);
    seg.segment (*duzlem_noktalari, *duzlem_parametreleri);
          std::cerr << "Duzlem parametreleri: " << *duzlem_parametreleri << std::endl;
    if (duzlem_noktalari->indices.size () == 0)
    {
      std::cerr << "Verilen veriseti icin duzlem model bulunamadı." << std::endl;
      break;
    }

  // düzlem yüzeye düşen noktalarından nokta bulutunda çıkarılması
  extract.setInputCloud (nokta_bulutu);
  extract.setIndices (duzlem_noktalari);
  extract.setNegative (false);
  extract.filter (*nokta_bulutu_cikan);
  std::cerr << "Cıkarılan duzlem yuzey: "      <<*nokta_bulutu_cikan->width
*nokta_bulutu_cikan->height << " noktaya sahiptir." << std::endl;

 // Cıkarılan düzlem diske yazdırılır.
```

```
 std::stringstream ss; ss << "D:\\TEZ PROGRAM\\ duzlem_" << i << ".pcd";
writer.write<pcl::PointXYZ> (ss.str (), *nokta_bulutu_cikan, false);

//kalan noktalar
extract.setNegative (true);
extract.filter (*nokta_bulutu_kalan);
nokta_bulutu.swap (nokta_bulutu_kalan);
   i++;
  }
  system ("PAUSE");
  return (0);
}
```

EK-4. Yüzey Normaline Bağımlı Düzlem Çıkarma C++ Kodları

```
#include <pcl/ModelCoefficients.h>
#include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/filters/extract_indices.h>
#include <pcl/filters/passthrough.h>
#include <pcl/features/normal_3d.h>
#include <pcl/sample_consensus/method_types.h>
#include <pcl/sample_consensus/model_types.h>
#include <pcl/segmentation/sac_segmentation.h>
#include <pcl/sample_consensus/sac_model_normal_sphere.h>

typedef pcl::PointXYZ PointT;

int
main (int argc, char** argv)
{
  // ihtiyac duyulan fonksiyonları tanımla
  pcl::PCDReader reader;
  pcl::PassThrough<PointT> pass;
  pcl::NormalEstimation<PointT, pcl::Normal> ne;
  pcl::SACSegmentationFromNormals<PointT, pcl::Normal> seg;
  pcl::PCDWriter writer;
  pcl::ExtractIndices<PointT> extract;
  pcl::ExtractIndices<pcl::Normal> extract_normals;
  pcl::search::KdTree<PointT>::Ptr tree (new pcl::search::KdTree<PointT> ());

  // Nokta bulutu için programda kullanılacak nokta bulutu veri setlerini tanımla
  pcl::PointCloud<PointT>::Ptr cloud (new pcl::PointCloud<PointT>);
  pcl::PointCloud<PointT>::Ptr cloud_c (new pcl::PointCloud<PointT>);
  pcl::PointCloud<PointT>::Ptr cloud_k (new pcl::PointCloud<PointT>);
  pcl::PointCloud<pcl::Normal>::Ptr cloud_normal (new pcl::PointCloud<pcl::Normal>);
  pcl::PointCloud<pcl::Normal>::Ptr cloud_normal_k (new
pcl::PointCloud<pcl::Normal>);
```

```
 pcl::ModelCoefficients::Ptr coefficients_duzlem (new pcl::ModelCoefficients);
 pcl::PointIndices::Ptr inliers_duzlem (new pcl::PointIndices) ;

 // Nokta bulutunu oku
 reader.read ("D:\\Cıktı_veriler\\tez_1.pcd", *cloud);
 std::cerr << "Nokta Bulutu: " << cloud->points.size () << " noktaya sahiptir." << std::endl;



//Nokta normallerini hesapla
ne.setSearchMethod (tree);
ne.setInputCloud (cloud);
ne.setRadiusSearch(0.03);
ne.compute (*cloud_normal);

// duzlem yüzey için segmentasyon parametrelerini gir
 seg.setOptimizeCoefficients (true);
 seg.setModelType (pcl::SACMODEL_NORMAL_PLANE);
 seg.setMethodType (pcl::SAC_RANSAC);
 seg.setNormalDistanceWeight (0.1);
 seg.setMaxIterations (1000);
 seg.setDistanceThreshold (0.01);


 int i = 0, nr_points = (int) cloud->points.size ();

/ /orijinal noktanın %10'i kalana kadar algoritma dö nsün /
 while (cloud->points.size ()> 0.10 * nr_points)
 {
  // En büyük düzlemin nokta bulutu içinden segmentasyonu
  seg.setInputCloud (cloud);
        seg.setInputNormals(cloud_normal);
  seg.segment (*inliers_duzlem, *coefficients_duzlem);
        std::cerr << "Duzlem parametreleri: " << *coefficients_duzlem << std::endl;
  if (inliers_duzlem->indices.size () == 0)
```

```
    {
      std::cerr << "Verilen veri seti icin duzlem yuzey hesaplanamamistir." << std::endl;
      break;
    }
  // düzlem yüzeye düşen noktalarından nokta bulutundan çıkar
  extract.setInputCloud (cloud);
  extract.setIndices (inliers_duzlem);
  extract.setNegative (false);
  extract.filter (*cloud_c);
  std::cerr << "Cikarilan duzlem yuzey: " << cloud_c->width * cloud_c->height << "
noktaya sahiptir." << std::endl;

 //Çıkarılan düzlemleri diske yaz
   std::stringstream ss;
   ss << "D:\\RANSAC\\duzlem_" << i << ".pcd"; writer.writeASCII<pcl::PointXYZ>
(ss.str (), *cloud_c, false);

//kalan noktları cloud_k' ya kaydet
extract.setNegative (true);
extract.filter (*cloud_k);
cloud.swap (cloud_k);

   extract_normals.setNegative (true);
   extract_normals.setInputCloud (cloud_normal);
   extract_normals.setIndices (inliers_duzlem);
   extract_normals.filter (*cloud_normal_k);
   cloud_normal.swap (cloud_normal_k);

    i++;
  }
  system ("PAUSE");
  return (0);
}
```

EK-5. Silindir Yüzey Çıkarma C++ Kodları

```
#include <pcl/ModelCoefficients.h>
#include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/filters/extract_indices.h>
#include <pcl/features/normal_3d.h>
#include <pcl/sample_consensus/method_types.h>
#include <pcl/sample_consensus/model_types.h>
#include <pcl/segmentation/sac_segmentation.h>

typedef pcl::PointXYZ PointT;

int
main (int argc, char** argv)
{
  // ihtiyac duyulan fonksiyonları tanımla
  pcl::PCDReader reader;
  pcl::PassThrough<PointT> pass;
  pcl::NormalEstimation<PointT, pcl::Normal> ne;
  pcl::SACSegmentationFromNormals<PointT, pcl::Normal> seg;
  pcl::PCDWriter writer;
  pcl::ExtractIndices<PointT> extract;
  pcl::ExtractIndices<pcl::Normal> extract_normals;
  pcl::search::KdTree<PointT>::Ptr tree (new pcl::search::KdTree<PointT> ());

  // Nokta bulutu için programda kullanılacak nokta bulutu veri setlerini tanımla
  pcl::PointCloud<PointT>::Ptr cloud (new pcl::PointCloud<PointT>);
  pcl::PointCloud<PointT>::Ptr cloud_c (new pcl::PointCloud<PointT>);
   pcl::PointCloud<PointT>::Ptr cloud_k (new pcl::PointCloud<PointT>);
 pcl::PointCloud<pcl::Normal>::Ptr cloud_normals (new pcl::PointCloud<pcl::Normal>);
  pcl::PointCloud<pcl::Normal>::Ptr cloud_normals_k (new
pcl::PointCloud<pcl::Normal>);
  pcl::ModelCoefficients::Ptr coefficients_cylinder (new pcl::ModelCoefficients);
  pcl::PointIndices::Ptr inliers_cylinder (new pcl::PointIndices) ;
```

```
// Nokta bulutunu oku
reader.read ("D:\\Cıktı_veriler\\tez_1.pcd", *cloud);
std::cerr << "Nokta Bulutu: " << cloud->points.size () << " data points." << std::endl;

  //Nokta normallerini hesapla
 ne.setSearchMethod (tree);
 ne.setInputCloud (cloud);
ne.setRadiusSearch(0.03);
 ne.compute (*cloud_normals);

// Silindir yüzey için segmentasyon parametrelerini gir
seg.setOptimizeCoefficients (true);
seg.setModelType (pcl::SACMODEL_CYLINDER);
seg.setMethodType (pcl::SAC_RANSAC);
seg.setNormalDistanceWeight (0.1);
seg.setMaxIterations (2000);
seg.setEpsAngle(0.2);
seg.setAxis(Eigen::Vector3f (0,0,1));
seg.setDistanceThreshold (0.015);
seg.setRadiusLimits (0, 0.25);


 int i = 0, nr_points = (int) cloud->points.size ();
// orijinal noktanın %80'i kalana kadar algoritma dönsün
while (cloud->points.size ()> 0.99 * nr_points)
{
 //En büyük silindirin nokta bulutu içinden segmentasyonu
 seg.setInputCloud (cloud);
        seg.setInputNormals(cloud_normals);
 seg.segment (*inliers_cylinder, *coefficients_cylinder);
        std::cerr << "Silindir Parametreleri: " << *coefficients_cylinder << std::endl;
 if (inliers_cylinder->indices.size () == 0)
 {
   std::cerr << "Verilen veri seti icin silindir yuzey hesaplanamamistır." << std::endl;
```

```
      break;
    }

  // silindir yüzeye düşen noktalarından nokta bulutundan çıkar
  extract.setInputCloud (cloud);
  extract.setIndices (inliers_cylinder);
  extract.setNegative (false);
  extract.filter (*cloud_c);
  std::cerr << "Cikarilan silindir yuzey:: " << cloud_c->width * cloud_c->height << "
noktaya sahiptir." << std::endl;

  //Çıkarılan düzlemleri diske yaz
   std::stringstream ss;
   ss << "D:\\Cıktı_veriler\\silindir\\silindir_" << i << ".pcd";
            writer.writeASCII<pcl::PointXYZ> (ss.str (), *cloud_c, false);

            //kalan noktları cloud_k' ya kaydet
            extract.setNegative (true);
    extract.filter (*cloud_k);
    cloud.swap (cloud_k);

    extract_normals.setNegative (true);
    extract_normals.setInputCloud (cloud_normals);
    extract_normals.setIndices (inliers_cylinder);
    extract_normals.filter (*cloud_normals_k);
    cloud_normals.swap (cloud_normals_k);

     i++;
  }
  system ("PAUSE");
  return (0);
}
```

EK-6. Koni Yüzey Çıkarma C++ Kodları

```
#include <pcl/ModelCoefficients.h>
#include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/filters/extract_indices.h>
#include <pcl/filters/passthrough.h>
#include <pcl/features/normal_3d.h>
#include <pcl/sample_consensus/method_types.h>
#include <pcl/sample_consensus/model_types.h>
#include <pcl/segmentation/sac_segmentation.h>
#include <pcl/sample_consensus/sac_model_cone.h>

typedef pcl::PointXYZ PointT;

int
main (int argc, char** argv)
{
 // ihtiyac duyulan fonksiyonları tanımla
  pcl::PCDReader reader;
  pcl::PassThrough<PointT> pass;
  pcl::NormalEstimation<PointT, pcl::Normal> ne;
  pcl::SACSegmentationFromNormals<PointT, pcl::Normal> seg;
  pcl::PCDWriter writer;
  pcl::ExtractIndices<PointT> extract;
  pcl::ExtractIndices<pcl::Normal> extract_normals;
  pcl::search::KdTree<PointT>::Ptr tree (new pcl::search::KdTree<PointT> ());

 // Nokta bulutu için programda kullanılacak nokta bulutu veri setlerini tanımla
pcl::PointCloud<PointT>::Ptr cloud (new pcl::PointCloud<PointT>);
pcl::PointCloud<PointT>::Ptr cloud_c (new pcl::PointCloud<PointT>);
pcl::PointCloud<PointT>::Ptr cloud_k (new pcl::PointCloud<PointT>);
pcl::PointCloud<pcl::Normal>::Ptr cloud_normals (new pcl::PointCloud<pcl::Normal>);
pcl::PointCloud<pcl::Normal>::Ptr cloud_normals_k (new
pcl::PointCloud<pcl::Normal>);
```

```
pcl::ModelCoefficients::Ptr coefficients_cone (new pcl::ModelCoefficients);
pcl::PointIndices::Ptr inliers_cone (new pcl::PointIndices) ;
   // Nokta bulutunu oku
  reader.read ("D:\\Cıktı_veriler\\tez_2.pcd", *cloud);
  std::cerr << "Nokta Bulutu " << cloud->points.size () << " noktaya sahiptir." << std::endl;

   ///Nokta normallerini hesapla
   ne.setSearchMethod (tree);
   ne.setInputCloud (cloud);
   ne.setKSearch (50);
         //ne.setRadiusSearch(0.03);
   ne.compute (*cloud_normals);

  // Koni yüzey için segmentasyon parametrelerini gir
  seg.setOptimizeCoefficients (true);
  seg.setModelType (pcl::SACMODEL_CONE);
  seg.setMethodType (pcl::SAC_RANSAC);
  seg.setNormalDistanceWeight (0.08);
  seg.setMinMaxOpeningAngle(5.0 / 180.0  * M_PI, 85.0/ 180.0 * M_PI);
  seg.setAxis(Eigen::Vector3f (0, 0, 1));
  seg.setEpsAngle(0.1);
  seg.setMaxIterations (1000);
  seg.setDistanceThreshold (0.033);


   int i = 0, nr_points = (int) cloud->points.size ();
  //orijinal noktanın %80'i kalana kadar algoritma dönsün
  while (cloud->points.size ()> 0.80* nr_points)
  {
   // En büyük koninin nokta bulutu içinden segmentasyonu
   seg.setInputCloud (cloud);
         seg.setInputNormals(cloud_normals);
   seg.segment (*inliers_cone, *coefficients_cone);
         std::cerr << "Silindir parametreleri: " << *coefficients_cone << std::endl;
   if (inliers_cone->indices.size () == 0)
```

```
    {
      std::cerr << "Verilen veri seti icin koni yuzey hesaplanamamistir" << std::endl;
      break;
    }
    // koni yüzeye düşen noktalarından nokta bulutundan çıkar.
  extract.setInputCloud (cloud);
  extract.setIndices (inliers_cone);
  extract.setNegative (false);
  extract.filter (*cloud_c);
  std::cerr << "Cikarilan koni yuzey: " << cloud_c->width * cloud_c->height << " noktaya
sahiptir." << std::endl;
   //Çıkarılan koni yüzeyleri diske yaz
   std::stringstream ss;
   ss << "D:\\TEZ PROGRAM\\Son_TEZ\\koni_" << i << ".pcd";
            writer.writeASCII<pcl::PointXYZ> (ss.str (), *cloud_c, false);

            //kalan noktları cloud_k' ya kaydet
            extract.setNegative (true);
    extract.filter (*cloud_k);
    cloud.swap (cloud_k);

  extract_normals.setNegative (true);
  extract_normals.setInputCloud (cloud_normals);
  extract_normals.setIndices (inliers_cone);
  extract_normals.filter (*cloud_normals_k);
  cloud_normals.swap (cloud_normals_k);

    i++;

  }
  system ("PAUSE");
  return (0);
}
```

EK-7. Küre Yüzey Çıkarma C++ Kodları

```
#include <pcl/ModelCoefficients.h>
#include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/filters/extract_indices.h>
#include <pcl/features/normal_3d.h>
#include <pcl/sample_consensus/method_types.h>
#include <pcl/sample_consensus/model_types.h>
#include <pcl/segmentation/sac_segmentation.h>
#include <pcl/sample_consensus/sac_model_normal_sphere.h>

typedef pcl::PointXYZ PointT;

int
main (int argc, char** argv)
{
 // ihtiyac duyulan fonksiyonları tanımla
  pcl::PCDReader reader;
  pcl::PassThrough<PointT> pass;
  pcl::NormalEstimation<PointT, pcl::Normal> ne;
  pcl::SACSegmentationFromNormals<PointT, pcl::Normal> seg;
  pcl::PCDWriter writer;
  pcl::ExtractIndices<PointT> extract;
  pcl::ExtractIndices<pcl::Normal> extract_normals;
  pcl::search::KdTree<PointT>::Ptr tree (new pcl::search::KdTree<PointT> ());

// Nokta bulutu için programda kullanılacak nokta bulutu veri setlerini tanımla
pcl::PointCloud<PointT>::Ptr cloud (new pcl::PointCloud<PointT>);
pcl::PointCloud<PointT>::Ptr cloud_c (new pcl::PointCloud<PointT>);
pcl::PointCloud<PointT>::Ptr cloud_k (new pcl::PointCloud<PointT>);

pcl::PointCloud<pcl::Normal>::Ptr cloud_normals (new pcl::PointCloud<pcl::Normal>);
pcl::PointCloud<pcl::Normal>::Ptr cloud_normals_k (new
pcl::PointCloud<pcl::Normal>);
```

```
pcl::ModelCoefficients::Ptr coefficients_sphere (new pcl::ModelCoefficients);
pcl::PointIndices::Ptr inliers_sphere(new pcl::PointIndices) ;


// Nokta bulutunu oku
reader.read ("D:\\Cıktı_veriler\\tez_1.pcd", *cloud);
std::cerr << "Nokta Bulutu " << cloud->points.size () << " noktaya sahiptir." << std::endl;


 ///Nokta normallerini hesapla
ne.setSearchMethod (tree);
ne.setInputCloud (cloud);
ne.setRadiusSearch(0.03);
ne.compute (*cloud_normals);

// küre yüzey için segmentasyon parametrelerini gir
seg.setOptimizeCoefficients (true);
seg.setModelType (pcl::SACMODEL_NORMAL_SPHERE);
seg.setMethodType (pcl::SAC_RANSAC);
seg.setNormalDistanceWeight (0.008);
seg.setMaxIterations (10000);
seg.setDistanceThreshold (0.01);
seg.setRadiusLimits (0.10,0.15);

int i = 0, nr_points = (int) cloud->points.size ();
//orijinal noktanın %20'i kalana kadar algoritma dönsü
  while (cloud->points.size ()> 0.10* nr_points)
  {
    // En büyük kürenin nokta bulutu içinden segmentasyonu
  seg.setInputCloud (cloud);
  seg.setInputNormals(cloud_normals);
  seg.segment (*inliers_sphere, *coefficients_sphere);
  std::cerr << "küre parametreleri: " << *coefficients_sphere << std::endl;
    if (inliers_sphere->indices.size () == 0)
    {
```

```
      std::cerr << "Verilen veri seti icin küre yuzey hesaplanamamistir" << std::endl;
      break;
    }

    // küre yüzeye düşen noktalarından nokta bulutundan çıkar.
  extract.setInputCloud (cloud);
  extract.setIndices (inliers_sphere);
  extract.setNegative (false);
  extract.filter (*cloud_c);
  std::cerr << "Cikarilan küre yuzey: " << cloud_c->width * cloud_c->height << " noktaya
sahiptir." << std::endl;

   //Çıkarılan koni yüzeyleri diske yaz
   std::stringstream ss;
   ss << "D:\\TEZ PROGRAM\\Son_TEZ\\kure_" << i << ".pcd";
          writer.writeASCII<pcl::PointXYZ> (ss.str (), *cloud_c, false);

//kalan noktları cloud_k' ya kaydet
extract.setNegative (true);
extract.filter (*cloud_k);
cloud.swap (cloud_k);

  extract_normals.setNegative (true);
  extract_normals.setInputCloud (cloud_normals);
  extract_normals.setIndices (inliers_sphere);
  extract_normals.filter (*cloud_normals_k);
  cloud_normals.swap (cloud_normals_k);

    i++;
  }
  system ("PAUSE");
  return (0);
}
```

EK-8. Geometrik Yüzeylerin Ardışık Çıkarılmasına Ait C++ Kodları

```
#include <pcl/ModelCoefficients.h>
#include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/filters/extract_indices.h>
#include <pcl/features/normal_3d.h>
#include <pcl/sample_consensus/method_types.h>
#include <pcl/sample_consensus/model_types.h>
#include <pcl/segmentation/sac_segmentation.h>
#include <pcl/sample_consensus/sac_model_normal_sphere.h>

typedef pcl::PointXYZ PointT;


// DÜZLEMLERİ ÇIKART
int
main (int argc, char** argv)
{
  //Nesnelerin kullanacagı fonksiyonları tanımla
  pcl::PCDReader reader;
  pcl::PassThrough<PointT> pass;
  pcl::NormalEstimation<PointT, pcl::Normal> ne;
  pcl::SACSegmentationFromNormals<PointT, pcl::Normal> seg;
  pcl::SACSegmentationFromNormals<PointT, pcl::Normal> seg2;
  pcl::SACSegmentation<PointT> seg3;
  pcl::PCDWriter writer;
  pcl::ExtractIndices<PointT> extract;
  pcl::ExtractIndices<pcl::Normal> extract_normals;
  pcl::search::KdTree<PointT>::Ptr tree (new pcl::search::KdTree<PointT> ());

  // Veri Setleri
  pcl::PointCloud<PointT>::Ptr cloud (new pcl::PointCloud<PointT>);
  pcl::PointCloud<PointT>::Ptr cloud_c (new pcl::PointCloud<PointT>);
  pcl::PointCloud<PointT>::Ptr cloud_k (new pcl::PointCloud<PointT>);
```

```
  pcl::PointCloud<pcl::Normal>::Ptr cloud_normal (new pcl::PointCloud<pcl::Normal>);
  pcl::PointCloud<pcl::Normal>::Ptr cloud_normal_k (new
pcl::PointCloud<pcl::Normal>);
  pcl::ModelCoefficients::Ptr coefficients_duzlem (new pcl::ModelCoefficients);
  pcl::PointIndices::Ptr inliers_duzlem (new pcl::PointIndices) ;

  // Read in the cloud data
  reader.read ("D:\\Cikti_veriler\\tez_1.pcd", *cloud);
  std::cerr << "Okunan nokta bulutu: " << cloud->points.size () << " noktaya sahiptir." <<
std::endl;




   // Nokta Normallerin hesapla
    ne.setSearchMethod (tree);
    ne.setInputCloud (cloud);
    ne.setKSearch (50);
   ne.compute (*cloud_normal);

 // Düzlem yüzey çıkarmak için segmentasyon parametrelerini tanımla
  seg.setOptimizeCoefficients (true);
  seg.setModelType (pcl::SACMODEL_NORMAL_PLANE);
  seg.setMethodType (pcl::SAC_RANSAC);
  seg.setNormalDistanceWeight (0.001);
  seg.setMaxIterations (1000);
  seg.setDistanceThreshold (0.01);

   int i = 0, nr_points = (int) cloud->points.size ();

  // %10 nokta kalana kadar algortimayı döndür.
  while (cloud->points.size ()> 0.10 * nr_points)
  {
    // En büyük düzlem yüzeyi çıkart.
```

```
  seg.setInputCloud (cloud);
        seg.setInputNormals(cloud_normal);
  seg.segment (*inliers_duzlem, *coefficients_duzlem);
        std::cerr << "duzlem parametreleri: " << *coefficients_duzlem << std::endl;
  if (inliers_duzlem->indices.size () == 0)
  {
    std::cerr << "Verilen veri seti için düzlem model hesaplanamadı." << std::endl;
    break;
  }

 // nokta bulutundan düzleme düşen noktaları çıkart
 extract.setInputCloud (cloud);
 extract.setIndices (inliers_duzlem);
 extract.setNegative (false);
 extract.filter (*cloud_c);
 std::cerr << "Cikarilan Duzlem yuzey: " << cloud_c->width * cloud_c->height << "
noktaya sahiptir." << std::endl;

  std::stringstream ss;
  ss << "D:\\Cikti_veriler\\yeni\\duzlem_" << i << ".pcd";
        writer.write<pcl::PointXYZ> (ss.str (), *cloud_c, false);

        extract.setNegative (true);
  extract.filter (*cloud_k);
  cloud.swap (cloud_k);

 extract_normals.setNegative (true);
 extract_normals.setInputCloud (cloud_normal);
 extract_normals.setIndices (inliers_duzlem);
 extract_normals.filter (*cloud_normal_k);
 cloud_normal.swap (cloud_normal_k);

  i++;
 }
```

```
 writer.write ("D:\\test_1_kalan.pcd", *cloud_k, false);

 //SİLİNDİRLERİ ÇIKART

// Veri setleri
pcl::PointCloud<PointT>::Ptr cloud_c1 (new pcl::PointCloud<PointT>);
pcl::PointCloud<PointT>::Ptr cloud_k1 (new pcl::PointCloud<PointT>);
pcl::PointCloud<pcl::Normal>::Ptr cloud_normal_k1 (new
pcl::PointCloud<pcl::Normal>);
pcl::ModelCoefficients::Ptr coefficients_silindir (new pcl::ModelCoefficients);
pcl::PointIndices::Ptr inliers_silindir (new pcl::PointIndices) ;

 // silindir model için gerekli parametreleri tanımla
 seg.setOptimizeCoefficients (true);
 seg.setModelType (pcl::SACMODEL_CYLINDER);
 seg.setMethodType (pcl::SAC_RANSAC);
 seg.setNormalDistanceWeight (0.01);
 seg.setMaxIterations (1000);
 seg.setEpsAngle(0.2);
 seg.setAxis(Eigen::Vector3f (0,0,1));
 seg.setDistanceThreshold (0.01);
 seg.setRadiusLimits (0.00, 0.10);

  int j = 0, nr_points2 = (int) cloud_k->points.size ();
 // %90nokta kalana kadar algortimayı döndür.
 while (cloud_k->points.size ()> 0.90 * nr_points2)
 {
   // En büyük silindiri çıkart
   seg.setInputCloud (cloud_k);
          seg.setInputNormals(cloud_normal_k);
   seg.segment (*inliers_silindir, *coefficients_silindir);
          std::cerr << "cylinder coefficients: " << *coefficients_silindir<< std::endl;
   if (inliers_silindir->indices.size () == 0)
   {
     std::cerr << "Could not estimate a cylinder model for the given dataset." << std::endl;
```

```
    break;
   }


  // silindir model üzerine düşen noktaları nokta bulutundan çıkart
  extract.setInputCloud (cloud_k);
  extract.setIndices (inliers_silindir);
  extract.setNegative (false);
  extract.filter (*cloud_c1);
  std::cerr << "PointCloud representing the cylinder component: " << cloud_c1->width *
cloud_c1->height << " data points." << std::endl;

   std::stringstream ss;
   ss << "D:\\Cikti_veriler\\yeni\\silindir_" << j << ".pcd";
          writer.write<pcl::PointXYZ> (ss.str (), *cloud_c1, false);

   extract.setNegative (true);
   extract.filter (*cloud_k1);
   cloud_k.swap (cloud_k1);
   extract_normals.setNegative (true);
   extract_normals.setInputCloud (cloud_normal_k);
   extract_normals.setIndices (inliers_silindir);
   extract_normals.filter (*cloud_normal_k1);
   cloud_normal_k.swap (cloud_normal_k1);

    j++;
  }


  //KÜRELERİ ÇIKART

  // veri setleri
  pcl::PointCloud<PointT>::Ptr cloud_c2 (new pcl::PointCloud<PointT>);
  pcl::PointCloud<PointT>::Ptr cloud_k2 (new pcl::PointCloud<PointT>);
```

```
  pcl::PointCloud<pcl::Normal>::Ptr cloud_normal_k2 (new
pcl::PointCloud<pcl::Normal>);
  pcl::ModelCoefficients::Ptr coefficients_kure (new pcl::ModelCoefficients);
  pcl::PointIndices::Ptr inliers_kure (new pcl::PointIndices) ;


  //küre modeli için gerekli parametreler
 seg.setOptimizeCoefficients (true);
 seg.setModelType (pcl::SACMODEL_NORMAL_SPHERE);
 seg.setMethodType (pcl::SAC_RANSAC);
 seg.setNormalDistanceWeight (0.008);
 seg.setMaxIterations (10000);
 seg.setDistanceThreshold (0.01);
 seg.setRadiusLimits (0.09,0.11);

    int r = 0, nr_points3 = (int) cloud_k1->points.size ();
  //orijinal noktanın %95'i kalana kadar algoritma dönsün
  while (cloud_k1->points.size ()> 0.95* nr_points3)
  {
    // En büyük kürenin nokta bulutu içinden segmentasyonu
    seg.setInputCloud (cloud_k1);
            seg.setInputNormals(cloud_normal_k1);
    seg.segment (*inliers_kure, *coefficients_kure);
            std::cerr << "koni parametreleri: " << *coefficients_kure<< std::endl;
    if (inliers_kure->indices.size () == 0)
    {
      std::cerr << "Verilen veri seti icin koni yuzey hesaplanamamistir" << std::endl;
      break;
    }

    // koni yüzeye düşen noktalarından nokta bulutundan çıkar.
  extract.setInputCloud (cloud_k1);
  extract.setIndices (inliers_kure);
extract.setNegative (false);
```

```
extract.filter (*cloud_c2);
std::cerr << "Cikarilan koni yuzey: " << cloud_c2->width * cloud_c2->height << "
noktaya sahiptir." << std::endl;

//Çıkarılan koni yüzeyleri diske yaz
std::stringstream ss;
ss << "D:\\Cikti_veriler\\yeni\\kure_" << r << ".pcd";
writer.write<pcl::PointXYZ> (ss.str (), *cloud_c2, false);

//kalan noktları cloud_k2' ya kaydet
extract.setNegative (true);
extract.filter (*cloud_k2);
cloud_k1.swap (cloud_k2);

extract_normals.setNegative (true);
extract_normals.setInputCloud (cloud_normal_k1);
extract_normals.setIndices (inliers_kure);
extract_normals.filter (*cloud_normal_k2);
cloud_normal_k1.swap (cloud_normal_k2);

r++;

}

writer.write ("D:\\test_3_kalan.pcd", *cloud_k1, false);

// KONİ ÇIKARMA

// Veri Setleri

pcl::PointCloud<PointT>::Ptr cloud_c3 (new pcl::PointCloud<PointT>);
pcl::PointCloud<PointT>::Ptr cloud_k3 (new pcl::PointCloud<PointT>);
pcl::PointCloud<pcl::Normal>::Ptr cloud_normal_k3 (new
pcl::PointCloud<pcl::Normal>);
pcl::ModelCoefficients::Ptr coefficients_koni (new pcl::ModelCoefficients);
pcl::PointIndices::Ptr inliers_koni (new pcl::PointIndices) ;
```

```
//Koni modeli için gerekli parametreler
seg.setOptimizeCoefficients (true);
seg.setModelType (pcl::SACMODEL_CONE);
seg.setMethodType (pcl::SAC_RANSAC);
seg.setNormalDistanceWeight (0.08);
seg.setMinMaxOpeningAngle(5.0 / 180.0  * M_PI, 85.0/ 180.0 * M_PI);
seg.setAxis(Eigen::Vector3f (0, 0, 1));
seg.setEpsAngle(0.1);
seg.setMaxIterations (1000);
seg.setDistanceThreshold (0.033);

int m = 0, nr_points4 = (int) cloud_k2->points.size ();
//orijinal noktanın %20'i kalana kadar algoritma dönsün
while (cloud_k2->points.size ()> 0.10* nr_points4)
{
// En büyük koninin nokta bulutu içinden segmentasyonu
seg.setInputCloud (cloud_k2);
seg.setInputNormals(cloud_normal_k2);
seg.segment (*inliers_koni, *coefficients_koni);
std::cerr << "koni parametreleri: " << *coefficients_koni << std::endl;
if (inliers_koni->indices.size () == 0)
u

// koni yüzeye düşen noktalarından nokta bulutundan çıkar.
extract.setInputCloud (cloud_k2);
extract.setIndices (inliers_koni);
extract.setNegative (false);
extract.filter (*cloud_c3);
std::cerr << "Cikarilan koni yuzey: " << cloud_c3->width * cloud_c3->height << "
noktaya sahiptir." << std::endl;

//Çıkarılan koni yüzeyleri diske yaz
std::stringstream ss;
ss <<  "D:\\Cikti_veriler\\yeni\\koni_" << m << ".pcd";
```

```
writer.write<pcl::PointXYZ> (ss.str (), *cloud_c3, false);

//kalan noktları cloud_k2' ye kaydet
extract.setNegative (true);
extract.filter (*cloud_k3);
cloud_k2.swap (cloud_k3);

extract_normals.setNegative (true);
extract_normals.setInputCloud (cloud_normal_k2);
extract_normals.setIndices (inliers_koni);
extract_normals.filter (*cloud_normal_k3);
cloudd_k2.swap (cloud_normal_k3);

m++;

  }

  system ("PAUSE");
  return (0);
}
```

EK-9. Nokta Bulutu Verilerinin Karşılaştırıldığı Matlab Kodları

```
set1 = veri_oto;
set2 = veri_el;

verisayisi1 = length(set1);
verisayisi2 = length(set2);
esitNoktaSayisi = 0;

for i=1:1:verisayisi1
   for k=1:1:verisayisi2
   if( set1(i,1)   == set2(k,1) )
      if (set1(i,2) == set2(k,2) )
         if( set1(i,3)== set2(k,3) )
      esitNoktaSayisi = esitNoktaSayisi + 1;
      esitNokta(esitNoktaSayisi,1) = set1(i,1);
      esitNokta(esitNoktaSayisi,2) = set1(i,2);
      esitNokta(esitNoktaSayisi,3) = set1(i,3);
         end
      end
   end
   end
end
Conten = who;
dlmwrite('esitNokta.txt', esitNokta, 'precision','%.3f', 'newline','pc', 'delimiter',' ');
yuzdeBenzerlikSet1 =  esitNoktaSayisi / verisayisi1 *100;
yuzdeBenzerlikSet2 =  esitNoktaSayisi / verisayisi2 *100;
```