ENDÜSTRİYEL ROBOTLAR İÇİN KOLAY PROGRAMLAMA ARAYÜZ TASARIMI VE GERÇEKLEMESİ

Yüksek Lisans Tezi

Cihan UYANIK

Eskişehir 2017

ENDÜSTRİYEL ROBOTLAR İÇİN KOLAY PROGRAMLAMA ARAYÜZ TASARIMI VE GERÇEKLEMESİ

Cihan UYANIK

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Metin ÖZKAN

Eskişehir

Anadolu Üniversitesi

Fen Bilimleri Enstitüsü

Aralık 2017

JÜRİ VE ENSTİTÜ ONAYI

Cihan UYANIK'ın "Endüstriyel Robotlar için Kolay Programlama Arayüz Tasarımı ve Gerçeklemesi" başlıklı tezi 19/12/2017 tarihinde aşağıdaki jüri tarafından değerlendirilerek "Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği"nin ilgili maddeleri uyarınca, Bilgisayar Mühendisliği Anabilim dalında Yüksek Lisans tezi olarak kabul edilmiştir.

		Unvanı-Adı Soyadı	İmza
Üye (Tez Danışmanı)	:	Doç. Dr. Metin ÖZKAN	……..
Üye	:	Prof. Dr. Osman PARLAKTUNA	……..
Üye	:	Doç. Dr. Serkan GÜNAL	……..

……..

Enstitü Müdürü

ÖZET

ENDÜSTRİYEL ROBOTLAR İÇİN KOLAY PROGRAMLAMA ARAYÜZ TASARIMI VE GERÇEKLEMESİ

Cihan UYANIK
Bilgisayar Mühendisliği Anabilim Dalı
Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Aralık 2017
Danışman: Doç. Dr. Metin ÖZKAN

Endüstriyel robotlar, sağlamış oldukları üretim standardizasyonu, hız ve kararlılık sayesinde modern üretim sistemlerinin vazgeçilmezi haline gelmiş olup, kullanım talepleri günden güne artmaktadır. Endüstriyel robotların hızla gelişmesine ve faaliyet alanlarının çeşitlenmesine rağmen, programlama teknikleri aynı hızda gelişmemektedir. Hali hazırda kullanılan robot programlama teknikleri, genellikle robot üreticisi tarafından sağlanan el terminali kullanımı ile hareketlerin öğretilmesini ya da robot üreticisi tarafından sunulan ve üreticiye özgü programlama dili ile kod yazılmasını kapsamaktadır. Bu teknikler, endüstriyel robotların programlanmasını kapsamlı eğitim ve deneyim gerektiren zor bir süreç haline getirmektedir. Bu durumda, robot programlama işlemini gerçekleştirecek, ilgili robota hakim, deneyimli operatörlere ihtiyaç duyulmaktadır. Bu, önemli bir maliyet oluşturmaktadır. Seri üretim yapan bir üretici için ortaya çıkan bu maliyet göz ardı edilebilir. Fakat küçük ve orta ölçekli ürün çeşitliliğinin çok, ürün adedinin az olduğu üreticiler için bu maliyetlere katlanılamamaktadır. Bu tez çalışması ile söz konusu probleme çözüm olabilme potansiyeli taşıyan, operatör deneyimi ve becerisine ihtiyaç duymayan bir kolay programlama yöntemi geliştirilmiştir. Geliştirilen yöntem, görüntü ve nokta bulutu işleme tekniklerine dayanmakta olup, basit kullanıcı arayüzü etkileşimleri ile bir endüstriyel robotun programlanmasına olanak sağlamaktadır. Uygulanan yöntemler detaylı matematiksel modeller, kod şemaları ve örnek uygulama görselleri ile açıklanmaktadır. Nihai olarak ortaya çıkan sistemin gerçek bir uygulama için nasıl kullanıldığı gösterilmektedir.

Anahtar Sözcükler: Endüstriyel robotlar, Robot programlama, Görüntü işleme, Nokta bulutları, Kullanıcı arayüzü

ABSTRACT

DESIGN AND IMPLEMENTATION OF SIMPLE PROGRAMMING INTERFACE FOR INDUSTRIAL ROBOTS

Cihan UYANIK

Department of Computer Engineering

Anadolu University, Graduate School of Sciences, December 2017

Supervisor: Assoc. Prof. Dr. Metin ÖZKAN

Industrial robots have become one of the irreplaceable equipment for modern manufacturing systems by their opportunities on production standardization, speed and stability and their usage demands grows day by day. Although industrial robots are developed rapidly and their service areas become diversified, programming techniques of them are not progressed by the same rate. Robot programming techniques which are utilized nowadays, generally, includes motion leading by using a hand terminal which is supplied by the robot manufacturer or coding by a producer specialized programming language which is offered by the manufacturer. These techniques make the programming task a tough and costly process which requires extensive education and experience. In this case, an operator who is experienced and trained for the related robot is required. This situation causes an important cost. The cost could be negligible for a mass production facility. However, it is not possible to afford the cost for small and medium sizes facilities which produces many diverse product range and few in number. In this thesis, a simple programming method is developed to generate a solution which does not require operator experience and talent, also applicable, for the mentioned issue. Developed method is based on image and point cloud processing and also provide an opportunity to program an industrial robot by simple user interface interactions. The applied techniques are explained by detailed mathematical models, pseudocodes and sample application visualizations. Eventually, it is presented how to utilize the designed and implemented system for a real-world application.

Keywords: Industrial robots, Robot programming, Image processing, Point clouds, Graphical user interface

TEŞEKKÜR

Birlikte yer aldığımız tüm akademik ve idari çalışmalarda paha biçilemez önerileri ile çalışma verimini yükselten, karşılaşmış olduğum zorluklarda desteğini esirgemeyen, yürütmekte olduğum akademik faaliyetlerde kendine layık olmaya çalıştığım ve kendime örnek aldığım, engin bilgi ve tecrübe birikimine sahip Sayın Metin Özkan'a en içten teşekkürlerimi sunarım.

Lisans ve yüksek lisans eğitimim boyunca sağlamış olduğu eşsiz tavsiyeler ile kariyerime yön vermemi sağlayan, her sıkıntıya düştüğümde fikirleri ile bana ışık tutan Sayın Osman Parlaktuna'ya bütün kalbimle teşekkür ederim. Eğitim-öğretim hayatım boyunca üzerimde emeği olan diğer bütün hocalarıma da teşekkürlerimi sunarım.

Sahip olduğu bilgi birikimi, tecrübe ve derin bakış açısı ve inanılmaz dikkati ile çalışmalarıma sağladığı yapıcı eleştiriler ve iyileştirici önerileri için arkadaşım Sayın Savaş Okyay'a en içten teşekkürlerimi sunarım.

Beni bugünlere getiren aileme, eğitim hayatımın sürdürebilmemde maddi manevi desteklerini esirgemeyen yakınlarıma minnettarım.

Cihan UYANIK

Aralık 2017

19/12/2017

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Bu tezin bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalardan bilimsel etik ilke ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilemeyen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; bu çalışmanın Anadolu Üniversitesi tarafından kullanılan "bilimsel intihal tespit programı"yla tarandığını ve hiçbir şekilde "intihal içermediğini" beyan ederim. Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçlara razı olduğumu bildiririm.

.....

(İmza)

Cihan UYANIK

(Öğrencinin Adı Soyadı)

İÇİNDEKİLER

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

GÖRSELLER DİZİNİ

Sayfa

SİMGELER ve KISALTMALAR DİZİNİ

3B	: Üç boyutlu
B	: Endüstriyel robot taban koordinat sistemi
CALIB	: Kamera kalibrasyon nesnesine ait koordinat sistemi
RT0	: Döner tablanın 0 pozisyonu için belirlenen koordinat sistemi
RTP	: Döner tabla desenine iliştirilmiş koordinat sistemi
TCP	: Endüstriyel robot ucu bağlantı noktasına ait koordinat sistemi
TOOL	: Endüstriyel robot ucuna bağlı olan donanıma ait koordinat sistemi

1. GİRİŞ

Endüstriyel robotlar, beyaz eşya, otomobil, havacılık gibi pek çok sanayi alanında üretim süreçlerinde yoğun olarak kullanılmaktadır. Uluslararası Robotik Federasyonunun istatistik grafikleri incelendiğinde robot kullanımının her geçen yıl arttığı görülmektedir. Endüstriyel robotlar üretim süreçlerinde kalite ve üretim hızının artmasını sağlarken, maliyetleri de önemli ölçüde düşürmektedir. Robot fiyatlarının azalması ve robotların ikinci el pazarının olması, küçük ölçekli firmaların da robot temin etmesini mümkün kılmaktadır. Ancak, temel problem robotun temin edilmesinden çok, robotun programlanmasında ortaya çıkmaktadır. Robotun parçalar üzerinde işlem gerçekleştirebilmesi için operatör tarafından programlanması gerekmektedir. Robotun programlanması, her yeni parça için ve her yeni tanımlanan işlem için yapılması gerekmektedir. Bu durumda firmanın robotu temin ederken, programlamada uzman bir operatörü de istihdam etmesi gerekmektedir. Bu durum, firmaya maliyetli gelmekte ya da operatör bulamamaktadır. Bu tez kapsamında, endüstriyel robotların programlanmasını mevcut robot programlama tekniklerine göre çok daha kolaylaştıracak kullanıcı arayüzü geliştirilmesi ve dolayısıyla uzman operatör ihtiyacını azaltacak endüstriyel robot programlama sistemi ortaya çıkarılması hedeflenmiştir.

Endüstriyel robotların geliştirilmesinde, hız, hassasiyet ve kapasiteler anlamında önemli teknolojik ilerlemeler gerçekleşmiştir. Ancak, robotların programlama yöntemlerinde geleneksel yöntemler kullanılmaya devam edilmektedir. Geleneksel iki tip programlama yöntemi mevcuttur. Bunlar, robota hareketi öğretme ve robot üreticisinin sağladığı programlama dili kullanılarak, tüm hareketlerin komut dizisi olarak yazılmasıdır. Bu yöntemler, deneyimli uzman operatör tarafından yapılabilmektedir. Bu yöntemlerin kullanım güçlüğü, seri üretim gerçekleştiren firmalar için sorun teşkil etmemektedir. Bunun sebebi, programlanan endüstriyel robot hücresinin kullanılarak üretilmesi hedeflenen ürünün adedinin fazla olması, programlama için harcanan zamanın ihmal edilebilmesine izin vermektedir. Ancak, küçük ölçekli firmaların ürettiği ürünlerin sayıca çok olmaması ve ürün üzerinde yapılacak işlemlerde çeşitlilik olması nedeniyle, robotun sıkça programlanması gerekmektedir. Tez konusunun seçilmesinde, kaynak işlerinde robot kullanmak isteyen küçük ölçekli bir firma sahibinin dile getirdiği sorun tetikleyici olmuştur. Firma sahibi, iyi bir kaynakçı istihdam etmekte sorun yaşadığını ve bu nedenle kaynak robotu kullanmak istediğini belirtmiştir. Ancak, bu durumda kaynak

robotunu programlayacak uzman bulmanın zor olduğunu belirtmiştir. Robotu programlamayı kolaylaştıracak bir yöntemin ortaya çıkarılması, birçok küçük ölçekli firmanın robot kullanımını kolaylaştıracağı belirtilmiştir. Bu çalışma kapsamında geliştirilecek robot programlama yönteminin, mevcut yöntemlerden çok daha kolay uygulanabilir olması sebebi ile söz konusu soruna çözüm üretme potansiyeli oldukça yüksektir.

1.1. Amaç ve Hedefler

Endüstriyel robotlar, üretim süreçlerinde yoğun olarak kullanılmakta, üretim hızında ve kalitesinde önemli artışlar sağlamaktadır. Böylece, maliyetler önemli ölçüde aşağı çekilmekte, sürdürülebilir kalite elde edilmektedir. Her ne kadar istihdamı azalttığı düşünülse de insanlar için sağlıksız olan çalışma koşullarında robotların yer alması nedeniyle, bu teknolojilerin sosyal anlamda da fayda sağladığı söylenebilir. Son yıllarda, Avrupa'da 4ncü sanayi devrimi olarak lanse edilen ve Endüstri 4.0 olarak adlandırılan üretimde yeni bir sürecin içine girilmiştir. Bu yeni sanayi devrimi sürecinde, ürünün pazara çıkış süresini kısaltma, üretim sürecini esnek hale getirme ve ucuz üretim sağlamak, ana hedefleri oluşturmaktadır. Bu hedeflerin elde edilmesinde, önemli teknolojik gelişmelerin robotlar üzerinde gerçekleşmesi beklenmektedir. Bu süreçten, Avrupa'ya komşu durumundaki ülkemiz de önemli ölçüde etkilenecektir. Bu etki, ülkemiz sürece ayak uydurabilirse olumlu, uyduramazsa olumsuz yönde gerçekleşebilir. Bu bağlamda, küçük ölçekli firmalarında üretimde robot kullanımına ağırlık vermesi gerekecektir. Ancak, özellikle küçük ölçekli firmalar için temel problem robotun temin edilmesinden çok, robotun programlanmasında ortaya çıkmaktadır. Robotun parçalar üzerinde işlem gerçekleştirebilmesi için operatör tarafından programlanması gerekmektedir. Robotun programlaması, her yeni parça için ve her yeni tanımlanan işlem için yapılması gerekmektedir. Bu durumda firmanın robotu temin ederken, programlamada uzman bir operatörü de istihdam etmesi gerekmektedir. Bu firmaya maliyetli gelmekte ya da operatör bulamamaktadır. Bu tezin amacı, endüstriyel robotların programlanmasını mevcut robot programlama tekniklerine göre çok daha kolaylaştıracak yeni bir yöntem ve bu yöntemin uygulanabilirliğine hizmet eden bir kullanıcı arayüzü geliştirilmesidir.

Endüstriyel robot kollarının, bir parça üzerinde endüstriyel işlem uygulayabilmesi için programlanmasının, kullanımı kolay olan bir kullanıcı arayüz üzerinden yapılabilmesi hedeflenmektedir. Kullanıcı, sadece geliştirilmiş olan kullanıcı arayüzü üzerinden işlem uygulanacak parçanın görüntüsünde işlem bölgesini belirleyebilecektir. Belirlenen bölgenin robot çalışma uzayında bulunduğu konum hesaplanarak, işlem görecek parçanın, belirlenen bölge için üç boyutlu nokta bulutu elde edilecektir. Elde edilen nokta bulutu yine aynı kullanıcı arayüzü üzerinden işleme tabi tutulmaya açık halde görüntülenecek ve endüstriyel robotun yapması beklenen işlem için ihtiyaç duyulacak hareket planı çıkartılacaktır. Söz konusu işlemlerin gerçekleştirilebilmesi için ihtiyaç duyulacak programlama paketleri ve bu paketlerin tasarımı, endüstriyel robotların çalışma uzaylarının algılanması süreçlerine destek verecektir.

1.2. Literatür Taraması

Tez kapsamında ele alınan konu, endüstriyel robotların mevcut yaklaşımlara göre daha kolay programlanmasını sağlayacak yöntem ve bu yöntemin destekleyici olan arayüzün geliştirilmesini kapsamaktadır. Bu hedefe ulaşıldığı takdirde, önemli bir probleme çözüm sunulmuş olacaktır. Ortaya koyulmuş olan bu araştırma problemine vurgu yapan yayınlara literatürde ulaşmak mümkündür. Endüstriyel robotların geliştirilmesi, hız, hassasiyet ve kapasiteler anlamında önemli teknolojik ilerlemeler kaydedilmiş olmasına rağmen, bu robotların programlanması hali hazırda kullanılan geleneksel öğretme yöntemlerine dayanmaktadır. Bu yöntemler hataya açık, zaman tüketen yöntemler olması yanı sıra programlama yapan kişinin teknik becerileri ile de doğrudan ilintilidir. Bu yöntemleri kullanan endüstri alanlarında, robot kullanımının verimli olabilmesi için programlanmış olan bu robotun, gerçekleştirilen programlama ile büyük çapta üretim yapılması gerekmektedir [1]. Robot programlama sürecinin uzun sürdüğü ve buna karşılık üretimi yapılan ürünün sayıca az olduğu veya ürün çeşitliliğinin fazlalığından kaynaklanan yeniden programlama gereksiniminin yüksek olduğu işletmelerde endüstriyel robot kullanımının zorluğu ön plana çıkmakta ve bu gibi işletmelerde robotların kullanımından uzak durulmaktadır [2]. Örneğin, otomobil gövdesi kaynağı yapması beklenen bir endüstriyel robotun geleneksel yöntemler ile programlanması 8 aydan daha uzun bir süre almakta, fakat programlanma süreci tamamlanan bir robotun ilgili işi gerçekleştirmesi 16 saatte tamamlanmaktadır. Bu

durumda programlama zamanı, işlem gerçekleştirme zamanının yaklaşık olarak 360 katıdır. Üretim yapan işletmenin böyle bir durumda endüstriyel robot kullanabilmesi için üretilen üründen binlerce üretmesi gerekmektedir. Fakat küçük ve orta ölçekli endüstriyel işletmeler için bu durum söz konusu değildir. Haliyle, bu işletmelerin endüstriyel robot kullanımının sağladığı avantajlardan faydalanmaları mümkün olmamaktadır [3].

Tez çalışması, yukarıda tanımlanan araştırma problemine çözüm olarak, endüstride yüksek ilgi çekme potansiyeli barındıran yeni bir yöntemin geliştirilmesi ve uygulanmasını kapsamaktadır. Endüstriyel robotların programlaması ile ilgili olarak, endüstriyel robot üreticilerinin altyapı olarak desteklediği ve kullanım kılavuzlarında tarif ettiği geleneksel yöntemler hâlihazırda kullanılmaya devam etmektedir. Ayrıca bunun yanında, yeni programlama teknikleri geliştirmek üzere yapılan çalışmalarda literatürde bulunmaktadır. Günümüzde kullanılan endüstriyel robot programlama yöntemlerini (i) çevrimiçi programlama ve (ii) çevrimdışı programlama olarak iki ana kategoriye ayırmak mümkündür.

(i) Çevrimiçi programlama yöntemleri temel olarak robot operatörü tarafından kullanılan kontrol paneli ile robotun işlem ucunun istenilen pozisyonlara getirilerek bu pozisyonların kaydedilmesine ve kaydedilen pozisyonların sırası ile robota tekrarlatılmasından oluşmaktadır [4]. Bu yöntem, eğitimli bir operatör tarafından kolaylıkla uygulanabilecek bir yaklaşım olmasına karşılık, yalnızca karmaşık olmayan işlemlerde kullanılabilmektedir. Ayrıca bu yöntem operatör becerisine dayanmakta ve gelecekte ihtiyaç duyulabilecek değişikliklerin uygulanmasına izin vermemektedir. Robot işlev ucunun, işlem yapılan çalışma bölgesindeki herhangi bir nesneye veya karmaşık geometrik yapıda olan bir ürüne çarpmadan istenen pozisyonlarda hareket ettirilmesi oldukça zor ve zaman alan bir süreçtir. Buna ek olarak programlama sürecinin tamamlanması ve tamamlanan programın güvenilirliğinin sağlanabilmesi için pek çok test sürecinden geçirilmesi gerekmektedir. Bütün bunlara ek olarak programlama esnasında endüstriyel robotun kullanılabilmesi mümkün olmamaktadır. Söz konusu dezavantajlara rağmen bu yöntem günümüzde, yüksek programlama yeteneğine ihtiyaç duymaması ve programlama maliyetinin düşük olması sebebi ile küçük ve orta ölçekli endüstriyel işletmelerde yoğun olarak kullanılmaktadır [3].

Robot operatörünün programlama esnasında takip etmek zorunda olduğu, robot ucu pozisyonu, robot çevresindeki nesnelere çarpma olasılığı, robot konumlanma açıları gibi pek çok parametrenin var olması ve bu parametrelerin takip edilmesinin zorluğundan

dolayı programlama işleminin oldukça zor bir süreç olduğu bilinmektedir. Bu parametrelerin azaltılması için robotun işlev ucunun konumlanma açısının kolaylıkla belirlenmesine olanak sağlayacak ek bir öğretme aracı geliştirmiş ve robot işlev ucunun, işlem görecek noktaya üç boyutlu sistemde hangi açıyla yaklaşması gerektiğinin gösterilmesi için bu aracın kullanılabileceğini önerilmiştir [5]. Bir başka çalışmada geliştirilen ve COSMO ismi verilen bir kuvvet/moment algılayıcısı, operatör tarafından kullanılan kontrol paneline ek olarak kullanılabilen programlama araçlarından biridir [6]. Bu yöntem ile operatör geliştirilen algılayıcıyı tutarak, itme, çekme hareket yönünde çevirme işlemlerini gerçekleştirerek, robot işlev ucunu istenen pozisyonlara getirebilmektedir. Bu sayede operatörün programlama yapabilmek için robot koordinat sistemine hâkim olmasına gerek kalmamaktadır. Fakat her iki yöntemin temel sorunu, operatörün robot çalışma bölgesine girmek zorunda olmasıdır. Bu durum güvenlik problemlerine sebep olmaktadır [7]. Bu yöntemlerle benzer özellikler taşıyan bir başka yöntemde ise, operatör sadece robot işlev ucunun pozisyonunu değil, robotun bütün eklemlerinin pozisyonlarını kuvvet uygulayarak değiştirebilmektedir [8]. Operatör tarafından kuvvet uygulayarak yapılan bu değişiklikler robot kontrol ünitesi tarafından kaydedilmektedir. Kuvvet uygulayarak robot eklemlerinin hareketinin kontrol edilebilmesi, robotun engellere takılmadan işlevini gerçekleştirebilmesine olanak sağlamaktadır. Ayrıca geliştirilen yöntem ile robot ucunun sabit bir yolu izlemesi de sağlanabilmektedir. Gerçekleştirilmiş olan bir başka çalışmada, iki kollu bir robotun gerçekleştirmesi beklenen işlemin, kullanıcı tarafından, robot kollarına kuvvet uygulanarak öğretilmesi amaçlanmıştır. Söz konusu çalışmada robot koluna uygulanan kuvvetler, robot tarafından kaydedilmekte ve öğretme sürecinin tamamlanmasının ardından, söz konusu kuvvetler ve bu kuvvetlerin sebep olduğu eksen hareketlerinin ve nesne temaslarının tekrarlanması beklenmektedir. İlgili çalışmanın, robot dinamik modeline doğrudan bağlı olduğu araştırmacılar tarafından vurgulanmaktadır [9]. Bu yöntem operatörün programlama becerisine bağlı değildir, fakat operatörün robot çalışma bölgesine girmesine ihtiyaç duymaktadır. Bu yöntemlerde [5, 6, 8, 9] temelde robot kola operatör tarafından elle uygulanan kuvvet etkisiyle robotun işlem sırasında hareket etmesi, istenen konum ve yönelimlere getirilmesi ile gerçekleştirilen bir öğretme yaklaşımı söz konusudur. Operatör işlem uygulama sırasında robotun gideceği her bir konum ve yönelimi kaydederken, her bir nokta arasındaki hareketi (doğrusal, noktadan noktaya, vb.) ve işlemi tanımlarken kontrol panelinde kod yazması gerekmektedir. Bu

yöntemlerde, operatörün hem robotu elle hareket ettirmesi ve hem de kontrol panelini kullanması, programlama sürecinde güçlüğe, karışıklığa neden olabileceği gibi operatör deneyimi de gerektirmektedir. Ayrıca, operatörün robota yakın temasta bulunuyor olması güvenlik sorunu da oluşturacaktır. Ayrıca, bu yöntemler yüksek yük kapasiteli ve geniş çalışma uzayına sahip robotlarda kullanıma uygun değildir. Bu tez çalışması kapsamında geliştirilen yöntemde, operatör tüm programlama sürecini kullanıcı arayüzü üzerinden gerçekleştiriyor olması nedeniyle, robotun çalışma uzayı dışında kalarak güvenlik sorunu yaşanmasının önüne geçilebilmektedir. Ayrıca, işlem bölgelerinin belirlenmesi ve noktalar arası hareketleri kullanıcı arayüzü tarafından sunulacak görseller ve kontrol noktaları üzerinden gerçekleştiriliyor olması, geliştirilen kolay programlama arayüzü sayesinde operatör deneyimine olan ihtiyacın oldukça azaltılmasını sağlayacaktır.

Bazı çalışmalar, geleneksel çevrimiçi programlama sırasında, operatörün robota hareketi öğretmesine destek olan sistemlerin geliştirilmesini kapsamaktadır. Operatör kontrol paneli kullanılarak gerçekleştirilen programlama yöntemlerinin büyük sorunlarından birisi, robot işlev ucunun izleyeceği yoldaki hataların giderilmesi veya gerçekleştirilen hareketin hızının değiştirilebilmesi için yeniden programlama gereksiniminin bulunmasıdır. Karşılaşılan bu sorunların üstesinden gelebilmek için otomatik yol düzeltme ve hatalı bölgelerin yeniden düzeltilmesine olanak sağlayan 3 boyutlu grafik arayüzü ve programlama sırasında robot hareketlerinin kontrolünü yönlendirmeye izin veren ses kontrol sistemleri literatürde önerilen yöntemlerden biridir [10]. Bu yöntemde, elle kuvvet uygulayarak gerçekleştirilen öğretme yönteminin uygulamadaki eksikliklerini ve güçlüklerini gidermek üzere ilave yaklaşımlar sunulmaktadır. Bunlardan biri, operatörün robot ucunu elle hareket ettirmesi sırasında öğretilen izlenecek yolun, işlem uygulanacak yoldan sapmalarının giderilmesidir. Bu yaklaşımda, işlem uygulanacak yüzeyden sağlanan herhangi bir bilgi kullanılmamaktadır. Hareketi sırasında kaydedilen, robot ucunun koordinatları kullanılarak çizgi ve eğrilere yakınlaştırma yapılmaktadır. Benzer düzeltme işlemi, geliştirilen yöntemde hassas algılayıcı ile işlem uygulanacak bölgeden alınan nokta bulutu kullanılarak yapılmaktadır. Operatörün, panel ekran üzerinden işlem noktaları seçiminde yaptığı hatalar, işlem uygulanacak cismin yüzeyinden alınan hassas nokta bulutunun işlenmesi ile elde edilen yüzey özellikleri kullanılarak düzeltilmektedir. Böylece, işlem uygulayana değil (robot kol), işlem uygulanacak olana (nesne) referansla düzeltme yapılmış olacaktır. Bu

durumda, operatör hiçbir noktayı işlem uygulanacak yol üzerinde seçemese dahi, seçilen noktaların işlem uygulanacak yol üzerindeki iz düşümleri saptanabilecektir.

Geleneksel yöntemlerden farklı olarak, cisim üzerinde işlem noktalarının işaret kalemiyle ya da lazer ışınıyla işaretlenmesi şeklinde gerçekleştirilen ilgi çekici yöntemlerde bulunmaktadır. Robot eklemlerine elle kuvvet uygulanarak gerçekleştirilen yöntemler robot işlev ucunun izleyeceği yolun hassas bir şekilde öğretilmesine ihtiyaç duymaktadır. Bu yapıya yeni algılayıcılar ekleyerek, yapılan geliştirme ile robot işlev ucunun işlem görecek parçanın üzerine standart bir kalem ile çizilmiş olan hareket yolunu takip etmesi hedeflenmiştir [11]. Robot ucunun çizilmiş olan yolu takip etmesi çalışma bölgesini gören bir kamera ile gerçekleştirilmiş olup, işlem gören parçaya temasın sağlanması için ise kuvvet algılayıcıları kullanılmıştır. Bir başka çalışmada ise benzer şekilde görüntü işleme tabanlı programlama olarak görülmektedir [12]. Bu yöntemde de işlem görecek parçanın üzerine çizilmiş olan hareket yolunun robot tarafından takip edilmesi hedeflenmiştir. Çizilen bu yol 2 boyutlu bir kamera ile algılanmakta olup, derinlik bilgisi (3. boyut) üzerinde çalışılacak parçanın benzetimlik ortamına aktarılarak, robotun hareketi esnasında parça üzerindeki temas yüzeyine çarpıp çarpmadığının kontrol edilmesi prensibi ile tespit edilmektedir. Eğer robot işlev ucu, parça üzerindeki istenen noktaya temas ediyor ise ilgili koordinat kaydedilmektedir. Bu işlemin gerçekleştirilmesi harici bir benzetim yazılımına ihtiyaç duymaktadır. Ortaya koyulan çalışmalardan birinde ise işlem görecek parça üzerine belli yapılarda lazer ışınları yönlendirerek, robot işlev ucunun dolaşabileceği kapalı hareket yollarının tespitine olanak sağlayan bir algoritma geliştirilmiştir [13]. Eğer işlenecek cismin geometrik yapısından dolayı ortaya çıkarılan hareket yolu sınırlı ise 2. Dereceden bir polinom ile 3 boyutlu hareket yolu çıkartılmaktadır. Geliştirilen algoritma görüntü işleme tabanlı olup, robot operatörünün gerçekleştirilecek işlemin başlangıç noktasının ve robot hareketinin parça üzerinden hangi yöne doğru olacağının belirlenmesini sağlamaktadır. Böylece robot hareket planlamasının diğer yöntemlere göre daha sağlıklı ve hızlı yapılabilmesi sağlanmıştır. Belli yapıda lazer ışınlarının ve 3 boyutlu görüntüleme tekniklerinin kullanıldığı bir başka uygulamada ise deri yüzeyinin profilinin çıkartılması amaçlanmıştır [14]. Gönderilen lazer ışınları işlem görecek parçanın üzerinde gezdirilip, bu esnada bölgede bulunan bir analog kamera vasıtası ile işlem görecek deri bölgesinin profili çıkartılmıştır. Benzer şekilde bir başka çalışmada işlem görecek parça üzerine yansıtılan lazer yansı görüntüsünün karakteristiğine uygun olarak robot işlem ucunun uygun pozisyona

getirilmesi hedeflenmiştir [15]. Bu çalışma kapsamında düşürülmesi gerçekleştirilecek görüntü, benzetim ortamında yaratılmakta ve geliştirilmiş olan arayüz üzerinden kontrollü olarak işlem görecek parçaya yansıtılmaktadır. İşlem gerçekleştirecek robotun hareket planı parça üzerine yansıtılmış olan görüntünün takip edilmesi ile gerçekleştirilmiştir. Bu yöntemlerden bazılarında, işlem uygulanacak parça üzerine robot ucun izleyeceği yolun kalemle işaretlenmesi ve kamera kullanımı ile işaretlerin algılanarak robotun izleyeceği yolun saptanması söz konusudur [11, 12]. Bu yöntemlerde, çizgiyi tanıyacak olan kamera ya çalışma uzayında sabitlenmekte ya da robotun ucuna yerleştirilmektedir. Çalışma ortamına sabitlenmiş ise, çizginin tamamının yüzey geometrisine bağlı olarak görünememe ihtimali vardır. Kamera robot ucunda ise, operatörün robotu hareket ettirerek çizginin görüneceği konuma getirmesi gerekmektedir. Ayrıca, çizginin operatör tarafından düzgün bir şekilde çizilmesi oldukça güç olacaktır. Yöntemler hatalı çizimleri düzeltecek bir yaklaşım içermediği için işlemde hatalı gerçekleşecektir. Bu tez kapsamında ortaya konmakta olan yöntemde, işlem bölgesini seçmek üzere döner tabla ve kamera sistemi kullanılmaktadır. Böylece, işlem bölgesini kamera görüş alanına taşınması mümkün olacaktır. Ayrıca, işlem uygulanacak bölgenin ekran üzerinden işaretlenmesi ve işlem bölgesinin 3 boyutlu nokta bulutu kullanılarak yüzey özelliklerine göre işlem yörüngesi düzeltilmektedir. Ayrıca, operatörün robotun çalışma uzayına çizgi çizmek için girmiyor olması güvenlik açısından da avantajlıdır. Bazı yöntemlerde, nesne üzerine düşürülen matris formunda lazer noktaların ya da profil formunda lazer ışınının kamera ile algılanarak elde edilen yüzey nokta bulutu kullanılarak yol üretme ve izleme yaklaşımı kullanılmaktadır [13, 14]. Bu yöntemlerde, belirli bir nesneye (ayakkabı tabanı) ya da kurguya (eğri yüzey üzerinde kapalı yol üretme) yönelik olarak yöntem sunulmaktadır. Ayrıca, nesnenin işlem uygulanacak bölgesinin lazer ışınların düştüğü yere gelecek şekilde yerleştirilmesi gerekmektedir. Bir başka yöntemde, işlem yapılacak parça üzerine lazer projeksiyonu düşürülmektedir. Operatör lazer kalemi parça üzerindeki işlem yapılacak noktalara tutmakta, kamera ile kalem lazer ışını algılanarak yüzey derinlik haritası oluşturulmaktadır. İlgi çekici bir kullanıcı arayüzü sunan bu çalışmada, işlem noktası seçimi ve robotik işlem gerçekleştirilmesine dair bir yaklaşım sunulmamaktadır. Ayrıca, yöntem ancak, kalem ile yansıtılan lazer ışının kamera tarafından görülebilecek yüzey bölgelerinde çalışabilecek bir yöntemdir. Bu tez çerçevesinde sunulan yöntemdeki gibi, robotik işlem uygulamaya dönük bir yöntem sunmamaktadır.

Çevrimiçi programlama yöntemlerinin daha verimli olması ve uzman operatöre olan bağımlılığın ortadan kaldırılabilmesi için yapılan çalışmalar günümüzde oldukça yoğun olmasına karşılık, önerilen yöntemlerin pek çoğu belli uygulamalar ile sınırlı kalmıştır.

(ii) Çevrimdışı programlama yöntemleri, işlem görecek parçanın ve işlemi gerçekleştirecek olan robotun 3 boyutlu modellerinin benzetim ortamına aktarılarak, gerçekleştirilecek olan işlemin bu ortamda geliştirilmesine olanak sağlamaktadır [16]. Bu yöntemlerin, çevrimiçi programlama yöntemlerine göre pek çok avantajı bulunmaktadır. Bunların başında programlama esnasında gerçek robota ihtiyaç duyulmamasından dolayı, üretim gerçekleştiren robotun, programlama esnasında kullanılmasına izin vermesi gelmektedir. Bir diğer avantaj ise benzetim ortamında gerçekleştirilen programlama yöntemleri, çevrimiçi programlama yöntemlerine göre daha esnek yapıdadır. İşlem görecek parça üzerindeki değişiklikler benzetim ortamına kolaylıkla aktarılıp, geliştirilen program üzerinde değişiklik yapmaya olanak sağlamaktadır. Ayrıca programlama sırasında, karşılaşılma ihtimali bulunan kazalarında önüne geçilmektedir. Çevrimdışı programlama yöntemleri, çevrimiçi yöntemlere göre pek çok avantaja sahip olmasına karşın, küçük ve orta ölçekli endüstriyel işletmelerce tercih edilmemektedir. Bunun başlıca sebepleri ise benzetim ortamlarının elde edilmesi sırasında ortaya çıkan maliyetler, programlama aşamasının uzun sürmesi ve programlama sürecinin yüksek beceri gerektiren bir süreç olması şeklinde sıralanabilir. Her ne kadar robot ve işlem gören parçanın 3 boyutlu modeli benzetim ortamında kullanılsa da ortaya çıkarılan programın gerçek robot ve iş parçası üzerinde test edilmesi ve gerekli düzeltmelerin yapılması gerekmektedir [3]. Bu tez kapsamında ortaya çıkarılan yöntem, çevrimiçi programlama yöntemidir. Bu anlamda, çevrimdışı yöntemlerde olduğu gibi programlama süreci uzun sürmemekte ve yüksek beceri gerektirmemektedir. Ayrıca arttırılmış gerçeklik (Augmented Reality) uygulamalarını da kapsamaktadır. Sanal gerçeklik teknolojilerinin ortaya çıkmasıyla, araştırmacıları bu teknolojilerin robotikte kullanılabileceğini gösteren çalışmalar ortaya koymaya yöneltmiştir. [17]' de gerçekleştirilmiş çalışmada, sanal gerçeklik teknolojileri ve robotiğin birlikte nasıl kullanılabileceğini ve bu teknolojinin robotik alanına nasıl katkı sağlayabileceği açıklanmaya çalışılmıştır. Arttırılmış gerçeklik ise, sanallaştırılmış gerçeklikten türetilmiş bir terimdir. Sanal gerçeklikte, gerçek dünya benzetim ortamına taşınılarak uygulamalarda kullanılırken, arttırılmış gerçeklikte gerçek dünyadan alınan veri (görüntü, ses, vb.) ve bu veriyle ilişkilendirilmiş bilgisayar ortamında oluşturulan bilgi (grafik, metin, vb.) birlikte gerçek zamanlı olarak

sunulmaktadır. Söz konusu teknoloji bilgisayar tarafından oluşturulmuş 3 boyutlu görüntünün gerçek dünyaya uyarlanmasını hedeflemektedir. Bu sayede gerçek dünyada bir robotun yapması planlanan görevlerin, gerçek dünyayı modellemeye ihtiyaç duymadan programlanmasına olanak sağlanmakta ve gerçek dünya ile etkileşimi artırmaktadır [18]. Kullanılan bu yöntem robot programlama alanında devrimsel nitelik taşımakta ve pek çok çalışmanın önünü açmaktadır. [19]'da araştırmacılar, gerçekleştirdikleri çalışmada robot hareket planlamasının arttırılmış gerçeklik kullanılarak nasıl etkili bir şekilde yapılabileceği ve çevrimiçi programlama yöntemlerinde karşılaşılan hareket planlama zorluklarının ve planlama sırasında karşılaşma ihtimali olan tehlike unsurlarının nasıl ortadan kaldırılacağını özetlemektedir. İlgili çalışmada önerilen yöntem ile çevrimdışı programlama ile kazanılan avantajlardan olan fiziksel robot olmadan programlama yapılabilme yeteneği kazanılmaktadır. Bir diğer avantajı ise gerçek robot kullanılarak yapılan programlamanın uygulama güçlüklerini ortadan kaldırmasıdır. Söz konusu çalışmada verilen örneğe göre bir uçağı yıkamakla sorumlu olan bir robotun programlanma sürecinin çevrimiçi yöntemlerle yapılmış olması, oldukça maliyetli ve iş gücü gerektiren bir süreç olacaktır. Fakat arttırılmış gerçeklik kullanılarak, yıkama işleminin gerçekleştirileceği uçağın görüntüsünün üzerine, yıkama işlemini gerçekleştirecek robotun bilgisayar ortamında yaratılmış 3 boyutlu görüntüsünün bindirilmesi ve oluşturulan yeni ortamın kullanılarak programlama işleminin gerçekleştirilmesi oldukça düşük maliyetli ve güvenilir bir yöntem olacaktır. Arttırılmış gerçeklik çerçevesinde sürdürülmüş olan bir başka çalışmada, araştırmacılar çalışma alanında bulunan bir nesneyi yine aynı çalışma alanında farklı konuma taşımakla sorumlu olan bir robot kolun izlemesi gereken yolun planlanması sürecinde, robot program geliştiriciye yardımcı olacak bir benzetim ortamı tasarlamışlardır [20]. Tasarlanan ortamda çalışma ortamının aktif görüntüsünün üzerine bindirilen robotun 3 boyutlu sanal modeli sayesinde robotun başlangıç pozisyonu ve işlevinin bitiş pozisyonu arasında izleyeceği yolu, etrafındaki diğer nesnelere çarpmadan tamamlamasını sağlayacak yolun bulunması kolaylaştırılmıştır. [21]'de gerçekleştirmiş olan çalışmada, robot çalışma ortamında bulunan nesneleri tutmak ile sorumlu bir robotun programlanmasını, insan elinin nesneyi tutma hareketini modelleyerek benzer davranışın robota aktartılmasını hedeflenmişdir. Gerçekleştirilen çalışmada insan elinin küre, silindir, dikdörtgen prizma şeklinde nesneleri farklı şekilde kavradığını ve bu kavrayış şekillerinin 3 boyutlu görüntüleme teknikleri ile görüntülenerek modellenebileceği ortaya koyulmuştur. Ortaya

konulan model yardımı ile nesneleri tutmakla sorumlu bir robot kolun yapması gereken davranışın programlanması öngörülmüştür. [22]'de sunulan bir başka çalışmada, robot çalışması esnasında insan planlamasına ve karar verme sistemine ihtiyaç duyulan durumlarda endüstriyel ortamda kullanılan bir robot kolun kontrolünün, bir başka giyilebilir robot kol ile sağlanması amaçlanmıştır. Söz konusu çalışmada kontrol edilmek istenen robot kol, operatör tarafından giyilen bir başka kolun pozisyon değişimlerine bağımlı olarak hareket etmektedir. Ayrıca operatör tarafından sergilenen davranışların kaydedilmesini veya kontrol edilen robotun davranışının şekillendirilmesi operatörün kontrolünde bulunan bir panel vasıtası ile sağlanmaktadır. Araştırmacılar gerçekleştirmiş oldukları çalışmanın uzaktan kontrol gerektiren tehlikeli faaliyetlerde, su altı bakım ve onarım işlerinde ya da uzay keşif çalışmalarında kullanılabileceğini vurgulamaktadırlar.

Tez çalışması kapsamında ortaya çıkan yöntem ile operatörün robotu programlama esnasında, geliştirilmiş olan arayüz yardımı ile tanımladığı işlem bölgesinin görüntü özellikleri ve bölgeden elde edilen nokta bulutuna göre, kullanıcı girdilerini düzeltme uygulamalarını da kapsamaktadır. İnsan robot etkileşiminin iyileştirilmesi ve insan davranışlarının robotlar tarafından algılanıp, sergilenen davranışa göre yapılacak işlemin belirlenmesi araştırmacıların ilgisini çeken araştırma konularındandır. [23]'de araştırmacılar tarafından gerçekleştirilmiş değerlendirmede, insan robot etkileşimi problemlerine güvenlik ve fiziksel etkileşim çerçevesinde değinilmiş olup, mevcut sorunların üstesinden gelmek adına yapılmış belli başlı çalışmalar özetlenmiştir. Söz konusu değerlendirme bu sorunların üstesinden gelebilmek için, görüntüleme teknolojisi temelli insan robot etkileşimini artıran sistemlerin geliştirilmesi gerektiğine ve bu teknolojilere yönelimin ciddi anlamda artacağına vurgu yapılmıştır. [24]'de gerçekleştirilen çalışmada pek çok farklı cihazdan gelen robot kontrol komutlarının etkin bir kullanıcı arayüz üzerinden kontrol edilmesi hedeflenmektedir. Çalışmada kontrol komut girdi aracı olarak kullanılan el hareketlerinin Kinect vasıtası ile algılanması ve bu hareketlere göre robotun kontrol edilmesi dikkat çekmektedir. [25]'de ortaya koyulan araştırmada, nesneleri taşımakla sorumlu bir robotun endüstriyel ortamda operatör tarafından hızlı bir şekilde programlanmasına olanak sağlanması hedeflenmiştir. Önerilen yöntemde çalışma ortamında bulunan bütün nesnelerin gerçek pozisyonları biliniyor, bu nesnelerden hangisinin alınacağının belirlenmesi gerekiyor ise, operatör tarafından hangi nesnenin alınacağının işaret edilmesinin yeterli olacağı savunulmaktadır. Söz konusu çalışmada gerekli işareti verecek olan operatörün izlenmesi ve sergilemiş olduğu işaret

davranışının tespit edilmesi gerekmektedir. Bir başka önemli arttırılmış gerçeklik tabanlı çalışmada, operatör davranışlarını tanımlayarak robot davranışının programlanması göz önünde bulundurulmuştur [26]. Söz konusu çalışmada araştırmacılar örnek faaliyet olarak tutma/taşıma ve kaynak faaliyetlerini göz önünde bulundurmuşlardır. 3 boyutlu görüntüleme teknolojisi kullanılarak çalışma alanının görüntüsü alınmakta taşınması gereken parçalar operatör tarafından taşınmaktadır. Söz konusu faaliyet robotun hangi parçayı nasıl kavrayacağını ve nereye taşıyacağını öğretmek amacıyla gerçekleştirilmektedir. Bir diğer faaliyet olan kaynak faaliyeti için ise operatörün parmakları ile kaynak yapılacak bölgeyi takip eden bir hareket gerçekleştirip, kaynak yapacak robotun işaret edilen bölgeyi takip edecek şekilde kaynak yapması amaçlanmıştır.

Gerçekleştirilmiş olan bu tez kapsamında, operatörün robotu programlama esnasında kullanacağı bir bilgisayar üzerinde koşan grafik arayüz ve arayüz üzerinden denetlenebilecek bir sistemi kapsamaktadır. El terminalleri, araba navigasyon sistemleri, cep telefonları kullanıcıları kullanımı kolay arayüzlere aşina hale getirmiştir. Söz konusu durum araştırmacıları kullanımı kolay arayüzler ile robot programlamaya yöneltmiştir [27]. Kullanıcıların söz konusu eğilimi robot programlama yöntemlerine aktarılabilmesi durumunda, robot programlamanın var olan yöntemlere göre daha kolay ve güvenilir olacağını öngörülmektedir [28]. Önerilen yöntemde gerçekleştirilen programlama sırasında kullanıcıya yardımcı olacak bir sistem geliştirmek hedeflenmektedir. Bu sistem kullanıcının vermiş olduğu komutlar doğrultusunda ortaya çıkacak davranışı özetleyerek, kullanıcının onayını bekleyecek veya kullanıcının programlama eğilimlerini takip ederek sergilemek istediği davranışları tahmin etmeye çalışacaktır. [29]'da gerçekleştirilen Android tabanlı uygulama, endüstriyel robotların tablet bilgisayarlar ile var olan tanımlı işlemleri belirleyerek programlanmasını amaçlamaktadır. Görsel programlama konseptini temel alan bu yaklaşımda programlamayı gerçekleştirecek operatörün ön tanımlı olarak uygulamada bulunan adımları ve yönergeleri uygulayarak robotu programlaması hedeflenmiştir. Gerçekleştirilen programlama, uygulama içerisinde bulunan benzetim ortamında görüntülenerek test edilmektedir. Ayrıca uygulama işlemi gerçekleştiren robotun görüntülenmesine ve elde edilen sonucun tasarlanan program sonucunda elde edilen arttırılmış gerçeklik ortamı ile kıyaslanmasına izin vermektedir. Ortaya koyulan bir başka robot programlama çalışmasında 3 boyutlu arayüzler kullanılarak, seramik endüstrisinde bir endüstriyel robotun programlanması

gerçekleştirilmiştir [30]. Kullanılan robotun gerçekleştirdiği faaliyetin sıklıkla değişmesi, kolay ve hızlı programlama gereksinimini ortaya çıkarmıştır. Araştırmacıların geliştirdikleri sistem iki adımdan oluşmaktadır. Söz konusu adımlar operatör tarafından robotun izleyeceği yolun arttırılmış gerçeklik ortamında çizilmesi ve bu çizimin robotun izleyeceği 3 boyutlu işlem yoluna çevrilmesidir. Gerçekleştirilen bir başka çalışmada, robot programlama arayüzlerinin basitleştirilmesi ile robot programlama sürecinde elde edilebilecek iyileştirmeler test edilmiştir. İlgili çalışma kapsamında tasarlanan arayüz, geleneksel programlama yöntemlerine göre, kullanıcıların program geliştirme sürelerinde ciddi azalmalar olduğu ve kullanıcıların robotun yapmasını istedikleri süreci daha iyi kavrayıp şekillendirebildikleri programlar ortaya koyduğu gözlemlenmiştir [31]. [32]'de sunulan çalışma ise benzer şekilde küçük ve orta ölçekli endüstriyel işletmelerin karşılaşmış olduğu robot programlama yapacak personel istihdam edememe problemine çözüm üretmek adına kuvvet algılayıcısı ve ses kontrolü yardımı ile robot programlama yapılabileceğini göstermektedir. Kullanılan ses kontrol arayüzü yardımı ile robotun programlanmasının sesli komutlar ile yapılabileceği ve geliştirilen sistem üzerinde ön tanımlı olarak bulunan komutlar yardımı ile görsel olarak robot programlama yapılabileceği vurgulanmaktadır. Hibrit bir programlama yönteminin önerildiği bir başka çalışmada [33] robot hareket planının robot çalışma alanına yansıtılmış olan lazer yansıtıcı vasıtası ile oluşturulması ve robotun görüntü işleme teknikleri kullanılarak yansıma ile oluşturulmuş hareket planını takip etmesi amaçlanmıştır. Bununla birlikte bir tablet bilgisayar kullanılarak, robotun hangi noktadan yansıtılan hareket plan yoluna gireceği, taşıma ve hareket parametreleri (hız, pozisyon vs.) belirtilebilmektedir.

1.3. Tez Organizasyonu

Tez çalışması kapsamında temel olarak kolay kullanıcı arayüzü destekli, uzman operatör ihtiyacını mümkün olduğunca azaltmayı hedefleyen bir sistem yaratılması amaçlanmıştır. Ortaya çıkan sistemin, endüstriyel robot kullanımının, programlama maliyetlerini azaltarak, robot kullanımından uzak duran küçük ve orta ölçekli, ürün çeşitliliğinin fazla olduğu ve sıklıkla yeniden programlamaya ihtiyaç duyulan işletmelerde de kullanımının önünün açılabilmesini sağlaması öngörülmektedir. Bu bağlamda ortaya koyulan sistemin mümkün olduğunca geleneksel yöntemlerde ihtiyaç duyulan robot hakkında deneyimli olan bir personele gereksinimini ortadan kaldırması

gerekmektedir. Bununla birlikte sistemin kolay öğrenilebilir olması gerekmektedir. Ortaya koyulan tez çalışmasında operatörün, sunulan kullanıcı arayüzünü kullanarak aşağıdaki adımları gerçekleştirmesi gerekmektedir:

- İlk olarak, programlama işleminin yapılacağı endüstriyel robot hücresinde bulunan kamera, döner tabla ve endüstriyel robot kolun, birbirleri arasındaki ilişkilerin (transformasyon) tespit edilmesi gerekmektedir. Bu süreç robot hücresinin ilk kurulumu sırasında yapılması gerekmektedir. Farklı endüstriyel parçalar için gerçekleştirilen programlama süreçlerinde, bu ekipmanların pozisyonlarının ve/veya yönelimlerinin değiştirilmediği takdirde tekrarlanmasına gerek yoktur. Sunulan tez kapsamında bu sürece *kalibrasyon süreci* olarak değinilecektir.
- Kalibrasyon süreci tamamlanan bir endüstriyel robot hücresi için gerçekleştirilecek olan programlama sürecinin ilk adımı, endüstriyel parçanın işlenecek bölgesinin, döner tablanın döndürülmesi ile yine aynı bölgeyi görmekte olan kameranın görüş alanına sokulması ile başlamaktadır.
- Kameranın görüş alanına giren işlem görecek bölge, operatör tarafından, kullanıcı arayüzü vasıtası ile belirlenir ve bu bölgenin döner tabla koordinat sisteminde, dolaylı olarak robot ana koordinat sisteminde nerede bulunduğu tespit edilir. Seçimi yapılan bu bölgeye ilerleyen bölümlerde *ilgilenilen bölge* olarak değinilecektir.
- Operatör tarafından kullanıcı arayüzü kullanılarak, işlem yapılacak bölgenin görüntüsü üzerinde belirlenen ilgilenen bölgenin döner tabla koordinat sistemine göre konumunun tespit edilmesi, bu bölgenin endüstriyel robota bağlı bulunan lazer profil tarayıcı vasıtasıyla taranarak üç boyutlu nokta bulutunun elde edilmesi sağlanır.
- Elde edilen nokta bulutu analiz edilerek gerçekleştirilecek bölgenin geometrik özellikleri tespit edilir ve robot hareket planı oluşturulur. Bu tez çalışması kapsamında kaynak yapacak olan bir robotun izleyeceği yörüngenin belirlenmesi ele alınmıştır.

Tezin bölümleri ise şu şekilde özetlenebilir:

Bölüm 1'de genel olarak, tezin araştırma konusuna, bu konu kapsamında hali hazırda var olan problemin belirlenmesine, bu tez çalışmasının bu probleme nasıl çözüm aradığına ve ortaya çıkarılması hedeflenen sistemin amaç ve hedeflerine değinilmiştir. İlgili alana ait literatür araştırmasıyla robot programlama yöntemlerinin iyileştirilmesine yönelik sunulan yöntemlere değinilmektedir. Bu yöntemlerin avantajları ve dezavantajları ortaya konmaktadır. Yürütülen tez çalışmasının literatürde nerede konumlandığı tarif edilmektedir.

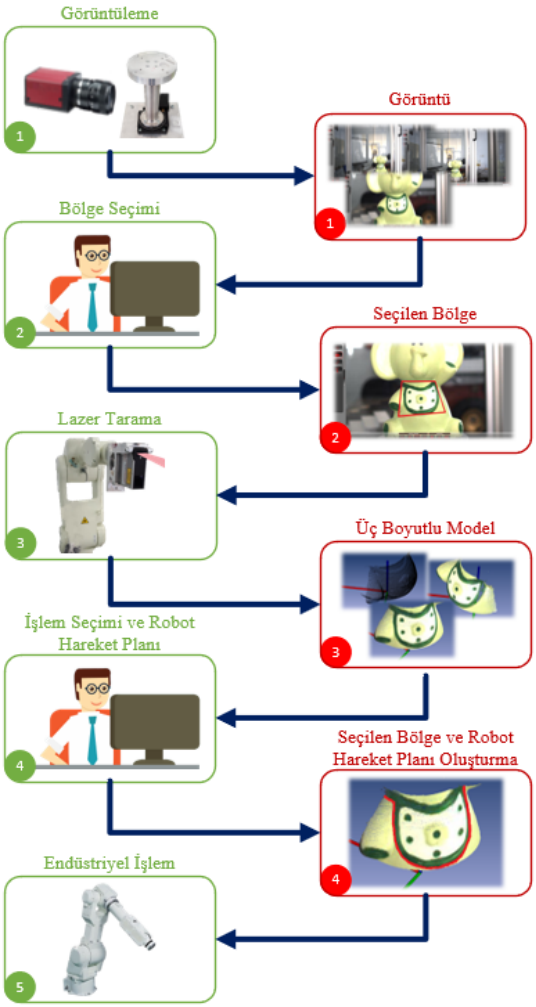
Bölüm 2'de, tez çalışması kapsamında yürütülen çalışmada uygulanan yöntemler bu yöntemlerin detayları ve yöntemin uygulanması sonucu elde edilmesi hedeflenen sonuçlar sunulmuştur. Sunulan her bir yönteme ait matematiksel arka plan ve/veya sahte kodlar ilgili alt bölümlerde sunulmaktadır. Ayrıca her alt bölüm için uygulanan yöntemler ile ortaya çıkartılan sonuç görsellerle desteklenmektedir. Bu bölümde tez çalışması kapsamında ortaya koyulan çalışmada uygulanan yöntemler hakkında okuyucunun yeterli arka plana sahip olması hedeflenmiştir.

Bölüm 3'te, ortaya çıkarılan sistem ile gerçekleştirilmiş olan uygulama detayları, bu uygulamaya ait sonuçlar ve ilgili yorumlar sunulmaktadır. Bölümün girişinde deneysel çalışmaların kapsamı ve öngörülen sonuçlar hakkında okuyucu bilgilendirilmektedir. Deneysel çalışmaların yürütüldüğü laboratuvar ortamında kullanılan donanımın teknik detayları, geliştirilen sistemin detaylı yazılım tasarımı ve alt yapısı hakkında detaylı açıklamalar da bu bölümde sunulmaktadır.

Bölüm 4'te, tez çalışması özetlenmiştir. Ayrıca bu tez ile geliştirilen sistemin hedeflerine ve hedeflerin gerçekleşmesi ile ortaya çıkacak olası sonuçlara değinilmektedir.

2. YÖNTEM

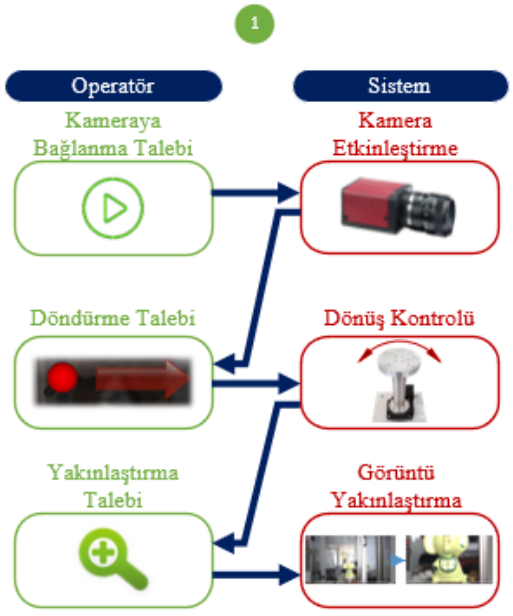
Kalibrasyon süreci tamamlanmış bir endüstriyel robot hücresinde bulunan bir endüstriyel robotun programlanma süreci sırasıyla takip edilmesi gereken belli ana adımlardan oluşmaktadır. Her bir adımda elde edilen çıktı(lar), bir sonraki süreç için girdi(leri) oluşturmaktadır. Bu adımlar Şekil 2.1 de özetlenmektedir. Gerçekleştirilen bütün adımlar geliştirilmiş olan arayüz kullanılarak yürütülmekte olup, programlama sürecini gerçekleştiren operatörün, teknik bilgi ve beceresine ihtiyaç duyulmamaktadır.



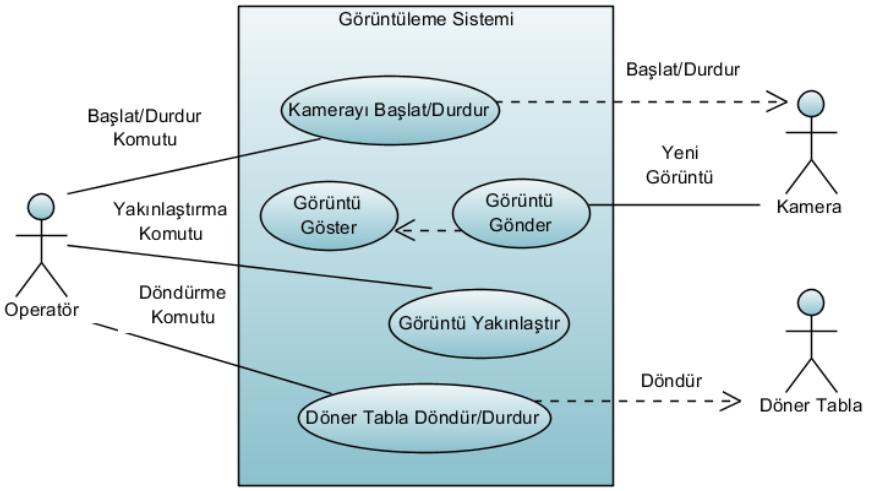
Şekil 2.1. *Robot Programlama Adımları*

İzlenmesi gereken ana adımları kendi içerisinde operatör ve sistem etkileşimini gösterecek şekilde detaylandırmak mümkündür. Bir numaralı adım, döner tabla üzerinde konumlandırılmış işlem gerçekleştirilecek endüstriyel parçanın, robot hücresinde

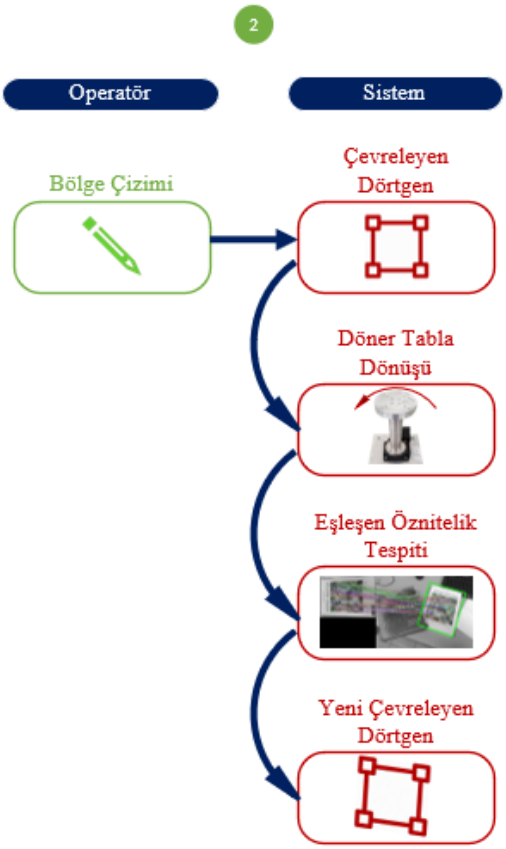
bulunan kamera yardımı ile uygun şekilde görüntülenebilir pozisyona getirilmesi ve işlem gerçekleştirilecek bölgenin operatör tarafından kolay görünebilir hale getirilmesi adımını temsil etmektedir (Bkz. Şekil 2.2, 2.3). İki numaralı adım, kameranın görüş alanına sokulmuş işlem görecek bölgenin, operatör tarafından belirlenmesini ve bu bölgenin üç boyutlu uzaydaki pozisyonunun bulunabilmesi için eşdeğer görüntünün elde edilmesi sürecini özetlemektedir (Bkz. Şekil 2.4, 2.5). Üç numaralı adım, kamera görüntüsü üzerinde belirlenen işlem görecek bölgenin üç boyutlu uzaydaki karşılığının belirlenmesi ve buna bağımlı olarak lazer tarama işlemi için robot hareket planının elde edilmesine karşılık gelmektedir (Bkz. Şekil 2.6, 2.7). Dört numaralı adımda ise lazer tarama ile elde edilen nokta bulutu kullanılarak gerçekleştirilecek endüstriyel işlemin belirlenmesi ve bu belirlen bu işlem için robot hareket planının oluşturulup, elde edilen hareket planının robota verilmesi sürecini göstermektedir (Bkz. Şekil 2.8, 2.9). Son adım ise ortaya çıkarılmış olan hareket planının endüstriyel robot tarafından uygulanması adımıdır. Bu adımlara ait detaylı süreç ve kullanım senaryosu diyagramları devam eden şekillerde sunulmakta olup, izleyen alt bölümlerde bu tez kapsamında sunulan yöntemin adımlarının detaylarına ve bu alt adımlarda kullanılan yöntemlerin teknik ayrıntılarına değinilmektedir.



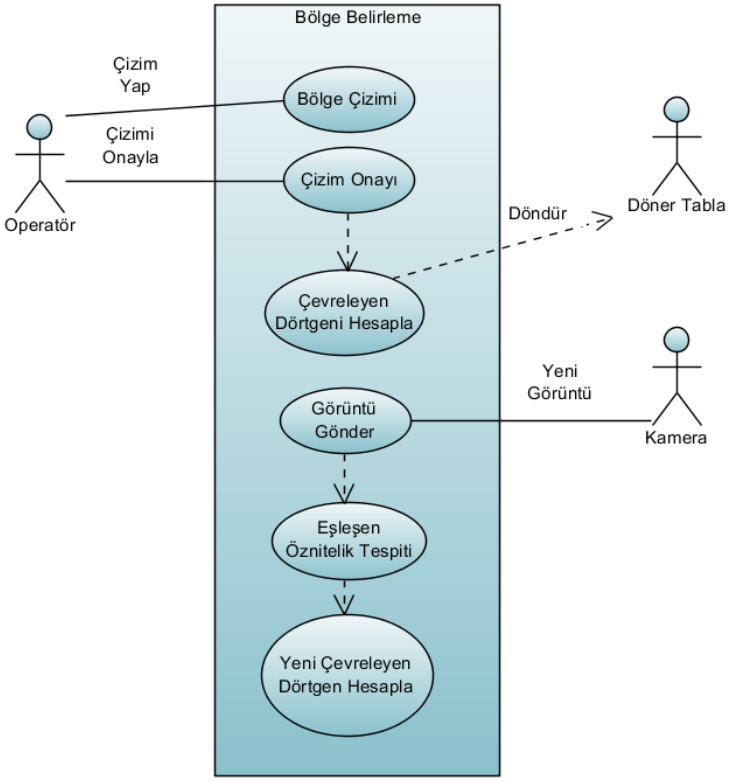
Şekil 2.2. *Robot Programlama Süreci 1. Adım Süreç Diyagramı*



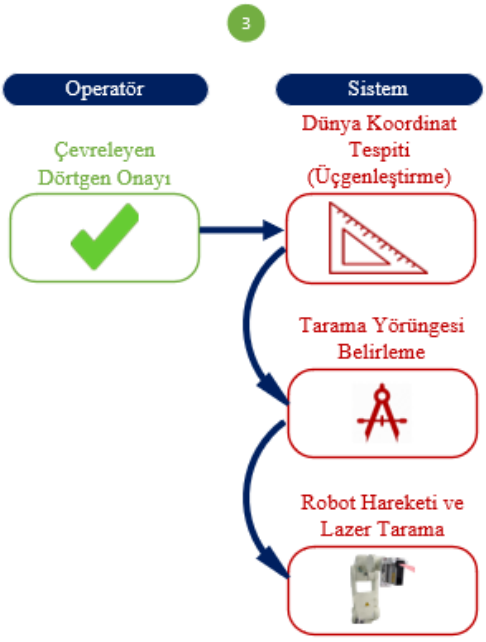
Şekil 2.3. *Robot Programlama Süreci 1. Adım Kullanım Senaryosu Diyagramı*



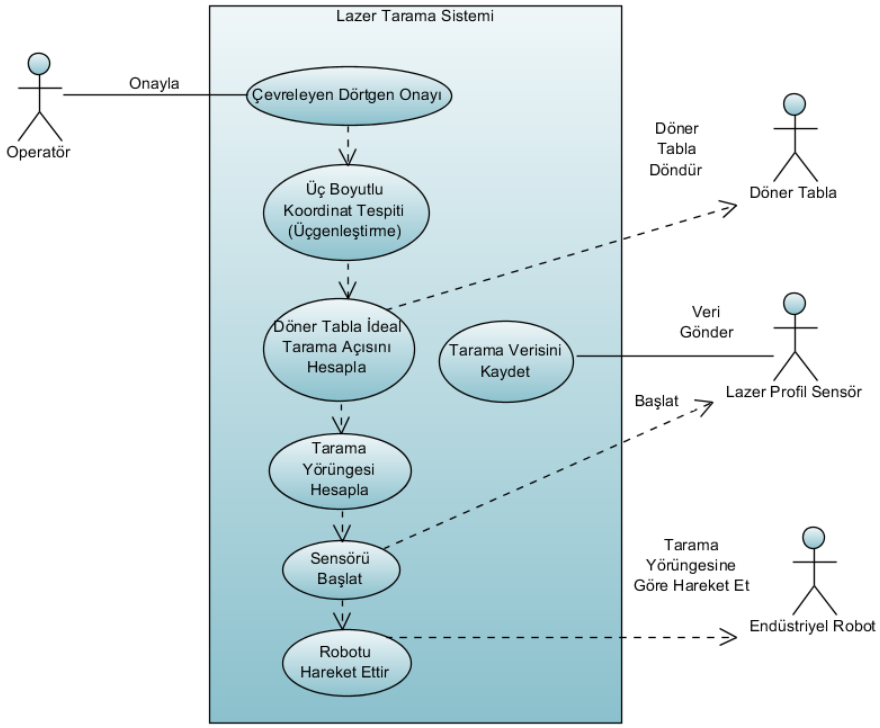
Şekil 2.4. *Robot Programlama Süreci 2. Adım Süreç Diyagramı*



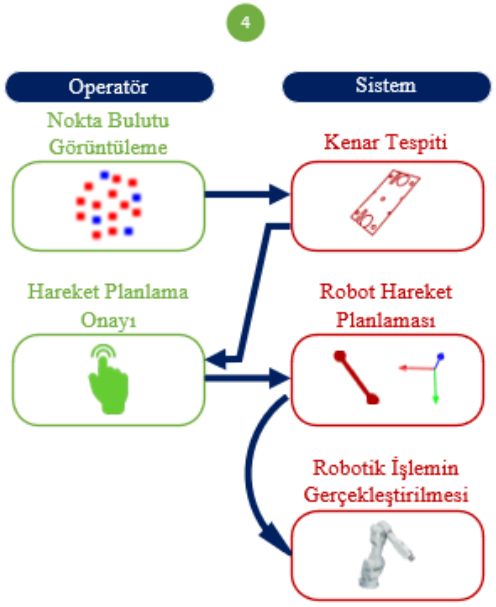
Şekil 2.5. *Robot Programlama Süreci 2. Adım Kullanım Senaryosu Diyagramı*



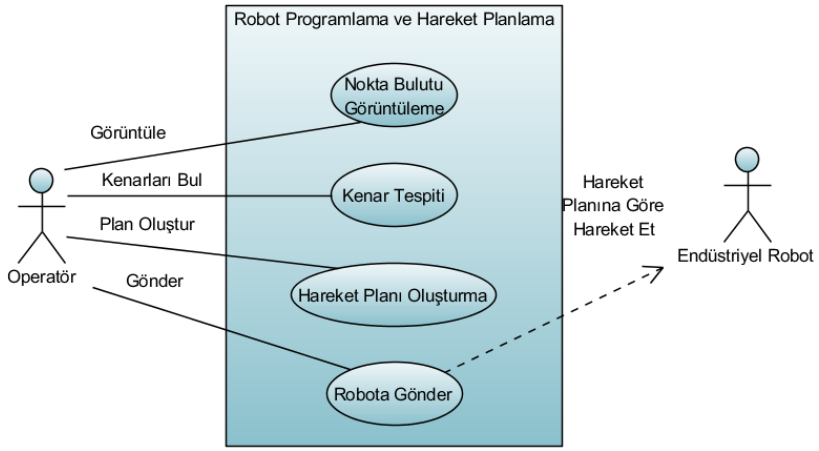
Şekil 2.6. *Robot Programlama Süreci 3. Adım Süreç Diyagramı*



Şekil 2.7. *Robot Programlama Süreci 3. Adım Kullanım Senaryosu Diyagramı*



Şekil 2.8. *Robot Programlama Süreci 4. Adım Süreç Diyagramı*



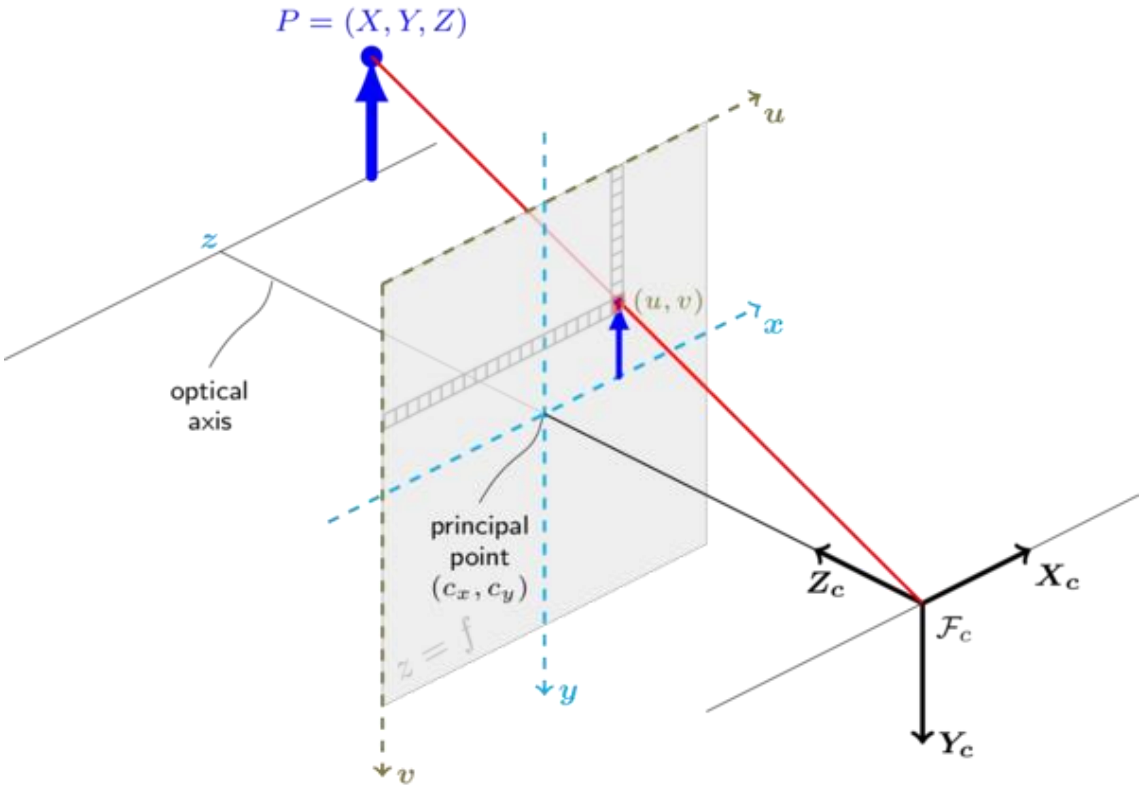
Şekil 2.9. *Robot Programlama Süreci 4. Adım Kullanım Senaryosu Diyagramı*

2.1. Kalibrasyon Süreci

Tez çalışması kapsamında geliştirilen yöntemin başarıya ulaşabilmesi için, programlama süreci gerçekleştirilecek endüstriyel robot hücresi içerisinde bulunan ana donanımlar (kamera, döner tabla, endüstriyel robot kol) arasındaki konum ilişkilerinin tespit edilmesi gerekmektedir. Bu ilişkiler dönüşüm matrisleri ile temsil edilmektedir. Tespit edilmesi gereken dönüşüm matrisleri kamera ve döner tabla kalibrasyon süreçlerinin işletilmesi ile elde edilmektedir. Bu süreçlerin mümkün olan en yüksek doğrulukta yürütülmesi, programlama sürecinin sağlıklı ilerlemesi adına oldukça önemlidir.

2.1.1. Kamera Kalibrasyonu

Herhangi bir görüntü yakalama sistemi (genellikle kamera), sağlıklı bir kalibrasyon sürecinin işletilmesine ihtiyaç duymaktadır. Bu süreç kameranın içsel (odak uzaklığı, açısal eksen bozulması vb.) ve dışsal (oryantasyon ve öteleme matrisleri) parametrelerinin elde edilmesini sağlamaktadır. Elde edilen bu parametreler, dünya uzayındaki herhangi bir üç boyutlu noktanın (P), koordinat sistemi merkezi (odak merkezi) $\mathcal{F}_c$ olan kameraya ait görüntü koordinat sisteminde $(\vec{u}, \vec{v})$ hangi noktaya (u, v) karşılık geleceğini belirleyen dönüşüm matrisinin elde edilmesini sağlamaktadır (Bkz. Şekil 2.10).

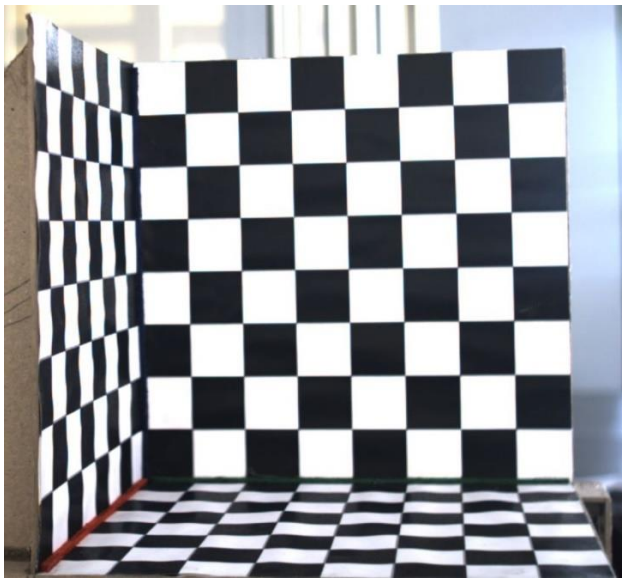


Şekil 2.10. *İğne Deliği Kamera Modeli [34]*

Pek çok farklı kamera kalibrasyon yöntemi bulunmaktadır. Bunların başlıca olanları şöyle sıralanabilir:

- *Otomatik kalibrasyon yöntemleri*: Bu yöntemler genellikle kameraya ulaşma imkanının bulunmadığı, ya da daha önce elde edilmiş (video görüntüsü vb.) görüntülerde tespit edilen özellikler kullanılarak kalibrasyon sürecinin gerçekleştirilmesi gereken uygulamalarda tercih edilmektedir.
- *Düzlemsel kalibrasyon deseni kullanılarak gerçekleştirilen kalibrasyon yöntemi*: Oldukça yaygın olarak kullanılan bu yöntemde, düzlemsel ve üzerinde aralarındaki uzaklıkları bilinen ve görüntü üzerinde tespit edilmesine olanak sağlayan bir desen barındıran (satranç tahtası deseni gibi) bir nesnenin farklı bakış açılarından birden fazla görüntüsünün alınarak, kamera parametreleri elde edilmektedir [35]. Kalibrasyon deseninin kolay elde edilebilmesi sebebi ile sıklıkla tercih edilmektedir. Genellikle kamera içsel parametrelerinin elde edilmesinde tercih edilen bu yöntem, dışsal parametrelerin (döndürme ve öteleme matrisleri), kameranın görüş alanında bulunan belli bir koordinat sistemine göre elde edilmesini oldukça zor hale getirmektedir.

- *Birbirine dik iki ya da üç düzlem kullanılarak gerçekleştirilen kalibrasyon yöntemi*: Bu yöntemde, düzlemsel kalibrasyon deseni kullanılarak gerçekleştirilen kalibrasyon yönteminde kullanılan nesnelerin (2 ya da 3) birbirine dik olacak şekilde birleştirilmesi ile elde edilen bir kalibrasyon nesnesi kullanılmaktadır (Bkz. Görsel 2.1). Kamera kalibrasyon yöntemleri arasında oldukça hassas sonuç elde edilebilmesi ve dünya koordinat sisteminin merkez koordinat sisteminin, kolaylıkla tanımlanabilmesi sebebi ile bu yöntem tez çalışmasında tercih edilmiştir. Kamera dönüşüm matrisinin dünya koordinat sisteminde bulunan belli bir eksene göre (döner tabla koordinat ekseni) elde edilmesi gereksinimi, bu yöntemin tercih edilmesinde ana etkendir.



Görsel 2.1. *Birbirine Dik Üç Eksenden Oluşan Kalibrasyon Nesnesi*

Kamera kalibrasyonunun gerçekleştirilmesi şu adımları içermektedir:

- Kalibrasyon nesnesinin uygun pozisyona yerleştirilmesi
- Desen köşelerinin tespit edilmesi ve uygun kalibrasyon noktalarının kullanıcı tarafından belirlenmesi
- Kamera dönüşüm matrisinin seçilen noktalar kullanılarak hesaplanması, içsel ve dışsal parametrelerin, dönüşüm matrisinden elde edilmesi

2.1.1.1. *Kalibrasyon Nesnesinin Uygun Pozisyona Yerleştirilmesi*

Gerçekleştirilen kamera kalibrasyon sürecinin asıl amacının bir önceki bölümde de belirtildiği gibi, dünya uzayında belirlenmiş olan bir koordinat sisteminin kamera dönüşüm matrisi üzerinden, kamera görüntü koordinat sistemine aktarılmasını sağlayacak dönüşüm matrisinin elde edilmesidir. Geliştirilen sistem için, kalibrasyon nesnesinin uygun görülen dünya koordinat ekseni ile bir şekilde bağlantılı (birebir ya da aralarındaki dönüşüm bilinecek şekilde) olacak şekilde konumlandırılması ve kalibrasyon sürecinin gerçekleştirilmesi gerekmektedir. Tez çalışması kapsamında seçilen koordinat ekseni döner tabla koordinat ekseni olarak belirlenmiştir. Bu eksen ile kalibrasyon nesnesine ait koordinat ekseni arasındaki bağlantı geometrik olarak bilinmekte, sürecin devamında gerçekleştirilecek hesaplamalarda göz önünde bulundurulacaktır.

2.1.1.2. *Desen Köşelerinin Tespit Edilmesi ve Uygun Kalibrasyon Noktalarının Kullanıcı Tarafından Belirlenmesi*

Uygun pozisyona yerleştirilmiş olan kalibrasyon nesnesi kullanılarak kalibrasyon sürecinin gerçekleştirilebilmesi, kamera tarafından görüntülenen bu nesnenin üzerinde bulunan köşe noktalarının (kalibrasyon nesnesi koordinat sistemine göre, üç boyutlu noktalar), görüntü koordinat sisteminde tanımlanan eş noktaların (kamera görüntüsü üzerindeki iki boyutlu pikseller) belirlenmesi gerekmektedir. Bu süreçte seçilecek noktalar, dünya uzayında nerde olduğu kolaylıkla görülebilen desendeki karelerin köşe noktalarıdır. Bu noktaların kamera görüntü uzayındaki konumları herhangi bir köşe bulma yöntemi kullanılarak tespit edilebilir. Harris [36] ve Shi-Tomasi [37] köşe bulma algoritmaları bu işlem için oldukça uygun yöntemlerdir.

Harris köşe bulma algoritması temel olarak görüntü düzlemi üzerinde belirlenen lokal pencerelerle sınırlı kalacak şekilde görüntüdeki piksel bölgelerinin parlaklık değişimlerinin büyüklüğünü değerlendirmektedir. Bu lokal pencerenin (w) görüntü düzleminin x ekseni ve y ekseni boyunca gerçekleştirilen küçük kaydırma işlemleri nedeni ile ortaya çıkan parlaklık değişimi şu şekilde hesaplanmaktadır:

$$E_{x,y} = \sum_{u,v} w(u,v)\,[I(x+u, y+v) - I(u,v)]^2 \tag{2.1}$$

Burada; $w(u,v)$ lokal pencereyi, $I(u,v)$ parlaklık seviyesini temsil etmektedir.

Ayrıca $I(u,v)$, (u,v) noktasındaki parlaklık seviyesini göstermektedir.

Gerçekleştirilen hesaplama işlemi sonucunda, görüntü üzerinde bulunan köşe noktalarında parlaklık değişimleri, köşe noktası olmayan piksellere oldukça yüksek çıkacaktır. Lokal penceredeki kaydırma miktarının oldukça küçük olduğu göz önünde bulundurulduğunda (2.1)'de verilen denklem şu şekilde yeniden düzenlenebilir.

$$\begin{aligned} E_{x,y} &= \sum_{u,v} w(u,v)\,[I(u,v) + xI_u + yI_v - I(u,v)]^2 \\ &= \sum_{u,v} w(u,v)\,[x^2 I_u^2 + 2xyI_uI_v + y^2 I_v^2] \end{aligned} \tag{2.2}$$

Burada; I_u ve I_v, (u,v) konumlarındaki sırası ile x ve y eksenleri boyunca birinci dereceden türeve karşılık gelmektedir. Ayrıca bu denklem matris formunda şu şekilde de ifade edilebilir.

$$E_{x,y} = [x \quad y]\left(\sum_{u,v} w(u,v)\begin{bmatrix} I_u^2 & I_uI_v \\ I_uI_v & I_v^2 \end{bmatrix}\right)\begin{bmatrix} x \\ y \end{bmatrix} \tag{2.3}$$

Buna ek olarak ortada bulunan terim M adında $2x2$'lik bir başka matris ile ifade edilirse denklem (2.3) şu hale dönüşür.

$$E_{x,y} = [x \quad y]M\begin{bmatrix} x \\ y \end{bmatrix} \tag{2.4}$$

Denklem (2.4)'de göründüğü üzere, parlaklık değişimi doğrudan matris M'e bağımlı haldedir. Bu nedenle parlaklık değişiminin değerlendirilmesi için bu matrisin öz değerlerine (λ_1 ve λ_2) bağımlı olan bir değerlendirme yöntemi geliştirilmesi uygun olacaktır.

$$R = Det(M) - k\,Trace(M)^2 \tag{2.5}$$

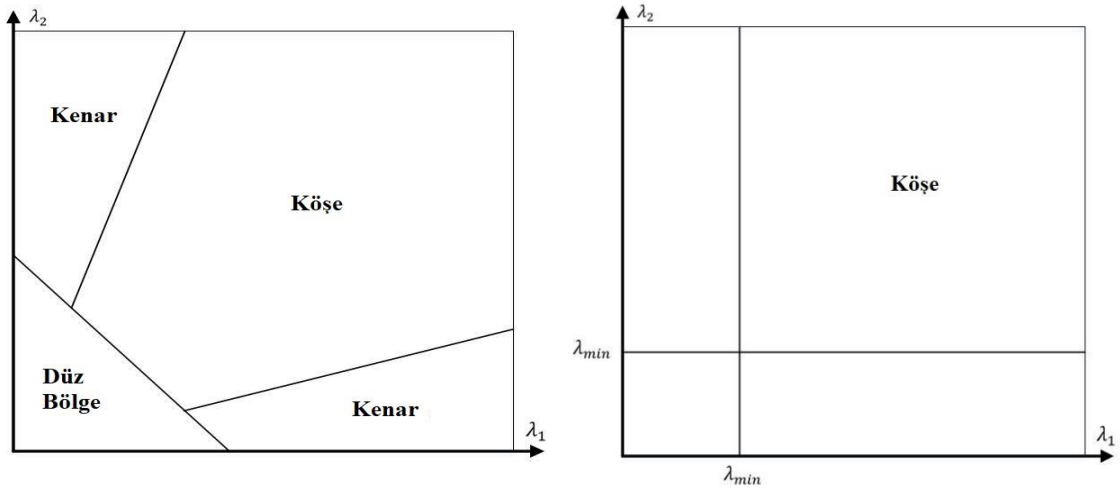
Burada; $Det(M) = \lambda_1\lambda_2$ ve $Trace(M) = \lambda_1 + \lambda_2$ şeklinde hesaplanmaktadır ve hesaplanan R değeri için üç farklı durum ortaya çıkmaktadır. Ortaya çıkan olası durumlar hesaplama için kullanılan pencerenin merkezinde konumlanmış bir pikselin hangi sınıfta değerlendirileceğini belirlemektedir (Bkz. Şekil 2.11).

- Eğer lokal pencere içerisindeki alan düz ise, λ_1 ve λ_2 küçük olacaktır, dolayısıyla R değeri de küçük çıkacaktır.
- Eğer lokal pencere içerisindeki alan görüntü üzerinde bir kenara karşılık geliyor ise, öz değerlerden birisi, diğerinden oldukça büyük olacaktır ve R değeri

negatif olacaktır.

- Eğer lokal pencere içerisindeki alan görüntü üzerindeki bir köşeye karşılık geliyor ise, öz değerlerin ikisi birden büyük değerlere sahip olacak ve R değerinin büyük olmasına sebep olacaktır.

Shi-Tomasi yönteminde ise deformasyon ve öteleme modellerinin birleştirilerek elde edilen yer değiştirme modelini kullanıp, birbirini izleyen az miktarda değişim barındıran görüntü serisi için hangi piksellerin takip edilmeye uygun olduğunu tespit etmeyi amaçlamaktadır [37]. Bu yaklaşımda takip edilebilecek piksellerin genellikle köşe noktalar olması nedeni ile, aynı yöntem köşe bulma algoritması olarak kullanılmaya uygundur. Bu yöntemde başvurulan değerlendirme fonksiyonu $R = \min(\lambda_1, \lambda_2)$ olarak tanımlanmakta olup, eğer elde edilen değerlendirme puanı (R) verilen en düşük değerlendirme puanından (λ_{min}) büyük ise, ilgili pencere köşe olarak kabul edilmektedir.

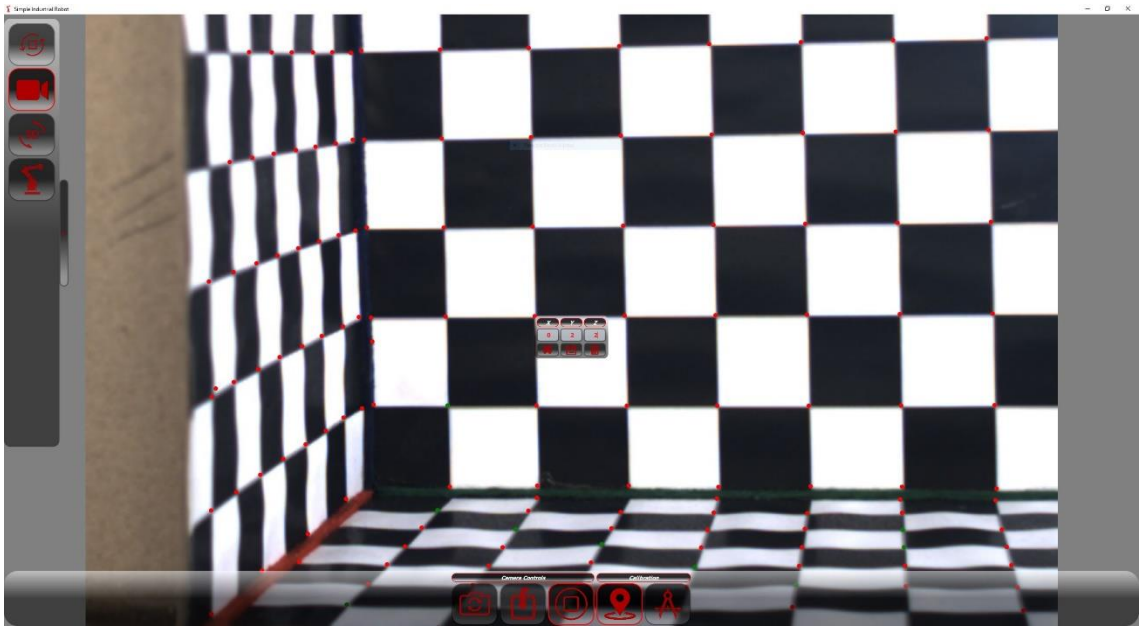


(a) (b)

Şekil 2.11. *Harris (a) ve Shi-Thomasi (b) Köşe Bulma Sınıflandırma Grafiği*

Köşelerin otomatik olarak sistem tarafından tespit edilmesinin ardından kullanıcıya (operatör) bu köşelerin pozisyonları, kalibrasyonu yapılacak kameradan gelen görüntüleri anlık olarak görüntüleyen kullanıcı arayüzü üzerinden sunulur. Kullanıcının bu noktalar arasından uygun görünenler için (20 adet ve üzeri tavsiye edilmektedir), kalibrasyon nesnesine ait koordinat sistemindeki karşılıklarını ve her bir kare arasındaki (bütün kareler için aynı) mesafeyi mm cinsinden belirtmesi gerekmektedir. Bu bilgiler verildiğinde, kamera görüntü düzlemi koordinat sistemindeki noktalar (pikseller) ve bu

noktaların, kalibrasyon nesnesi koordinat eksenindeki karşılıkları elde edilmiş olacaktır. Eşleşen bu nokta çiftleri, kamera kalibrasyon sürecinde elde edilmesi beklenen dönüşüm matrisinin oluşması için yeterlidir. Görsel 2.2 tarif edilen sürecin, geliştirilen arayüz üzerinden nasıl yürütüldüğünü sergilemektedir.



Görsel 2.2. *Kalibrasyon Noktalarının Tespiti ve Gerçek Konumlarının Verilmesi*

2.1.1.3. *Kamera Dönüşüm Matrisinin Elde Edilmesi*

Kamera dönüşüm matrisinin, hesaplanabilmesi için, sistem tarafından otomatik olarak tespit edilen köşe noktaları arasından, kullanıcı tarafından uygun görülen belli sayıda (20 ve üzeri) noktanın, kalibrasyon nesnesi koordinat sistemindeki karşılıklarının belirlenmiş olması gerekmektedir. Elde edilen bu noktalar kamera görüntü düzlemine ait koordinat ekseni ile, kalibrasyon nesnesi arasındaki ilişkinin tespit edilmesi için kullanılacaktır. Elde edilen bu dönüşüm matrisi kalibrasyon nesnesine ait koordinat sisteminde tanımlı bir noktayı, görüntü düzlemindeki iki boyutlu bir noktaya dönüştürmek için kullanılmaktadır. Bu matris **M** ile tanımlanabilir ve temel olarak kamera içsel parametrelerini barındıran kamera matrisi (**K**), kameranın kalibrasyon koordinat sistemine göre oryantasyonuna ait bilgiyi taşıyan döndürme matrisi (**R**) ve kameranın yine aynı koordinat sistemine göre pozisyonuna ait bilgiyi içeren pozisyon vektörü (**t**) ile doğrudan bağlantılıdır. Bu bağlantı şu şekilde tanımlanmaktadır.

$$\mathbf{M} = \mathbf{K}\,[\,\mathbf{R}\quad \mathbf{t}\,] \tag{2.6}$$

Kamera matrisi, sistemde kullanılmakta olan kamera ait, özellikleri barındırmaktadır. Bu özellikler; görüntü düzleminin merkezinde konumlanan odak noktası (δ_x, δ_y), odak uzaklığı (φ_x, φ_y) ve görüntü düzlemi ana eksenleri arasındaki açının kosinüsüne karşılık gelen eğiklik (γ) parametreleridir.

$$\mathbf{K} = \begin{bmatrix} \varphi_x & \gamma & \delta_x \\ 0 & \varphi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

Kalibrasyon koordinat sisteminde tanımlı üç boyutlu bir noktanın (**w**), kamera görüntü düzlemindeki karşılığı olan iki boyutlu bir noktaya (**x**) dönüştürülme işlemi şu şekilde tanımlanmaktadır.

$$\lambda \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \mathbf{M} \begin{bmatrix} \mathbf{w} \\ \mathbf{1} \end{bmatrix} \tag{2.8}$$

Burada; λ herhangi bir sayısal değere karşılık gelmektedir. Bu değer elde edilen homojen koordinat sisteminde tanımlı noktanın son elemanını birim değere sabitlemek için kullanılmaktadır.

Denklem (2.8), kullanıcı tarafından belirlenen kalibrasyon nokta çiftlerinden herhangi biri için açılacak olursa:

$$\lambda_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix} \tag{2.9}$$

Burada; m_{ij} dönüşüm matrisi **M**'nin i'inci satırı ve j'inci sütununda bulunan elemana, x_i ve y_i, dönüşüm sırasında elde edilen görüntü koordinat sistemindeki noktanın x ve y eksenlerindeki değerine, u_i, v_i ve w_i ise dönüşümü yapılacak üç boyutlu noktanın sırası ile x, y, ve z eksenlerindeki koordinatlarına karşılık gelmektedir.

Denklem (2.9)'da verilen ilişki kullanılarak, kullanıcı tarafından sağlanan kalibrasyon noktaları için aşağıdaki lineer sistem oluşturulabilir.

$$\begin{bmatrix} \mathbf{w_1^T} & \mathbf{0^T} & -x_1\mathbf{w_1^T} \\ \mathbf{0^T} & \mathbf{w_1^T} & -y_1\mathbf{w_1^T} \\ \mathbf{w_2^T} & \mathbf{0^T} & -x_2\mathbf{w_2^T} \\ \mathbf{0^T} & \mathbf{w_2^T} & -y_2\mathbf{w_2^T} \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{22} \\ \cdots \\ \cdots \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \cdots \\ \cdots \end{bmatrix} \tag{2.10}$$

Burada; $\mathbf{w_i^T}$ homojen kalibrasyon nesnesi koordinat sisteminde tanımlanan bir noktaya, $\mathbf{0^T}$ 4 boyutlu sıfır vektörüne karşılık gelmektedir.

Oluşturulan sistemin sol tarafını P matrisi olarak isimlendirilirse ($\mathbf{Pm = 0}$) bu lineer sistemin çözümü kamera dönüşüm matrisini verecektir. Bu sistemin çözümü $\mathbf{P^TP}$ matrisinin en küçük öz değerine karşılık gelen vektördür. Bu vektörün elemanları aynı zamanda **M** matrisinin elemanlarıdır. Denklem (2.6) kullanılarak aşağıdaki eşitlik yazılabilir.

$$\mathbf{M} = [\mathbf{KR} \quad \mathbf{Kt}] \tag{2.11}$$

Eğer, $3x3$'lük **KR** matrisini **B** ve $3x1$'lik **Kt** vektörünü **b** olarak tanımlarsak ve **B** matrisini sağ taraftan $\mathbf{B^T}$ ile çarparsak:

$$\mathbf{BB^T} = \mathbf{KRR^TK^T} \tag{2.12}$$

R bir döndürme matrisi olması sebebi ile $\mathbf{RR^T} = \mathbf{I}$

$$\mathbf{BB^T} = \mathbf{KK^T} \tag{2.13}$$

Denklem (2.7) kullanılarak,

$$\mathbf{KK^T} = \begin{bmatrix} \varphi_x^2 + \gamma^2 + \delta_x^2 & \gamma\varphi_y + \delta_x\delta_y & \delta_x \\ \varphi_y\gamma + \delta_x\delta_y & \varphi_y^2 + \delta_y^2 & \delta_y \\ \delta_x & \delta_y & 1 \end{bmatrix} \tag{2.14}$$

Denklem (2.14)'de ortaya çıkarılan eşitliğin sağlanabilmesi için ($[\mathbf{KK^T}]_{\mathbf{3,3}} = 1$) **M** matrisinin $\sqrt{m_{31}^2 + m_{32}^2 + m_{33}^2}$ ile normal hale getirilmesi gerekmektedir. Bu işlemin yapılmasının ardından elde edilecek olan $\mathbf{KK^T}$ matrisi eğer şu şekilde tanımlanır ise:

$$\mathbf{KK^T} = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix} \tag{2.15}$$

Burada; yukarıda verilen normalleştirme işleminden sonra elde edilmesi sebebi ile $f = 1$ olacaktır ve kamera matrisinin barındırdığı içsel parametreler şöyle

hesaplanabilecektir:

$$\delta_x = c \tag{2.16}$$

$$\delta_y = e \tag{2.17}$$

$$\varphi_y^2 + \delta_y^2 = d \;\rightarrow\; \varphi_y = \sqrt{d - e^2} \tag{2.18}$$

$$\varphi_y \gamma + \delta_x \delta_y = b \;\rightarrow\; \gamma = \frac{b - ce}{\sqrt{d - e^2}} \tag{2.19}$$

$$\varphi_x^2 + \gamma^2 + \delta_x^2 = a \;\rightarrow\; \varphi_x = \sqrt{a - c^2 - \gamma^2} \tag{2.20}$$

Kamera matrisinin hesaplanmasının ardından, kameraya dışsal parametrelerin (döndürme matrisi ve pozisyon vektörü) hesaplanması mümkün olacaktır.

$$\mathbf{B} = \mathbf{KR} \rightarrow \mathbf{R} = \mathbf{K^{-1}B} \tag{2.21}$$

$$\mathbf{b} = \mathbf{Kt} \rightarrow \mathbf{t} = \mathbf{K^{-1}b} \tag{2.22}$$

Elde edilen dönüşüm matrisinin (**M**) doğruluğu ve ne hatayla elde edildiğinin tespit edilmesi için, kullanıcı tarafından verilen nokta çiftlerinin (kamera görüntüsü üzerindeki piksel koordinat(lar)ı ve karşılığı olan üç boyutlu nokta(lar)), bu matris kullanılarak geri elde edilmeye çalışılabilir.

$$\widetilde{\mathbf{x}}_\mathbf{i} = \mathbf{M}\mathbf{w}_\mathbf{i} \tag{2.23}$$

Burada $\widetilde{\mathbf{x}}_\mathbf{i}$, elde edilen dönüşüm matrisi ile, kullanıcı tarafından sağlanan i'nci üç boyutlu noktanın çarpılması sonucu ortaya çıkan, kamera görüntü düzlemine ait koordinat sisteminde tanımlı, homojen bir noktadır. Bu noktanın mümkün olduğunca, kullanıcı tarafından sağlanan kamera görüntü noktasına ($\mathbf{x}_\mathbf{i}$) (piksel) yakın olması beklenmektedir. Kullanıcı tarafından sağlanan noktalar için elde edilen uzaklıkların karelerinin ortalaması kamera kalibrasyon sürecinin başarısını göstermektedir. Bu değerin mümkün olduğunda 0'a yakın olması beklenmektedir.

$$MSE = \frac{1}{n}\left(\sum_i^n (\widetilde{x}_i - x_i)^2 + \sum_i^n (\widetilde{y}_i - y_i)^2\right) \tag{2.24}$$

Elde edilen kamera kalibrasyon verisine robot programlama sürecinin ilerleyen aşamalarında ihtiyaç duyulmasından dolayı geri okunması ve rahatlıkla parçalarına ayrılmaya izin verebilecek bir dosya yapısında saklanması gerekmektedir. Bu tez

kapsamında kamera kalibrasyon verisi XML dosya formatında saklanmaktadır. Örnek bir kamera kalibrasyon verisinin saklandığı XML dosyasının görüntüsü, Görsel 2.3'de verilmektedir.

```xml
<?xml version="1.0"?>
<CameraCalibration dateTime="2017-04-30 05:52:31" MeanSquErr="0.63479986756352236">
    <ObjectPoints size="25">
        <ObjectPoint x="0" y="75" z="150" />
        <ObjectPoint x="0" y="75" z="75" />
        ...
    </ObjectPoints>
    <ImagePoints size="25">
        <ImagePoint x="1090.0382080078125" y="535.64935302734375" />
        <ImagePoint x="1094.82958984375" y="761.3468017578125" />
        ...
    </ImagePoints>
    <Matrixes>
        <Matrix name="CameraIntrinsic" rowSize="3" columnSize="3">
            <Element rowIndex="0" columnIndex="0" value="2177.5421580765433" />
            <Element rowIndex="0" columnIndex="1" value="2.2800215545146711" />
            <Element rowIndex="0" columnIndex="2" value="997.12651521062685" />
            <Element rowIndex="1" columnIndex="0" value="0" />
            <Element rowIndex="1" columnIndex="1" value="2168.4719638095949" />
            <Element rowIndex="1" columnIndex="2" value="545.56590811640899" />
            <Element rowIndex="2" columnIndex="0" value="0" />
            <Element rowIndex="2" columnIndex="1" value="0" />
            <Element rowIndex="2" columnIndex="2" value="1" />
        </Matrix>
        <Matrix name="Rotation" rowSize="3" columnSize="3">
            <Element rowIndex="0" columnIndex="0" value="-0.68645281257979707" />
            <Element rowIndex="0" columnIndex="1" value="0.72695673356393331" />
            <Element rowIndex="0" columnIndex="2" value="-0.017788862452168919" />
            <Element rowIndex="1" columnIndex="0" value="-0.013108481363195729" />
            <Element rowIndex="1" columnIndex="1" value="-0.036829727143207619" />
            <Element rowIndex="1" columnIndex="2" value="-0.99923557728641133" />
            <Element rowIndex="2" columnIndex="0" value="-0.72705619027530222" />
            <Element rowIndex="2" columnIndex="1" value="-0.68569488748612761" />
            <Element rowIndex="2" columnIndex="2" value="0.034811168577773784" />
        </Matrix>
        <Matrix name="Translation" rowSize="3" columnSize="1">
            <Element rowIndex="0" columnIndex="0" value="-21.151766849501087" />
            <Element rowIndex="1" columnIndex="0" value="149.3135073895794" />
            <Element rowIndex="2" columnIndex="0" value="768.90616050132519" />
        </Matrix>
        <Matrix name="Projection" rowSize="3" columnSize="4">
            <Element rowIndex="0" columnIndex="0" value="-2219.7768319143061" />
            <Element rowIndex="0" columnIndex="1" value="899.17040820456339" />
            <Element rowIndex="0" columnIndex="2" value="-6.3031373737083838" />
            <Element rowIndex="0" columnIndex="3" value="720978.2943273033" />
            <Element rowIndex="1" columnIndex="0" value="-425.08244502341239" />
            <Element rowIndex="1" columnIndex="1" value="-453.95598472695104" />
            <Element rowIndex="1" columnIndex="2" value="-2147.8225477889519" />
            <Element rowIndex="1" columnIndex="3" value="743271.14230258646" />
            <Element rowIndex="2" columnIndex="0" value="-0.72705619027530222" />
            <Element rowIndex="2" columnIndex="1" value="-0.68569488748612761" />
            <Element rowIndex="2" columnIndex="2" value="0.034811168577773784" />
            <Element rowIndex="2" columnIndex="3" value="768.90616050132519" />
        </Matrix>
    </Matrixes>
</CameraCalibration>
```

Görsel 2.3. *Örnek Kamera Kalibrasyon Verisi XML Dosya İçeriği*

2.1.2. Döner Tabla Kalibrasyonu

Döner tabla kalibrasyonu temel olarak, döner tabla dönüş açısının 0° olduğu konumdaki koordinat ekseni ile endüstriyel robotun taban koordinat ekseni arasındaki ilişkinin tespit edilmesi sürecidir. Bu süreçte elde edilecek dönüşüm matrisi, döner tabla koordinat sisteminde konumu bulunan herhangi bir noktanın, endüstriyel robot taban koordinat sistemine göre karşılığının hesaplanmasında kullanılacaktır.

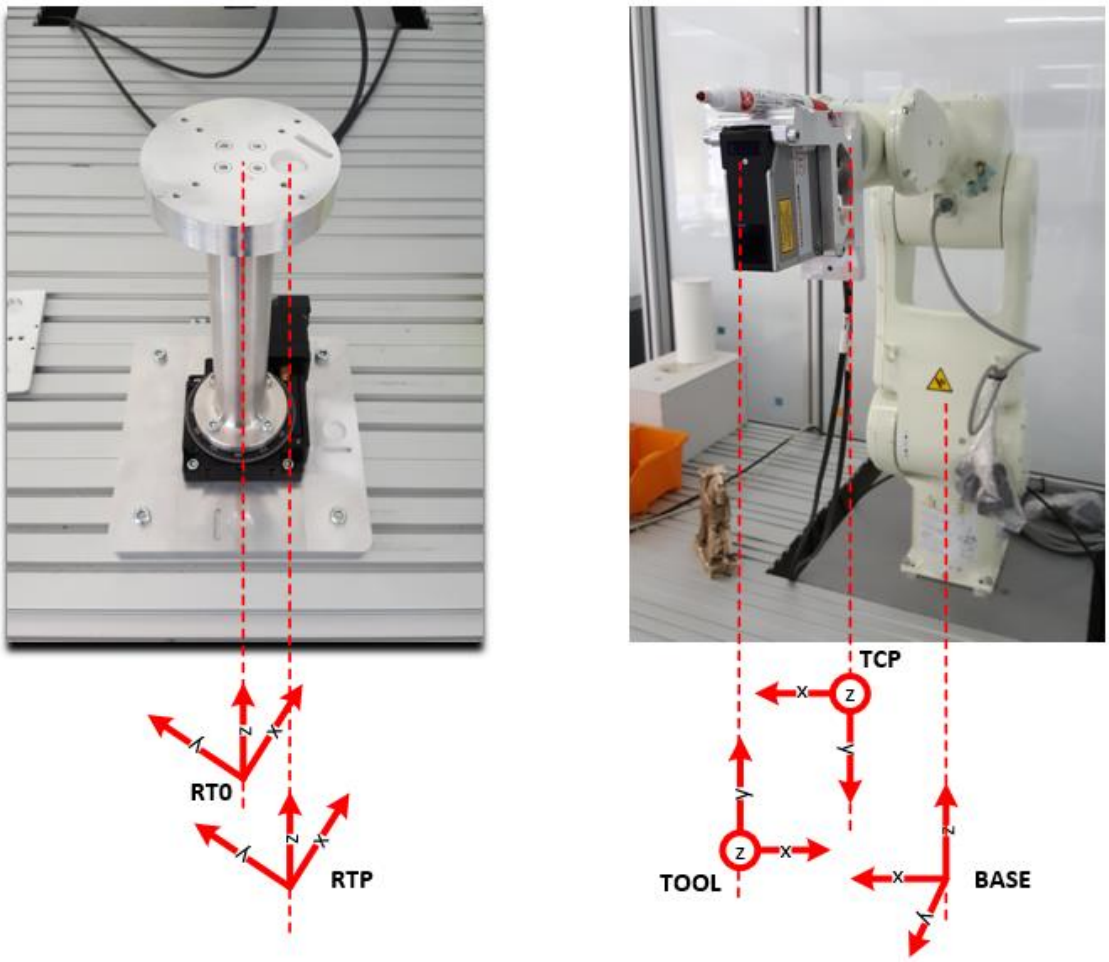
Döner tabla koordinat ekseni ve robot taban koordinat ekseni arasındaki ilişkinin bulunabilmesi için döner tabla yüzeyi üzerinde bulunan noktaların öncelikli olarak robot taban eksenine ait koordinat sistemine göre konumlarının tespit edilmesi gerekmektedir. Bu tespitin yapılabilmesi için döner tabla üzerinde belirlenen kontrol noktalarının hassas bir şekilde robot koordinat sisteminde ölçülmesi ve bu elde edilen ölçümler doğrultusunda aradaki ilişkinin çıkartılması beklenmektedir. Yapılacak ölçümlerin mümkün olan en yüksek hassasiyette yapılması, sürecin doğru tamamlanması için çok büyük öneme sahiptir. Buradaki temel sorun, alınan ölçümlerin robot koordinat sisteminde nasıl temsil edileceği ve döner tabla koordinat sisteminin bu ölçümlere göre nasıl konumlandığının tespit edilmesinin oldukça zorlu bir süreç olmasıdır.

Endüstriyel robot koluna bağlı olan bir lazer profil sensörü yardımı ile endüstriyel robot kolunu düşük hızlarda, döner tabla yüzeyi üzerinde tarama yapacak şekilde hareket ettirilerek elde edilecek nokta bulutu, ihtiyaç duyulan ölçümleri oldukça hassas bir şekilde verecektir. Fakat lazer profil sensöründen elde edilen ölçümlerin robot taban koordinat eksenine dönüştürülmesi gerekmektedir. Bu sürecin gerçekleştirilebilmesi için profil sensörü ölçüm lensine iliştirilmiş bir koordinat ekseni tanımlanması gerekmektedir. Ayrıca tanımlanan bu yeni koordinat ekseni ile robot bağlantı ucu arasındaki ilişkinin bilinmesi de gerekmektedir. Bu dönüşüm, bağlantısı yapılan lazer profil sensörü ve bağlantı ekipmanlarının fiziksel ölçüleri ile belirlenebilmektedir ve robotun uç noktasının pozisyonu ve oryantasyonuna bağımlı değildir. Tez kapsamında kullanılan endüstriyel robot hücresi için tanımlanmış olan koordinat sistemleri Görsel 2.4'de verilmiştir.

$$P^{B} = T_{TCP}^{B} T_{TOOL}^{TCP} \, P^{TOOL} \tag{2.25}$$

Burada; P^{TOOL} lazer profil sensörü yardımı ile ölçümü alınan noktaya, T_{TOOL}^{TCP} lazer profil sensörü ile robot ekipman bağlantı ucunda konumlanmış olan koordinat ekseni arasındaki dönüşüm matrisine, T_{TCP}^{B} ise robot ucunda konumlandırılmış olan koordinat

ekseni ile robot taban koordinat sistemi arasındaki dönüşüm matrisine karşılık gelmektedir. Gerçekleştirilen dönüşüm sonrasında elde edilen yeni nokta, P^B, lazer profil sensörü ile ölçülen noktanın robot taban koordinat sistemindeki ifade edildiğini göstermektedir.



Görsel 2.4. *Döner Tabla ve Endüstriyel Robot Koordinat Sistemleri [38]*

Söz konusu dönüşüm işlemi, sensör tarafından elde edilen bütün veri için gerçekleştirildiğinde, tarama yapılan bölgenin nokta bulutu (döner tabla kalibrasyonu için, döner tabla yüzeyi) robot taban koordinat sistemine göre tanımlanmaktadır.

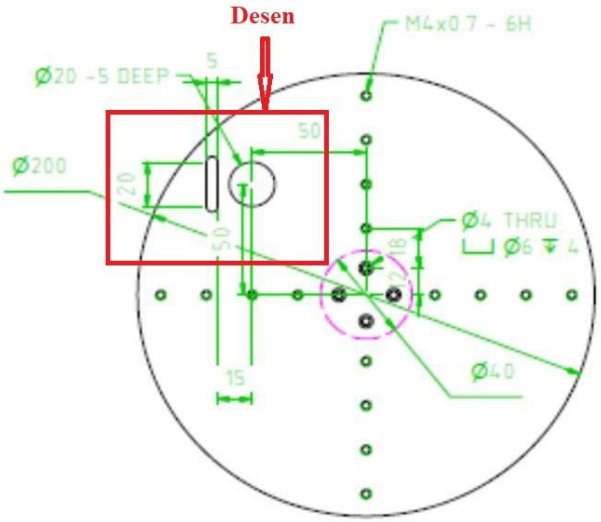
Elde edilen döner tabla yüzeyi nokta bulutu kullanılarak, döner tabla yüzeyinin merkezinde konumlandırılmış olan döner tabla koordinat sistemi ile robot taban koordinat sistemi arasında hassas bir ilişki (dönüşüm matrisi) bulmak mümkün olabilir. Fakat elde edilen yüzey taraması kullanılarak, döner tabla merkez noktasını (döner tabla koordinat sistemi merkezi) bulmak kolay olmasına karşın bu koordinat sisteminin oryantasyonunun

hesaplanması mümkün olmamaktadır. Bunun nedeni hesaplanacak koordinat sisteminin x ve y eksenlerinin elde edilen yüzey taraması üzerinde sonsuz farklı pozisyona oturtulma ihtimalinin bulunmasıdır. Bu nedenle döner tabla üzerinde konumunun ve oryantasyonunun yalnız tek bir şekilde hesaplanması mümkün olan ve aynı zamanda tespit edilmek istenen döner tabla taban koordinat sistemi ile arasındaki ilişkinin (dönüşümün) doğrudan bilindiği başka bir koordinat sistemine ihtiyaç duyulmaktadır. Bu amaçla döner tabla yüzeyi üzerine hassas bir şekilde işlenmiş olan ve sensör ölçümleri ile (dönüşümden sonra nokta bulutu) kolaylıkla tespit edilebilecek bir desen yerleştirilmesi uygun bir çözüm olacaktır.

Tez kapsamında kullanılan döner tabla ve üzerine yerleştirilen ayırt edici desene ait geometrik özellikleri gösteren CAD çizimi Şekil 2.12'de verilmiştir. Döner tabla üzerine yerleştirilmiş desene ait koordinat sistemi ve döner tabla taban koordinat sistemi arasındaki ilişki bilinmektedir. Bu desene ait koordinat sistemi, yüzey taraması ile elde edilen nokta bulutu aracılığı ile tespit edilebilmektedir. Döner tabla koordinat sistemi ile robot taban koordinat sistemi arasındaki dönüşüm aşağıdaki ifade ile elde edilmektedir.

$$T_{RT0}^{B} = T_{RTP}^{B} T_{RT0}^{RTP} \tag{2.26}$$

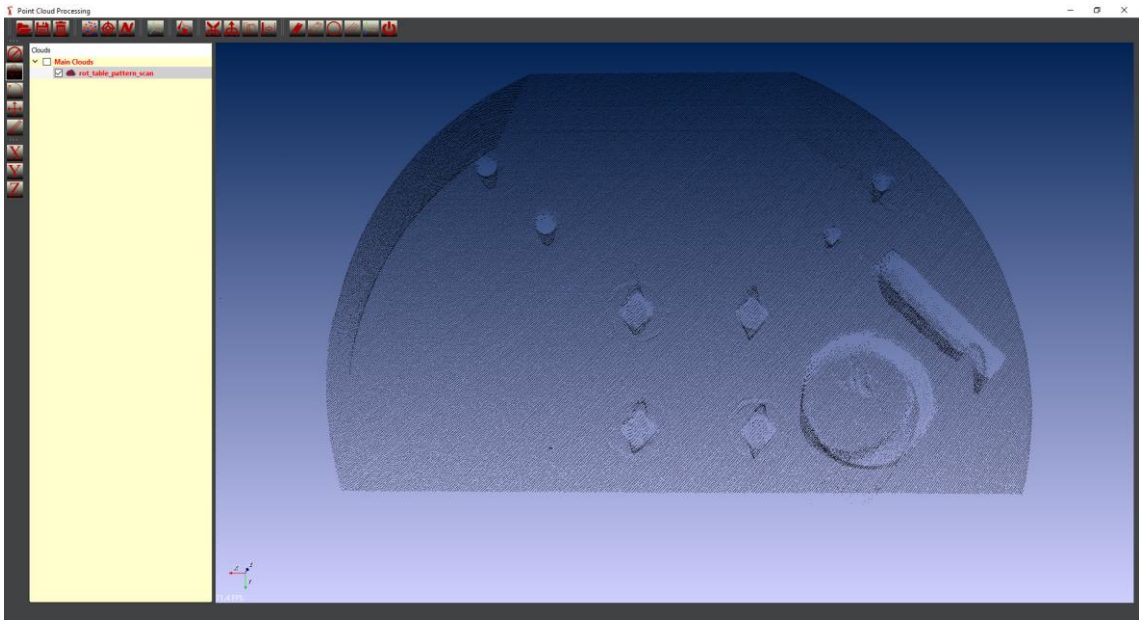
Burada; T_{RT0}^{RTP} döner tabla koordinat sisteminden, döner tabla ayırt edici desenine geçişi sağlayan dönüşüm matrisine karşılık gelmektedir. Bu matris döner tabla ve ilgili desen arasındaki geometrik ilişkiden dolayı bilinmektedir ve hesaplanması gereken bir matris değildir. T_{RTP}^{B} döner tabla ayırt edici desenine ait koordinat sisteminden, robot taban koordinat sistemine geçişi sağlayan dönüşüm matrisini temsil etmektedir. Bu matris, döner tablanın yüzeyinin, lazer profil sensör aracılığı ile taranarak elde edilen nokta bulutu kullanılarak hesaplanması gereken bir matristir. Bu matrisin elde edilmesi ilgili alt bölümde detaylı olarak verilmektedir. T_{RT0}^{B} nihai olarak döner tabla kalibrasyon sürecinde elde edilmesi gereken dönüşüm matrisini temsil etmektedir. Bu matris döner tabla koordinat sisteminden, robot taban koordinat sistemine geçişi sağlayan dönüşüm matrisidir.



Şekil 2.12. *Döner Tabla ve Ayırt Edici Desen CAD Çizimi [38]*

2.1.2.1. *Ayırt Edici Desene ait Koordinat Sisteminin Elde Edilmesi*

Döner tabla yüzeyi üzerinde gerçekleştirilen lazer profil sensör taraması ile elde edilmiş nokta bulutu kullanılarak döner tabla koordinat sisteminin doğrudan elde edilmesi mümkündür değildir. Bunun nedeni düzlemsel olan döner tabla yüzeyi üzerinde gerçekleştirilmiş bir tarama ile elde edilen nokta bulutu üzerine sonsuz oryantasyonda koordinat ekseni yerleştirilebilir. Fakat döner tabla üzerinde geometrik özellikleri belirli ve nokta bulutu üzerinde tespit edilmeye açık bir ayırt edici desen yerleştirildiği taktirde, bu desene ait koordinat sistemini yine aynı nokta bulutu kullanılarak eşsiz bir şekilde bulmak mümkün hale gelmektedir. Bu bölümde bu koordinat sisteminin lazer profil sensör taraması sonucunda elde edilen nokta bulutu kullanılarak nasıl elde edileceği açıklanmaktadır. Görsel 2.5'de döner tabla yüzeyi üzerinde gerçekleştirilmiş olan tarama sonucunda elde edilmiş nokta bulutunu gösterilmektedir. Bu nokta bulutunun temsil edildiği koordinat sistemi bir önceki bölümde anlatıldığı gibi, robot taban koordinat sistemidir.



Görsel 2.5. *Döner Tabla Ayırt Edici Desenini İçeren Döner Tabla Yüzeyi Taraması*

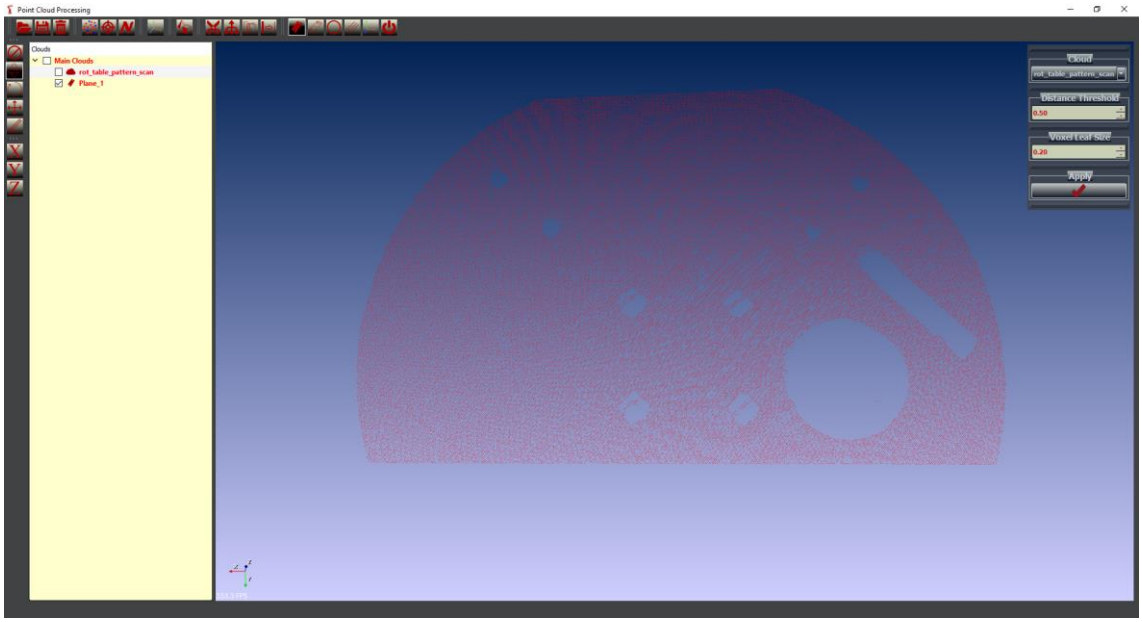
Nokta bulutu formunda üç boyutlu modeli elde edilmiş döner tabla yüzeyi üzerinde konumlanmış olan ayırt edici desene ait koordinat sisteminin tespit edilmesi için sırası ile şu aşağıdaki adımların uygulanması gerekmektedir.

2.1.2.1.1. *Yüzey Düzleminin Tespit Edilmesi*

Lazer tarama sonucunda elde edilen nokta bulutu üzerinde gerçekleştirilmesi gereken ilk işlem yüzey düzleminin tespit edilmesidir. Sürecin devamında uygulanacak olan bütün adımlar, tespit edilen bu yüzey düzlemine ait nokta bulutu kullanılarak yürütülecektir. Yukarıda verilen nokta bulutu için elde edilen yüzey düzlemini oluşturan nokta bulutu Görsel 2.6'da gösterilmektedir. Yüzey düzleminin tespiti Ransac tabanlı düzlem modeli uydurma yöntemi ile gerçekleştirilmektedir [39]. Bu yöntemde verilen nokta bulutuna ait bütün noktaları barındıracak bir düzlem yerleştirilmeye çalışılmaktadır. Fakat bu sürecin her bir tekrarlanmasında o andaki nokta bulutunun bazı noktaları düzlem modeline fazlasıyla uzak bazılarıysa düzlem modeline yakın konumlanmaktadır. Düzlem modeline uzak kalan noktaların dışarıda bırakılmasıyla elde edilen yeni nokta bulutu için sürecin tekrarlanması devam ettirilir. Nihai olarak elde edilen model ve bu modele uzak olmayan noktaların kümesi istenen düzlemi ve bu düzleme ait nokta bulutunu oluşturur (Bkz. Algoritma 2.1).

Algoritma 2.1: Düzlem Modeli Uydur

```
1:  Fonksiyon DüzlemModeliUydur( noktaBulutu,duzlemEsikDegeri )
2:      duzlemModeli ← {noktaBulutu}
3:      ransac ← { duzlemModeli, duzlemEsikDegeri}
4:      if !ModelUydur(ransac) then
5:          return {}
6:      end if
7:      duzlemIcindeKalanlar ← DuzlemIcindeKalanlariAl(ransac)
8:      modelParametreleri ← ModelParametreleriniAl( ransac )
9:      idealModelParametreleri ← ModelParametreleriniIdealYap(
                                    duzlemModeli,
                                    duzlemIcindeKalanlar,
                                    modelParametreleri)
10:     duzlemNoktaBulutu ← NoktalariDuzenle( duzlemModeli,
                            duzlemIcindeKalanlar,
                            idealModelParametreleri)
11:     return {duzlemNoktaBulutu, duzlemIcindeKalanlar,
                idealModelParametreleri}
12: end Fonksiyon
```



Görsel 2.6. *Döner Tabla Yüzeyi Nokta Bulutu*

2.1.2.1.2. *Sınırların Tespiti ve Kümelendirilmesi*

Yüzey düzlemini temsil eden nokta bulutunun elde edilmesinin ardından uygulanacak işlem, yüzey nokta bulutunun sınırlarında bulunan noktaların tespit edilip, bu noktaların birbirlerine yakınlıklarına bağımlı olarak kümelendirilmesidir. Görsel 2.6'da verilen yüzey düzlemine ait nokta bulutu için elde edilen sınır noktaları ve bu

noktaların kümelendirilmiş hali Görsel 2.7'de verilmektedir. Sınır noktalarının tespit edilebilmesi için kaynak olarak kullanılan nokta bulutu üzerinde istatistiksel aykırılık tespiti uygulanması gerekmektedir [40, 41]. Bu işlem bütün nokta bulutu üzerinde gezerek, her bir noktanın belli sayıda komşuluğunda bulunan noktalara olan ortalama uzaklıklarını hesaplar. Ortalama uzaklıkla birlikte bu noktaların komşuluklarına olan uzaklıklarına ait standart sapma değeri de hesaplanmaktadır. Daha sonra bütün nokta bulutu için elde edilen ortalama uzaklık ve standart sapma parametreleri kullanılarak, aykırı noktalar tespit edilip işaretlenir. Sınır bölgelerinde bulunan noktalar bu metot ile elde edilen aykırı noktalardan oluşmaktadır. Belirlenen sınır noktalarını barındıran yeni nokta bulutu daha sonra noktaların birbirleri arasındaki uzaklıklarına bağımlı olarak kümeleme işlemine tabi tutulur [42, 43] (Bkz. Algoritma 2.2-2.5).

Algoritma 2.2: Sınırların Tespiti ve Kümelendirilmesi

```
1: Fonksiyon FindBoundaries( noktaBulutu, ortKomsu,
   stdEsikCarpani, aykiriOlanlarYadaOlmayanlar, kumeHataPayi,
   enKucukKumeBoyutu, enBuyukKumeBoyutu )
2:     sinirNoktaBulutu ← IstatistikselAykirilikTespiti(
                          noktaBulutu, ortKomsu, stdEsikCarpani,
                          aykiriOlanlarYadaOlmayanlar )
3:     kumelemeIndeksleri ← UzaklikTabanliKumeleme(
                            sinirNoktaBulutu, kumeHataPayi,
                            enKucukKumeBoyutu,
                            enBuyukKumeBoyutu )
4:     sinirNoktaBulutlari ← {}
5:     for all kumeNoktaIndeksleri in kumelemeIndeksleri do
6:         kumeNoktaBulutu ← Kopyala(sinirNoktaBulutu,
                             kumeNoktaIndeksleri )
7:         sinirNoktaBulutlari ← {sinirNoktaBulutlari,
                                 kumeNoktaBulutu}
8:     end for
9: end Fonksiyon
```

Algoritma 2.3: İstatistiksel Aykırılık Tespiti

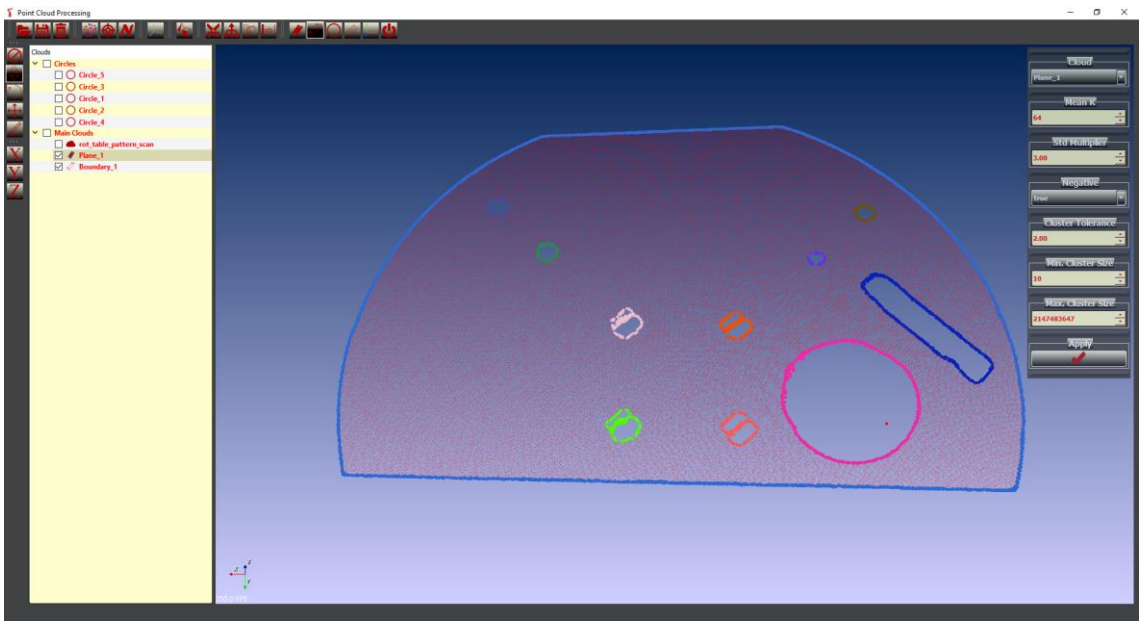
```
1: Fonksiyon IstatistikselAykirilikTespiti( noktaBulutu, ortKomsu,
   stdEsikCarpani, aykiriOlanlarYadaOlmayanlar )
2:     istAykiriSilmeFiltresi ← { noktaBulutu, ortKomsu,
                                  stdEsikCarpani,
                                  aykiriOlanlarYadaOlmayanlar }
3:     sinirIndeksleri ← Filtrele( istAykiriSilmeFiltresi )
4:     sinirNoktaBulutu ← Kopyala( noktaBulutu, sinirIndeksleri )
5:     return sinirNoktaBulutu
6: end Fonksiyon
```

Algoritma 2.4: Uzaklık Tabanlı Kümeleme

```
1: Fonksiyon UzaklikTabanliKumeleme( noktaBulutu, kumeHataPayi,
   enKucukKumeBoyutu, enBuyukKumeBoyutu )
2:     aramaAgaci ← {noktaBulutu}
3:     uzaklıkTabanlıKumlemeFiltresi ← { kumeHataPayi,
                                         enKucukKumeBoyutu,
                                         enBuyukKumeBoyutu, tree,
                                         noktaBulutu }
4:     kumelemeIndeksleri ← Filtrele(
                            uzaklıkTabanlıKumlemeFiltresi )
5:     return kumelemeIndeksleri
6: end Fonksiyon
```

Algoritma 2.5: Nokta Indekslerine Bağımlı Nokta Bulutu Kopyalama

```
1: Fonksiyon Kopyala( noktaBulutu, noktaIndeksleri )
2:     yeniNoktaBulutu ← { {} }
3:     for all noktaIndeksi in noktaIndeksleri do
4:         yeniNoktaBulutu ← {yeniNoktaBulutu,
                              noktaBulutu[noktaIndeksi]}
5:     end for
6:     return yeniNoktaBulutu
7: end Fonksiyon
```



Görsel 2.7. *Sınır Noktaları Tespiti ve Kümelendirilmesi*

2.1.2.1.3. *Çemberlerin ve Düz Çizgilerin Tespiti*

Sınırların tespit edilmesi ve bu sınır bölgelerinin kümelenmesinin ardından ortaya çıkan nokta bulutu kümeleri kullanılarak, bu kümelerin içerisinde çember modeline ve

düz çizgi modeline uygun olanların tespit edilmesini içeren süreçtir. Bu süreçte tespit edilecek çemberlerden biri, döner tabla ayırt edici desenine ait olan çember, düz çizgilerden bir ya da ikisi ise yine ayırt edici desene ait köşeleri yuvarlatılmış dikdörtgen oyuğa ait çizgi(ler) olacaktır (Bkz. Algoritma 2.6-2.10). Görsel 2.7'de verilen sınırları tespit edilmiş ve kümelendirilmiş nokta bulutu serisi için tespit edilen çemberler Görsel 2.8'de, düz çizgiler Görsel 2.9'da sergilenmektedir. Çemberlerin ve düz çizgilerin tespit edilebilmesi için, sınırları içeren ve kümelendirilmiş olan nokta bulutu serilerinin her birine Ransac tabanlı çember modeli ve düz çizgi modeli uydurma yöntemleri uygulanmaktadır [44, 45]. Bu yöntem düzlem modelinin tespiti sürecinde uygulanan yöntemle benzerlik göstermektedir.

Algoritma 2.6: Çemberlerin Tespit Edilmesi

```
1: Fonksiyon CemberTespiti( cemberMesafeEsigi, esikCarpani )
2:     cemberListesi ← {}
3:     for all sinirNoktaBulutu in sinirNoktaBulutlari do
4:         modelUydurmaVerisi ← CemberModeliUydur(
                                  sinirNoktaBulutu,
                                  cemberMesafeEsigi, esikCarpani )
5:         cemberNoktaBulutu ← NoktaBulutunuAl(
                               modelUydurmaVerisi )
6:         modelParametreleri ← ModelParametreleriniAl(
                               modelUydurmaVerisi )
7:         if cemberNoktaBulutu ≠ null then
8:             cemberListesi ← {cemberListesi,
                              [cemberNoktaBulutu,modelParametreleri]
                              }
9:         end if
10:    end for
11: end Fonksiyon
```

Algoritma 2.7: Çember Modeli Uydurma

```
1: Fonksiyon CemberModeliUydur( noktaBulutu, cemberMesafeEsigi,
   esikCarpani )
2:     cemberModeli ← {noktaBulutu}
3:     ransac ← {cemberModeli, cemberMesafeEsigi}
4:     if !ModelUydur(ransac) then
5:         return {}
6:     end if
7:     cemberUzerindeKalanlar ← ModeleUygunOlanlariAl(ransac)
8:     modelParametreleri ← ModelParametreleriniAl( ransac )
9:     idealModelParametreleri ← ModelParametreleriniIdealYap(
                                     cemberModeli,
                                     cemberUzerindeKalanlar,
                                     modelParametreleri)
10:    toplamHata ← ToplamHatayiHesapla( noktaBulutu,
                    idealModelParametreleri )
11:    if toplamHata > esikCarpani * cemberMesafeEsigi then
12:        return {}
13:    end if
14:    cemberNoktaBulutu ← NoktalariDuzenle( cemberModeli,
                           cemberUzerindeKalanlar,
                           idealModelParametreleri )
15:    return {cemberNoktaBulutu, cemberUzerindeKalanlar,
           idealModelParametreleri}
16: end Fonksiyon
```

Algoritma 2.8: Çember Modeli Hatasını Hesaplama

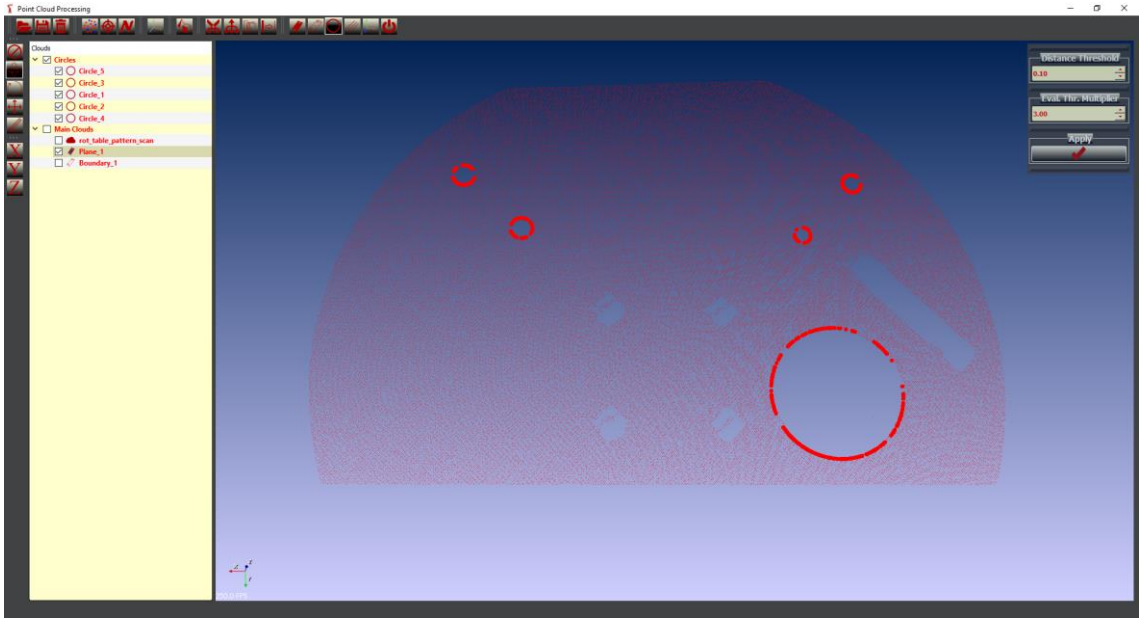
```
1: Fonksiyon ToplamHatayiHesapla(noktaBulutu,modelParametreleri)
2:     toplamHata ← {0}
3:     merkez ← CemberMerkeziniAl(modelParametreleri)
4:     yaricap ← CemberYaricapiniAl(modelParametreleri)
5:     for all nokta in noktaBulutu do
6:         centerDistance ← ||merkez − nokta||
7:         toplamHata ← toplamHata + |centerDistance − yaricap|
8:     end for
9:     toplamHata ← toplamHata / NoktaSayisiAl(noktaBulutu)
10:    return toplamHata
11: end Fonksiyon
```

Algoritma 2.9: Düz Çizgilerin Tespit Edilmesi

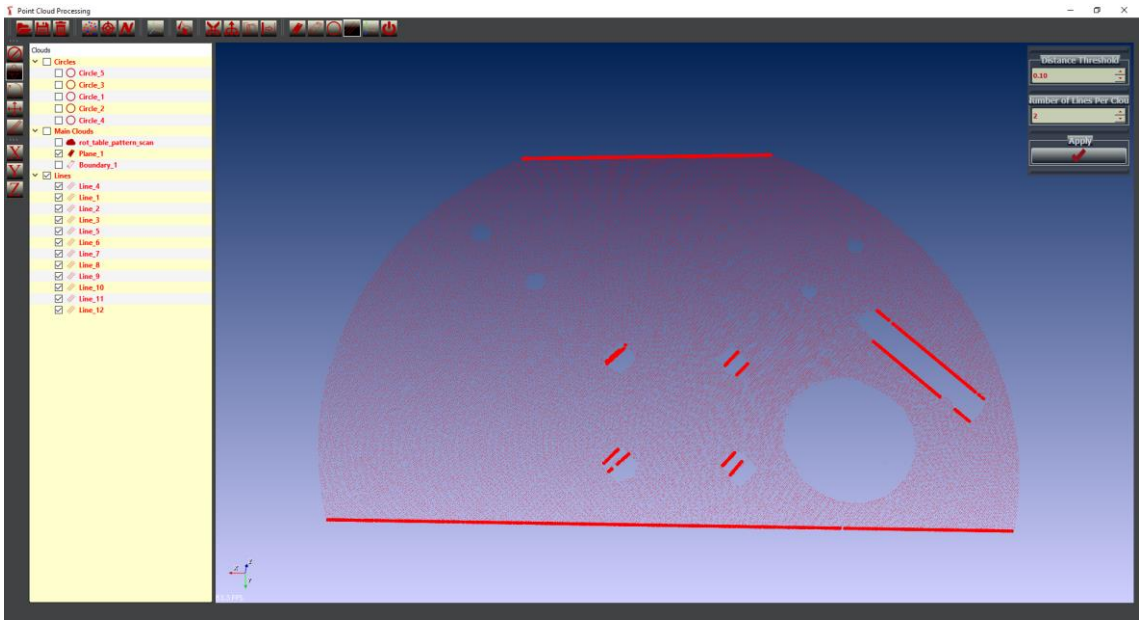
```
1: Fonksiyon DuzCizgiTespiti(uzaklikEsigi,
   noktaBulutuBasinaAranacakDuzCizgiSayisi)
2:     duzCizgiListesi ← {}
3:     n ← {noktaBulutuBasinaAranacakDuzCizgiSayisi}
4:     for all sinirNoktaBulutu in sinirNoktaBulutlari do
5:         for k ← 1, n do
6:             modelUydurmaVerisi ← DuzCizgiModeliUydur(
                                        sinirNoktaBulutu,
                                        uzaklikEsigi)
7:             duzCizgiNoktaBulutu ← NoktaBulutunuAl(
                                        modelUydurmaVerisi )
8:             modelParametreleri ← ModelParametreleriniAl(
                                        modelUydurmaVerisi )
9:             if duzCizgiNoktaBulutu ≠ null then
10:                duzCizgiListesi ← {duzCizgiListesi,
                                      [duzCizgiNoktaBulutu,
                                      modelParametreleri] }
11:                sinirNoktaBulutu ← {sinirNoktaBulutu \
                                        duzCizgiNoktaBulutu}
12:            end if
13:        end for
14:    end for
15: end Fonksiyon
```

Algoritma 2.10: Düz Çizgi Modeli Uydurma

```
1: Fonksiyon DuzCizgiModeliUydur( noktaBulutu, uzaklikEsigi )
2:     duzCizgiModeli ← {noktaBulutu}
3:     ransac ← {duzCizgiModeli, uzaklikEsigi}
4:     if !ModelUydur(ransac) then
5:         return {}
6:     end if
7:     modelIcerisindeKalanlar ← ModeleUygunOlanlariAl(ransac)
8:     modelParametreleri ← ModelParametreleriniAl( ransac )
9:     idealModelParametreleri ← ModelParametreleriniIdealYap(
                                    duzCizgiModeli,
                                    modelIcerisindeKalanlar,
                                    modelParametreleri)
10:    duzCizgiNoktaBulutu ← NoktalariDuzenle( duzCizgiModeli,
                             modelIcerisindeKalanlar,
                             idealModelParametreleri )
11:    return {duzCizgiNoktaBulutu, modelIcerisindeKalanlar,
           idealModelParametreleri}
12: end Fonksiyon
```



Görsel 2.8. *Çemberlerin Tespit Edilmesi*



Görsel 2.9. *Düz Çizgilerin Tespit Edilmesi*

2.1.2.1.4. *Desen koordinat Sisteminin Hesaplanması*

Desen koordinat sisteminin hesaplanması için önceki adımların başarı ile gerçekleştirilip, desen koordinat sistemine ait çemberin ve düz çizgilerden birinin seçilmiş olması gerekmektedir (Bkz. Görsel 2.10). Tespit edilen çemberin sahip olduğu

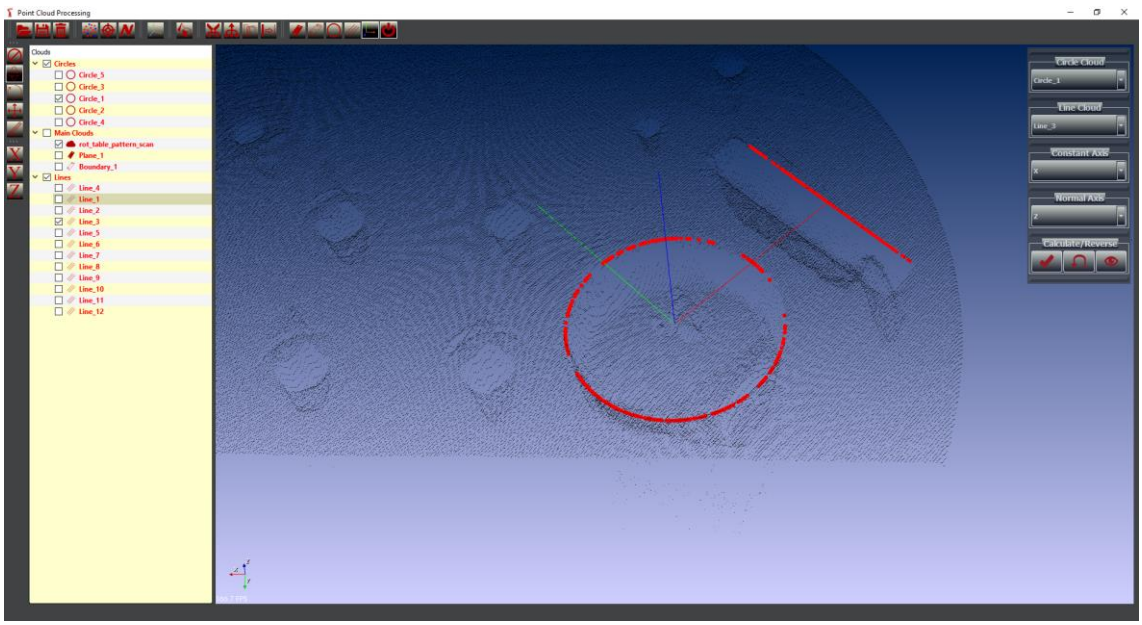
tanımlayıcı özelliklerden biri olan, çember merkezini temsil eden üç boyutlu nokta, tespit edilecek olan koordinat sisteminin merkezinin robot taban koordinat sistemindeki karşılığının ne olacağını göstermektedir. Yine aynı çemberin özelliklerinden olan yüzey normal vektörü ise aranan koordinat sisteminin z eksenini ortaya çıkartmaktadır. Bu çemberin merkezinden, desene ait olan köşeleri yuvarlatışmış dikdörtgen üzerinde tespit edilmiş olan düz çizgilerden birine olan dik uzaklık vektörü (çember merkezi ile ilgili düz çizgi arasındaki en kısa mesafeyi gösteren vektör) ise elde edilecek olan koordinat sisteminin x eksenini oluşturacaktır. Koordinat sisteminin y ekseni ise tespit edilmiş olan iki vektör arasında gerçekleştirilen vektörel çarpım ile elde edilmektedir (Bkz. Algoritma 2.11 ve 2.12). Görsel 2.10 hesaplanmış olan koordinat sistemini göstermektedir. Elde edilen bu koordinat sistemi Denklem (2.6)'da ihtiyaç duyulan döner tabla ayırt edici desen ve robot taban ekseni arasındaki ilişkiyi veren dönüşüm matrisini (T^B_{RTP}) oluşturmaktadır. Döner tabla koordinat sistemi ve robot taban koordinat sistemi arasındaki dönüşüm matrisi (T^B_{RT0}) ise yine aynı denklem ile hesaplanmaktadır.

Algoritma 2.11: Desen Koordinat Sisteminin Hesaplanması

Fonksiyon DesenKoordinatSistemiHesapla(*cemberIndeksi*, *duzCizgiIndeksi*, *sabitEksen*, *normalEksen*)
  cemberParametreleri ← ModelParametreleriniAl(*cemberListesi*[*cemberIndeksi*])
  duzCizgiParametreleri ← ModelParametreleriniAl(*duzCizgiListesi*[*duzCizgiIndeksi*])
  o_c ← CemberMerkeziniAl(*cemberParametreleri*)
  n_c ← CemberNormaliniAl(*cemberParametreleri*)
  ▷ Çember merkezi ve düz çizgi arasındaki en kısa mesafe vektörünü hesapla
  p_1 ← DuzCizgiNoktasiniAl(*duzCizgiParametreleri*)
  v ← DuzCizgiVectorunuAl(*duzCizgiParametreleri*)
  $p_2 \leftarrow p_1 + v$
  $p_1p_2 \leftarrow p_2 - p_1$
  $t \leftarrow -\frac{(p_1 - o_c)\cdot p_1p_2}{||p_1p_2||^2}$
  $p_3 \leftarrow p_1 + v * t$
  $d \leftarrow p_3 - o_c$
  $\{X,Y,Z\}$ ← KoordinatSistemiOlustur(d, n_c, *sabitEksen*, *normalEksen*)
  return $\{X,Y,Z\}$
end Fonksiyon

Algoritma 2.12: Koordinat Sistemi Olustur

```
1: Fonksiyon KoordinatSistemiOlustur( d, n_c, sabitEksen, normalEksen )
2:     if sabitEksen == "X" && normalEksen == "Z" then
3:         X ← ||d||
4:         Z ← ||n_c||
5:         Y ← Z × X
6:     else if sabitEksen == "X" && normalEksen == "Y" then
7:         X ← ||d||
8:         Y ← ||n_c||
9:         Z ← X × Y
10:    else if sabitEksen == "Y" && normalEksen == "X then
11:        Y ← ||d||
12:        X ← ||n_c||
13:        Z ← X × Y
14:    else if sabitEksen == "Y" && normalEksen == "Z" then
15:        Y ← ||d||
16:        Z ← ||n_c||
17:        X ← Y × Z
18:    else if sabitEksen == "Z" && normalEksen == "X" then
19:        Z ← ||d||
20:        X ← ||n_c||
21:        Y ← Z × X
22:    else if sabitEksen == "Z" && normalEksen == "Y" then
23:        Z ← ||d||
24:        Y ← ||n_c||
25:        X ← Y × Z
26:    end if
27:    return {X,Y,Z}
28: end Fonksiyon
```



Görsel 2.10. *Desen Koordinat Sistemi (Kırmızı: X, Yeşil: Y, Mavi: Z)*

Elde edilen döner tabla kalibrasyon verisine robot programlama sürecinin ilerleyen aşamalarında ihtiyaç duyulmasından dolayı geri okunması ve rahatlıkla parçalarına ayrılmaya izin verebilecek bir dosya yapısında saklanması gerekmektedir. Bu tez kapsamında döner tabla kalibrasyon verisi XML dosya formatında saklanmaktadır. Görsel 2.11, örnek bir döner tabla kalibrasyon verisinin saklandığı XML dosyasının göstermektedir.

```
<?xml version="1.0"?>
<TransformationConfigs dateTime="2017-11-06 06:45:54">
    <CalibrationHomeToRotaryTableHomeTransformationInfo>
        <RotationAngles EulerConvension="ZYZ" Alpha="90" Beta="0" Gamma="0" />
        <TranslationVector x="0" y="0" z="8" />
    </CalibrationHomeToRotaryTableHomeTransformationInfo>
    ...
    <RotaryTablePredefinedRotionForTriangulation angle="5" />
    <RotaryTablePatternToRotaryTableHomeTransformationInfo>
        <RotationAngles EulerConvension="ZYZ" Alpha="0" Beta="0" Gamma="0" />
        <TranslationVector x="15" y="-20" z="0" />
    </RotaryTablePatternToRotaryTableHomeTransformationInfo>
    <RobotBaseToRotaryTableTransformationInfo>
        <RotationAngles EulerConvension="ZYZ" Alpha="-170.05000000000001"
                        Beta="0.208616" Gamma="-53.989600000000003" />
        <TranslationVector x="603.62" y="635.04999999999995" z="-10.52" />
    </RobotBaseToRotaryTableTransformationInfo>
    <ToolToToolConnectionPointTransformationInfo>
        <RotationAngles EulerConvension="ZYZ" Alpha="180" Beta="0" Gamma="0" />
        <TranslationVector x="0" y="0" z="132.5" />
    </ToolToToolConnectionPointTransformationInfo>
    ...
</TransformationConfigs>
```

Görsel 2.11. *Örnek Kamera Kalibrasyon Verisi XML Dosya İçeriği*

2.2. İlgilenilen Bölgenin Belirlenmesi

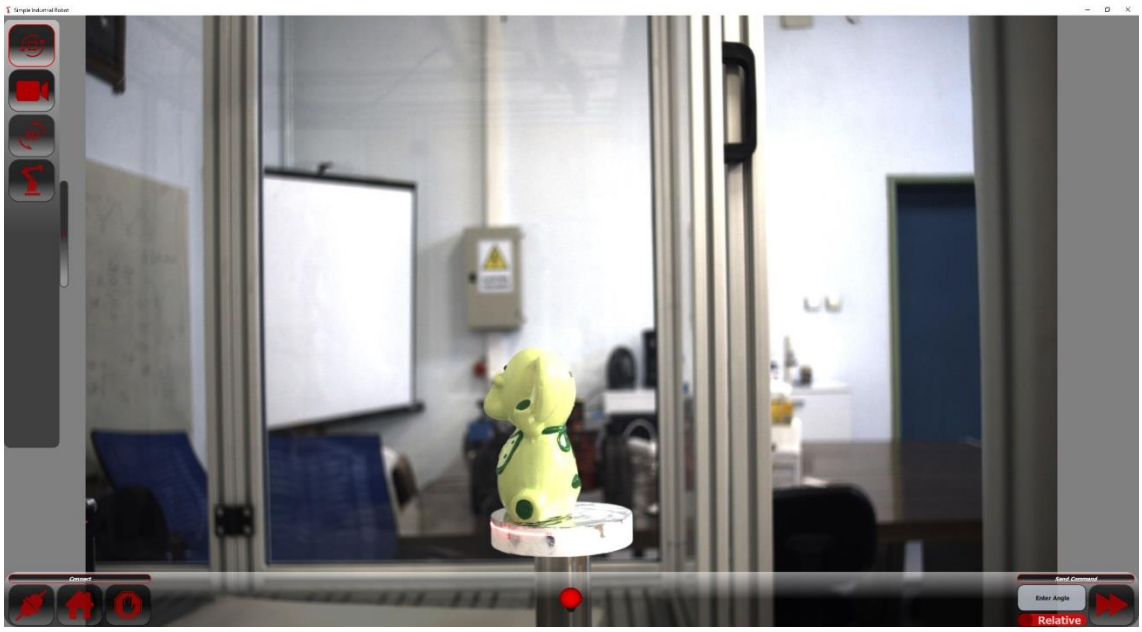
İlgilenilen bölge, endüstriyel robot hücresi içerisinde, döner tabla üzerinde konumlandırılmış olan nesnenin, robot tarafından gerçekleştirilecek olan endüstriyel işlemin yapılacağı bölümünü ifade etmektedir.

İlgilenilen bölgenin tez kapsamında geliştirilen kullanıcı arayüzü kullanılarak operatör tarafından belirlenmesi ve belirlenen bu bölgenin detaylı üç boyutlu modelinin çıkartılmasının sağlanması bu bölümün kapsamında ele alınacaktır. Ayrıca bu işlem adımı, endüstriyel robot programlama sürecinin (kalibrasyon sürecinden geçirilmiş bir robotik hücre için) başlangıç adımıdır. Bu bağlamda tez kapsamında sunulan operatör deneyimi ve programlama becerisi ihtiyacının en aza indirilmesi hedefini koruması önem arz etmektedir. Geliştirilen kullanıcı arayüzü kullanılarak gerçekleştirilen bu süreç

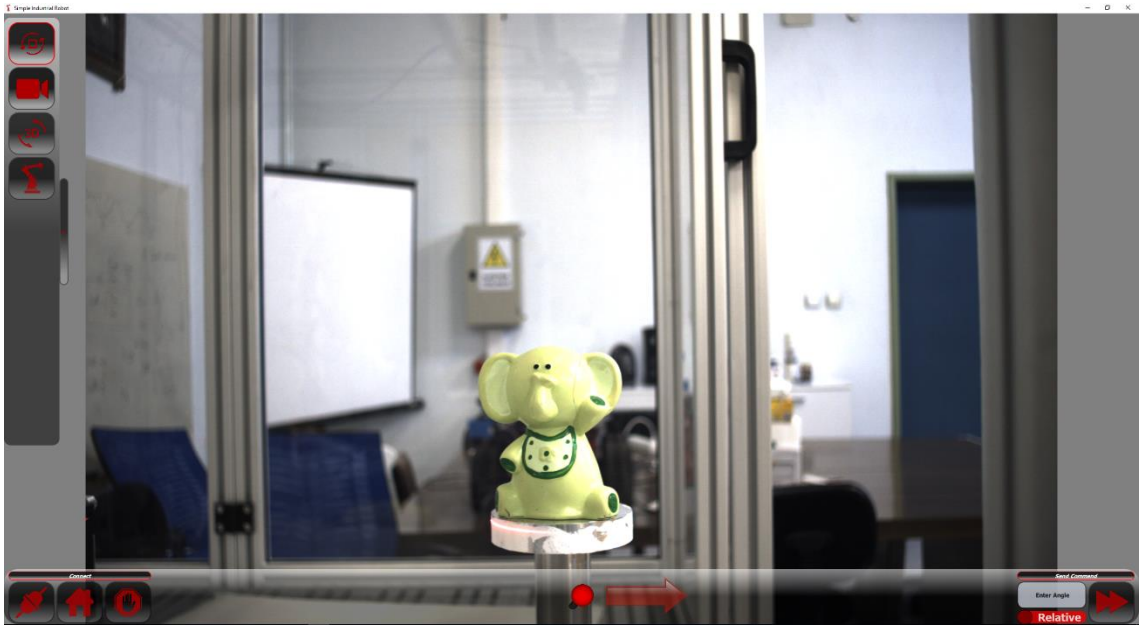
kapsamında aşağıdaki adımların gerçekleştirilmesi gerekmektedir. Bu adımların tamamında operatörün yapması beklenen işlemler basit arayüz komutlarından ibarettir.

2.2.1. İlgilenilen Bölgenin Kamera Görüş Alanına Sokulması

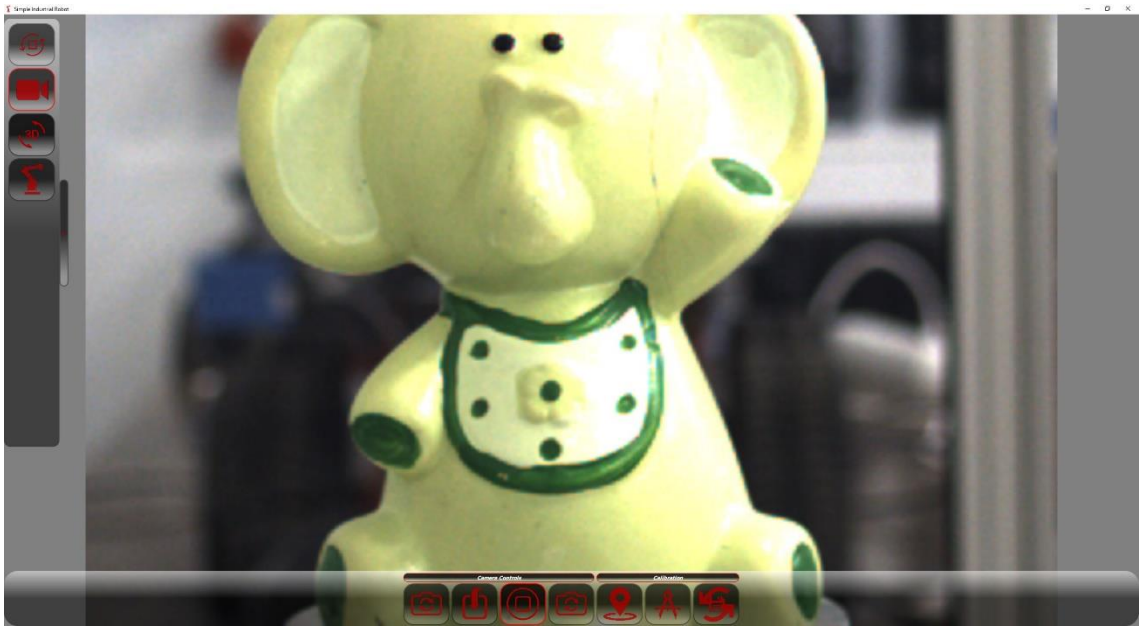
İşlem görecek endüstriyel parça endüstriyel robot hücresinde sabit konumda bulunan ve robot taban ekseni ile arasındaki ilişkinin belli olduğu (T_{RT0}^{B}) bir döner tabla üzerinde konumlandırılmaktadır. Ayrıca yine aynı hücre içerisinde konumlandırılmış olan bir kamera tarafından izlenmektedir. Kamera ile döner tabla arasındaki ilişki (T_{CALIB}^{RT0}) bir kalibrasyon nesnesi (Bkz. Görsel 2.1) kullanılarak elde edilmiştir. Operatör bu adımda, işlem uygulanacak nesnenin ilgili bölgesinin kameranın görüş alanına sokması gerekmektedir. Bu işlem için kullanıcı arayüzüne eklenmiş kontrol bileşenlerini kullanarak, işlem gerçekleştirilecek nesneyi taşıyan döner tablayı döndürecektir (Görsel 2.12 ve 2.13). Döner tablanın dönüşü ile kamera tarafından görüntülenen çalışma alanı içerisinde nesnenin ilgili bölgesi aktif olarak görüntülenebilecektir. Daha sonra operatör yine kullanıcı arayüzünü kullanarak nesne üzerinde yakınlaştırma işlemi uygulayarak, istenen bölgenin daha detaylı görüntülenmesini sağlayacaktır. (Görsel 2.14).



Görsel 2.12. *İşlenecek Nesne Görüntüsü (Döner Tabla Dönüşü Öncesi)*



Görsel 2.13. *İşlenecek Nesne Görüntüsü (Döner Tabla Dönüşü Esnası)*

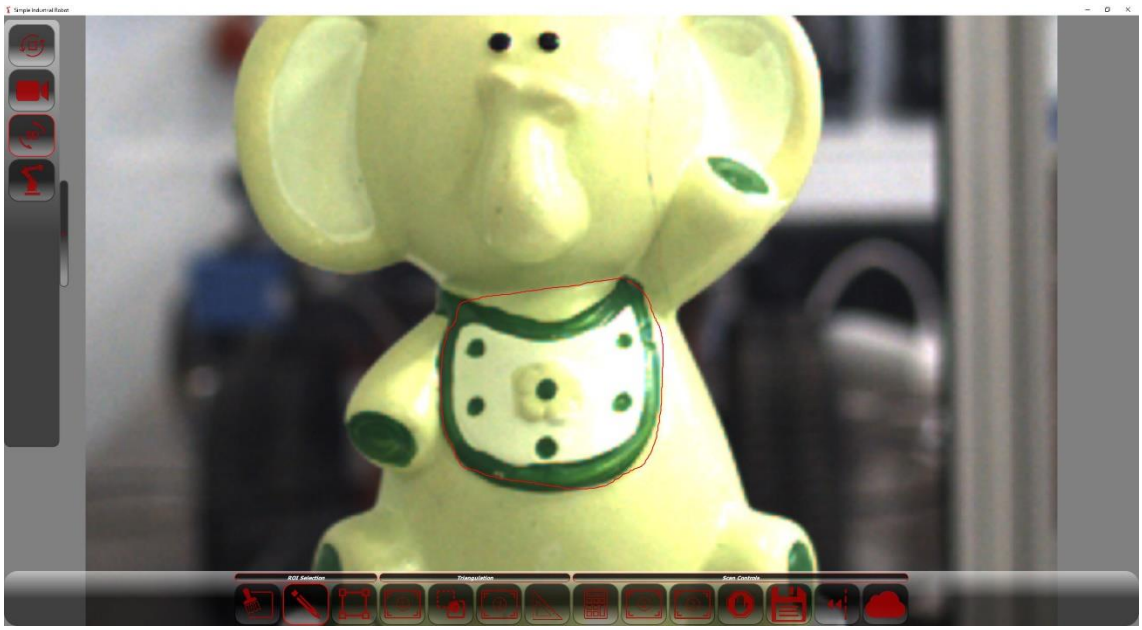


Görsel 2.14. *İşlenecek Nesnenin Görüntüsü (Yakınlaştırma İşlemi Sonrası)*

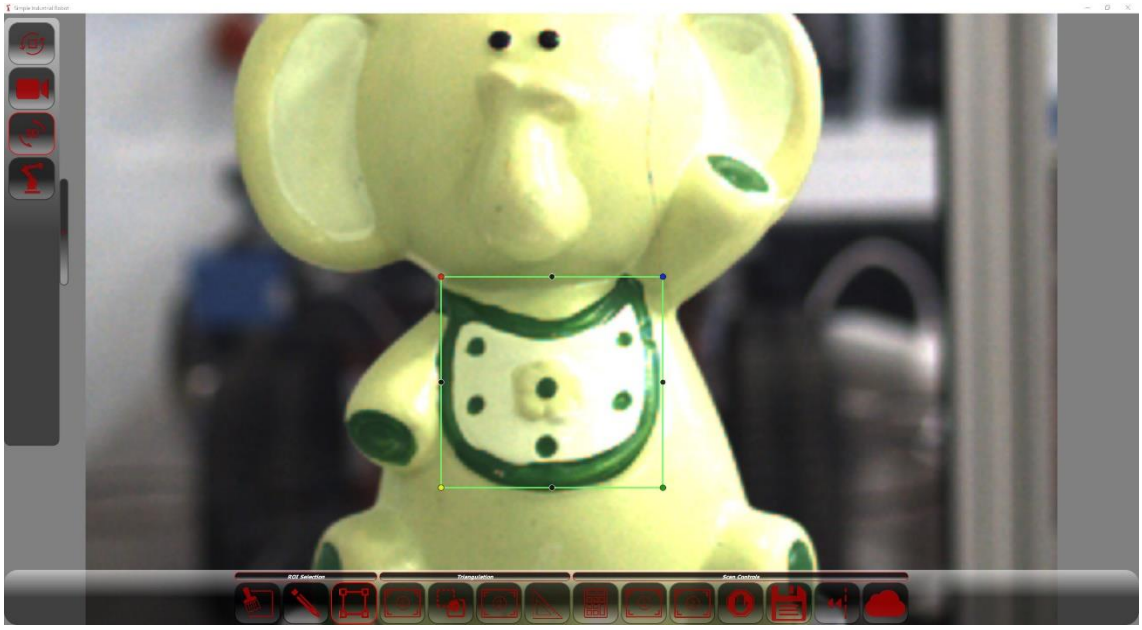
2.2.2. Bölgenin Seçilmesi

İşlem görecek nesneye ait ilgili bölgenin kamera görüş alanına sokulmasının ardından, programlama işlemini gerçekleştirecek operatörün, işlem yapılacak bölgeyi belirlemesi gerekmektedir. Belirleme işlemi operatör tarafından, arayüz üzerinden Görsel 2.15'de gösterildiği gibi çizim aracıyla serbest çizim hareketi kullanılarak çizilir. Daha

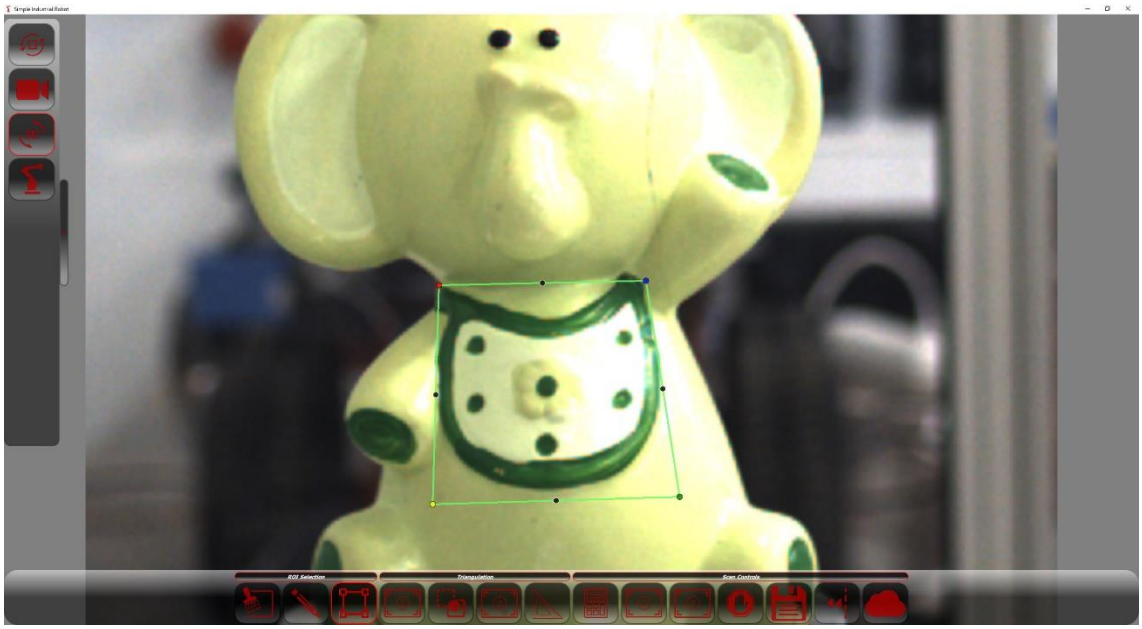
sonra bu çizimi içerisinde barındıracak en küçük alana sahip dörtgen hesaplanarak görüntülenir. Bu dörtgene bu tez kapsamında *çevreleyen dörtgen* ismiyle değinilmektedir. (Bkz. Görsel 2.16). Serbest çizim hareketi kullanılarak gerçekleştirilen çizim esnasında, işlem görecek bölgenin dışarısına taşan ya da bölgeyi tam olarak kapsamayan çizimlerden dolayı, hesaplanan dörtgen istenen bölgeyi doğru olarak ifade edememesi durumuna karşın, kullanıcı (operatör) bu dörtgene Görsel 2.17'de gösterildiği gibi dörtgen üzerine konumlandırılmış, tutma noktalarını kullanarak müdahale edebilir.



Görsel 2.15. *Serbest Çizim Hareketi ile Belirlenen Bölge (Kırmızı Çizgi)*



Görsel 2.16. *Serbest Çizim Hareketi ile Belirlenen Bölgeye ait Çevreleyen Dörtgen*



Görsel 2.17. *Hesaplanan Çevreleyen Dörtgene Operatör Müdahalesi*

2.2.3. Seçilen Bölgenin Üç Boyutlu Uzayda Karşılığının Bulunması

Operatör tarafından belirlenen çevreleyen dörtgen, ilgilenilen bölgenin kamera görüntü düzlemine ait koordinat eksenindeki konumunu ifade etmektedir. Bu dörtgen iki boyutlu bir uzayda kapalı bir bölge tanımlamaktadır ve köşelerinin bulunduğu görüntü koordinatları ile ifade edilmektedir. Fakat operatör bu bölgeyi seçerek, işlenecek bölgenin üç boyutlu uzayda nerde olduğunu belirlemeye çalışmaktadır. Bu bilgiye ilgilenilen bölgenin lazer profil sensörü ile tarama yapılması sırasında ihtiyaç duyulacaktır.

Seçilen bölgenin üç boyutlu uzaya geri yansıtılması süreci, çevreleyen dörtgenin görüntü koordinat sisteminde tanımlı olan köşe noktalarının, kamera kalibrasyon nesnesine ait koordinat sistemindeki karşılıkları olan üç boyutlu noktaların bulunmasını içermektedir. Fakat iki boyutlu görüntü koordinat sisteminde tanımlı iki boyutlu bir noktayı, üç boyutlu bir uzaya taşınması, tek bir görüntü, ya da fazladan sağlanan bir derinlik bilgisine sahip olmadan mümkün değildir. Çünkü, üç boyutlu uzayda konumlanmış olan (bulunmak istenen) noktanın, iki boyutlu görüntü düzlemine aktarılması sırasında üçüncü boyut olan derinlik bilgisi, geri kazanılamayacak şekilde kaybolmaktadır.

Kaybolan derinlik bilgisini, dolayısıyla, iki boyutlu görüntü düzleminde

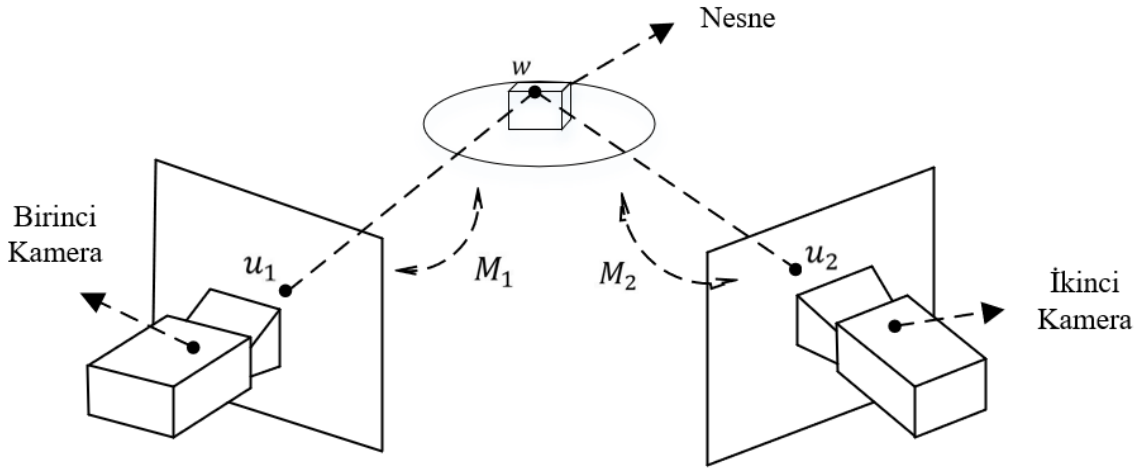
konumlanmış olan noktanın, üç boyutlu uzaydaki karşılığını, aynı noktanın bir başka görüntü üzerindeki konumu yeniden tespit edilebilir ve bu yeni görüntünün bir önceki görüntü ile arasındaki ilişki belirlenebilir ise geri kazanmak mümkün hale gelmektedir [46].

İhtiyaç duyulan ikinci görüntünün kontrollü olarak elde edilebilmesi ve çevreleyen dörtgenin, elde edilen ikinci görüntü üzerinde yeniden oluşturulabilmesi belli başlı işlem basamakları ile mümkün olmaktadır.

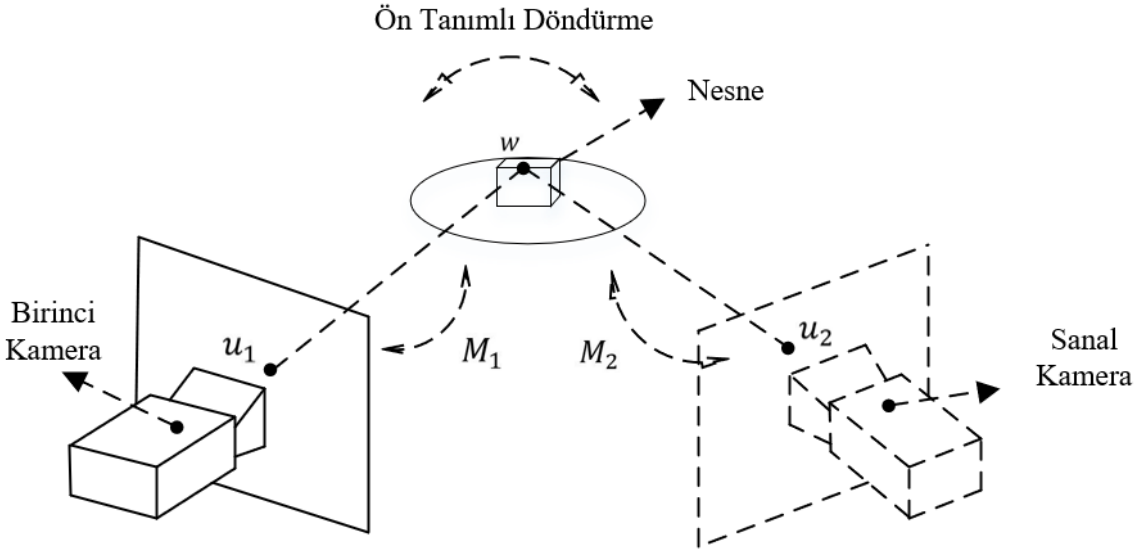
2.2.3.1. *İkinci Görüntünün Elde Edilmesi*

Derinlik bilgisini yeniden ortaya çıkarabilmek için ihtiyaç duyulan ikinci görüntünün oluşturulabilmesi için sistem içerisine ikinci bir kamera yerleştirilebilir. Yerleştirilen diğer kameradan elde edilecek görüntü ihtiyaç duyulan ikinci görüntü olarak kullanılabilir (Bkz. Şekil 2.13). Fakat bu durum sistem maliyetini artıracaktır. Bir başka seçenek ise kameranın hareket ettirilmesi olarak ön görülebilir. Fakat bu durum, sistem kurulumu sırasında gerçekleştirilen kamera kalibrasyon süreci ile elde edilen kalibrasyon verisinde değişikliğe sebep olacağından dolayı tercih edilebilecek bir yöntem değildir. Nesnenin döner tabla üzerindeki konumunun değiştirilmesi ise nesnenin pozisyonunun kaybedilmesine sebep olabilir. Dahası, işlem görecek nesneye operatörün müdahalesine ihtiyaç duyulacağından, güvenlik sorunları yaratmaktadır.

Döner tablanın üzerinde konumlandırışmış olan nesnenin, önceden belirlenmiş ve ilgilenilen bölgeyi kaybetmeyecek kadar küçük bir açı ile döner tabla vasıtasıyla döndürülmesi tek bir kamera kullanılarak, aynı nesneye ait ikinci bir görüntü elde edilmesine olanak sağlamaktadır (Bkz. Şekil 2.14).



Şekil 2.13. *İki Kamera ile İki Farklı Görüntünün Elde Edilmesi*



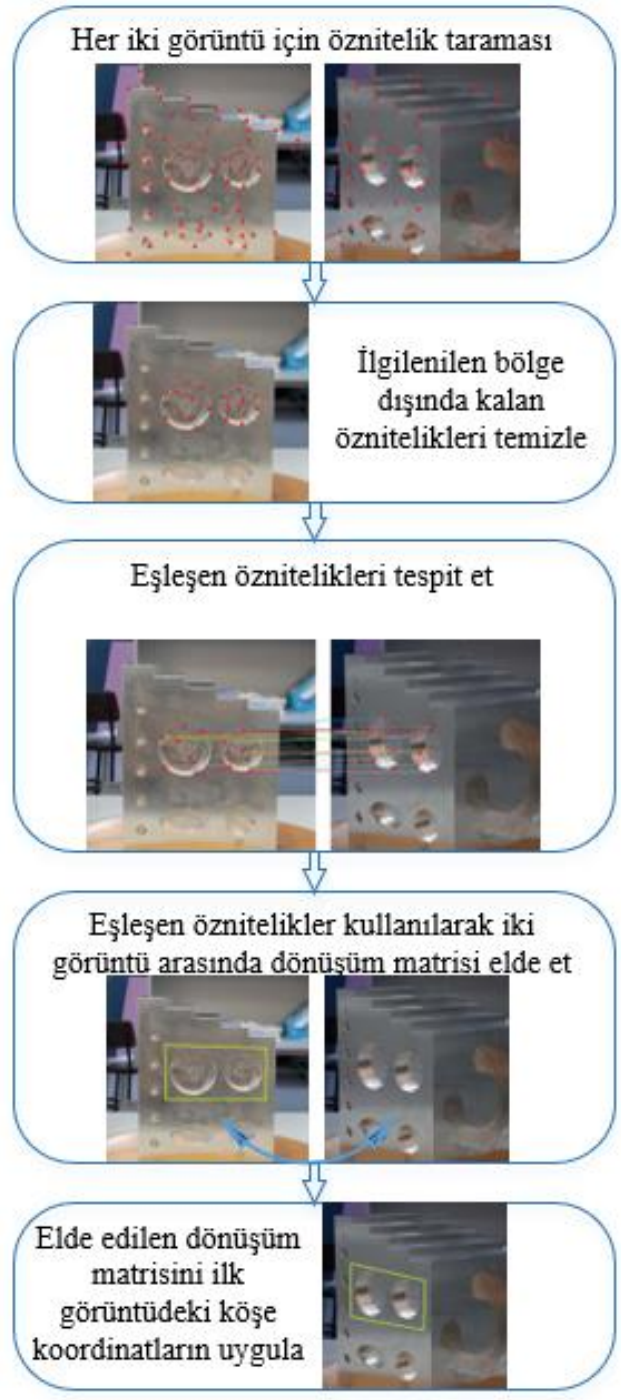
Şekil 2.14. *Bir Kamera ve Döndürme İşlemi ile İki Farklı Görüntünün Elde Edilmesi*

2.2.3.2. *Çevreleyen Dörtgenin İkinci Görüntü Üzerinde* Yeniden Oluşturulması

Döner tablanın önceden belirlenmiş bir açı ile döndürülmesi sonrası elde edilen ikinci görüntü üzerinde çevreleyen dörtgenin yeniden oluşturulması gerekmektedir. Oluşturulan yeni çevreleyen dörtgenin köşe koordinatları ile döndürme işlemi gerçekleştirilmeden önce operatör tarafından belirlenen dörtgenin köşe noktalarının, nesne üzerinde gösterdiği noktanın mümkün olduğunca aynı noktaya karşılık gelmesi bu süreçte önem arz etmektedir. İki görüntü üzerinde birbirinin karşılığı olan bu noktalar daha sonraki adımlarda, bu köşe koordinatlarının üç boyutlu uzaydaki yerinin tespitinde

kullanılacaktır.

Çevreleyen dörtgeni ikinci görüntü üzerinde oluşturulması süreci, her iki görüntü üzerinde eşleşen öznitelikler ve öznitelikleri konumlarındaki değişimlerinin tespit edilmesi ile mümkün kılınabilmektedir. Bu işlem için izlenmesi gereken süreç Şekil 2.15'de özetlenmektedir.



Şekil 2.15. *İlgilenilen Bölgenin İkinci Görüntü Üzerinde Yeniden Oluşturulması [47]*

Görüntüler üzerinde bulunan öznitelikler kullanılarak, döndürme işlemi ile birinci görüntü üzerindeki bir bölgenin ikinci görüntüde nereye ne şekilde kaydığını tespit etmek için önemli bir bilgi kaynağıdır. Bu kapsamda, elde edilen her iki görüntü üzerinde yaygın olarak kullanılan öznitelik tespiti ve çıkarımı algoritmalarından birini kullanmak uygun bir seçim olacaktır (Bkz. Algoritma 2.13).

Algoritma 2.13: Cevreleyen Dörtgenin İkinci Görüntü Üzerinde Oluşturulması

```
1: Fonksiyon CevreleyenDortgeniOlustur(ilkGoruntu,ikinciGoruntu,
   algoritma , cevreleyenDortgen)
2:     oznitelikTaramaSonucu ← OznitelikleriBul( ilkGoruntu,
                               ikinciGoruntu, algoritma,
                               cevreleyenDortgen)
3:     ilkGoruntuTanimlayicilari ← IlkGoruntuTanimlayicilariniAl(
                                   oznitelikTaramaSonucu )
4:     ikiGoruntuTanimlayicilari ← IkiGoruntuTanimlayicilariniAl(
                                   oznitelikTaramaSonucu )
5:     uygunEslesmeListesi ← OznitelikleriEslestir(
                             ilkGoruntuTanimLayicilari,
                             ikiGoruntuTanimlayicilari )
6:     donustumMatrisi ← DonusumMatrisiniBul(
                         uygunEslesmeListesi )
7:     ikinciGoruntuCevreleyenDortgen ← DonusumUygula(
                                        dortgenKoseleri,
                                        donusumMatrisi )
8:     return ikinciGoruntuCevreleyenDortgen
9: end Fonksiyon
```

Tez kapsamında geliştirilen sistemde AKAZE [48, 49], SURF [50, 51], SIFT [52, 53], ORB [54, 55] algoritmalarından biri kullanıcı (operatör) tercihine bağlı olarak öznitelik tespiti için kullanılmaktadır. Birinci görüntü için tespit edilen özniteliklerin pek çoğu, operatör tarafından tanımlanmış olan ilgilenen bölgenin dışında kalmaktadır. Bu nedenle tespit edilen özniteliklerin birinci bölgenin içinde kalanları saklayıp, diğerlerini atacak şekilde filtrelenmesi gerekmektedir (Bkz. Algoritma 2.14).

Algoritma 2.14: Oznitelik Taraması

```
1: Fonksiyon OznitelikleriBul( ilkGoruntu, ikinciGoruntu, algoritma ,
   cevreleyenDortgen)
2:     oznitelikIslemcisi ← Yarat(algoritma)
3:     ilkGoruntuOznitenikleri ← OznitelikleriBul(
                                 oznitelikIslemcisi, ilkGoruntu )
4:     ikinciGoruntuOznitenikleri ← OznitelikleriBul(
                                    oznitelikIslemcisi, ikinciGoruntu
                                    )
5:     filtrelenmisOznitelikler ← {}
6:     for all oznitelik in ilkGoruntuOznitenikleri do
7:         if PIKSEL(oznitelik) ∈ cevreleyenDortgen then
8:             filtrelenmisOznitelikler ← {filtrelenmisOznitelikler,
                                          oznitelik}
9:         end if
10:    end for
11:    ilkGoruntuTanimLayicilari ← OznitelikleriTanimla(
                                    oznitelikIslemcisi, ilkGoruntu,
                                    ilkGoruntuOznitenikleri, )
12:    ikinciGoruntuTanimLayicilari ← OznitelikleriTanimla(
                                       oznitelikIslemcisi,
                                       ikinciGoruntu,
                                       ikinciGoruntuOznitenikleri, )
13:    return {ilkGoruntuTanimLayicilari,
               ikinciGoruntuTanimLayicilari}
14: end Fonksiyon
```

Gerçekleştirilen filtreleme işlemi sonrasında, ikinci görüntü üzerinde tespit edilmiş olan bütün öznitelikler arasından, birinci görüntü üzerinde bulunan filtrelenmiş öznitelikler için eşleşenler tespit edilmelidir (Bkz. Algoritma 2.15).

Algoritma 2.15: Oznitelik Eşleştirmesi

```
1: Fonksiyon OznitelikleriEslestir( ilkGoruntuTanimLayicilari,
   ikinciGoruntuTanimLayicilari )
2:     eslesmeListesi ← Eslestir( ilkGoruntuTanimLayicilari,
                                  ikinciGoruntuTanimLayicilari )
3:     { minUzaklik, maxUzaklik } ← EslesmeMesafeSinirlariniBul(
                                    eslesmeListesi )
4:     eslesmeUzakligiUstSiniri ← {3*minUzaklik}
5:     uygunEslesmeListesi ← {}
6:     for all eslesme in eslesmeListesi do
7:         if UzakligiAl(eslesme) ≤ eslesmeUzakligiUstSiniri then
8:             uygunEslesmeListesi ← { uygunEslesmeListesi, eslesme
                                     }
9:         end if
10:    end for
11:    return uygunEslesmeListesi
12: end Fonksiyon
```

İkinci görüntü üzerinde tespit edilen eşleşen öznitelikler dışında kalanlar dikkate alınmayacak şekilde iki görüntü arasında dönüşüm matrisi elde edilir (Bkz. Algoritma 2.16). Bu dönüşüm matrisi ikinci görüntünün birinci görüntüye göre nasıl değiştiği

bilgisini taşımaktadır [46].

Algoritma 2.16: İki Görüntü Arasındaki Dönüşüm Matrisinin Bulunması

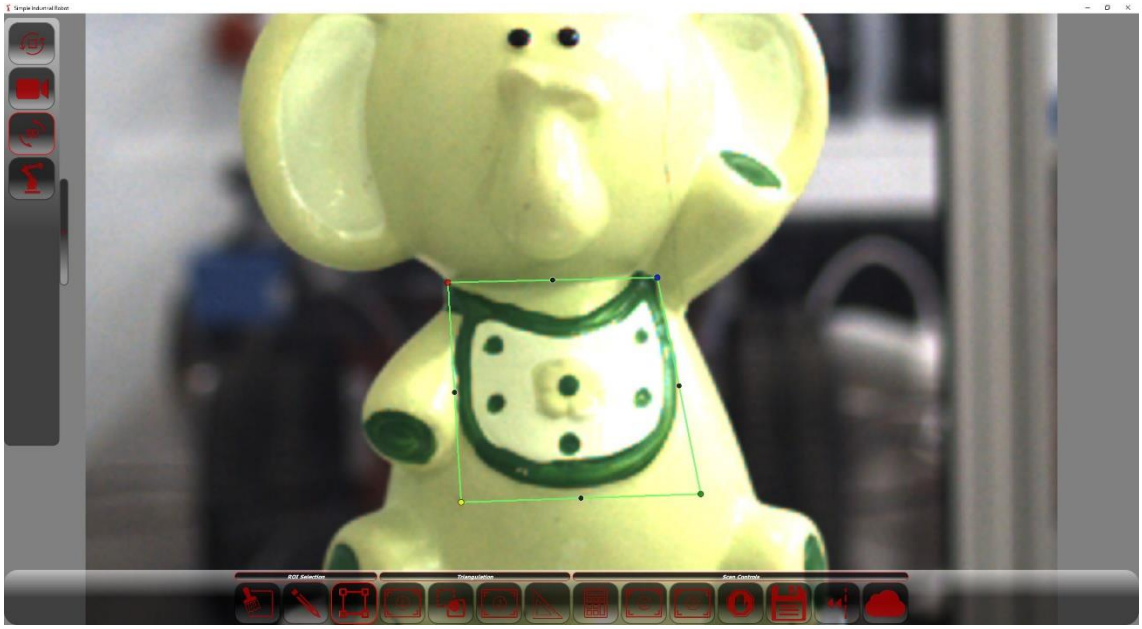
```
1: Fonksiyon DonusumMatrisiniBul( uygunEslesmeListesi )
2:     ilkGorEslesmeNoktalari ← {}
3:     ikinciGorEslesmeNoktalari ← {}
4:     for all eslesme in uygunEslesmeListesi do
5:         ilkGoruntuKonumu ← IlkGoruntuKonumunuAl(eslesme)
6:         ikinciGoruntuKonumu ← IkinciGoruntuKonumunuAl(eslesme
7:         ilkGorEslesmeNoktalari ← { ilkGorEslesmeNoktalari,
                                      ilkGoruntuKonumu }
8:         ikinciGorEslesmeNoktalari ← {
                                      ikinciGorEslesmeNoktalari,
                                      ikinciGoruntuKonumu }
9:     end for
10:    donusumMatrisi ← FindHomography(ilkGorEslesmeNoktalari,
                         ikinciGorEslesmeNoktalari, RANSAC)
11:    return donusumMatrisi
12: end Fonksiyon
```

Elde edilen dönüşüm matrisinin birinci görüntü üzerinde kullanıcı tarafından belirlenen çevreleyen dörtgenin bütün köşelerine uygulanması ile, bu dörtgenin ikinci görüntü üzerinde oluşturulmasını sağlayacak köşe koordinatları tespit edilmiş olacaktır (Bkz. Algoritma 2.17). Görsel 2.17'de gösterildiği üzere operatör tarafından belirlenmiş olan çevreleyen dörtgenin, ikinci görüntü üzerinde oluşturulması ile elde edilen görüntü Görsel 2.18'de sergilenmektedir.

Algoritma 2.17: Donusum Matrisinin Uygulanması

```
1: Fonksiyon DonusumUygula( dortgenKoseleri , donusumMatrisi )
2:     homDortgenKoseleri ← {HomojeneDonustur(dortgenKoseleri)}
3:     ikinciGorDortgenKoseleri ← {}
4:     for all homKose in homDortgenKoseleri do
5:         homDonusturulenKose ← donusumMatrisi * homKose
6:         donusturulenKose ← HomojenSil(homDonusturulenKose)
7:         ikinciGorDortgenKoseleri ← { ikinciGorDortgenKoseleri,
                                        donusturulenKose }
8:     end for
9:     return ikinciGorDortgenKoseleri;
10: end Fonksiyon
```

Tespit edilen bu köşe koordinatlarında oluşacak bir hatanın, elde edilecek üç boyutlu noktanın hatalı tespit edilmesine sebep olacağından, elde edilen dörtgen operatöre kullanıcı arayüzü üzerinde görüntülenir. Görüntülenen dörtgen, operatörün belirlediği dörtgen ile uyuşmuyor ise yine kullanıcı arayüzü üzerinden operatörün bu dörtgenin köşe koordinatlarını değiştirmesine izin verilir ve bir sonraki adım için operatör onayı beklenir.



Görsel 2.18. *İkinci Görüntü Üzerinde Oluşturulmuş Çevreleyen Dörtgen*

2.2.3.3. *Çevreleyen Dörtgenini Tanımlayan Köşelerin Kamera Kalibrasyon Nesnesi Koordinat Sistemindeki Karşılıklarının Bulunması*

Operatör tarafından birinci görüntü üzerinde tanımlanmış olan çevreleyen dörtgen ve bu dörtgenin eş karşılığı olan ve sistem tarafından tespit edilen ikinci görüntü üzerinde konumlandırışmış olan çevreleyen dörtgen birbirleri dünya koordinat sisteminde örtüşmektedir. Bir başka deyişle her iki görüntü üzerinde bulunan ve çevreleyen dörtgeni tanımlayan dört köşeye ait noktalar Şekil 2.14'da olduğu gibi gerçek dünyada gerçek bir üç boyutlu noktada karşılığı bulunmaktadır. Bu bölümde bu noktanın nerede konumlandığının tespiti ele alınmaktadır.

Kullanıcı (operatör) tarafından kullanıcı arayüzü kullanılarak, kamera görüntü düzleminde tanımlanmış olan çevreleyen dörtgenin köşelerini temsil eden dört noktanın ikinci görüntü üzerindeki karşılıkları olan noktalar kullanılarak bu her bir noktaya karşılık gelen üç boyutlu nokta üçgenleştirme yöntemi kullanılarak bulunabilir [46, 56]. Bu yöntemin uygulanabilmesi için, üçgenleştirme yöntemi ile gerçek dünyadaki üç boyutlu noktanın her iki görüntü üzerindeki konumlarının ve bu iki görüntünün elde edilebilmesi için kullanılan dönüşüm matrislerinin (üç boyutlu dünya koordinat sistemindeki noktaları, iki boyutlu kamera görüntü düzlemi koordinat sistemine taşıyan matris) bilinmesi gerekmektedir. Çevreleyen dörtgenin ilk görüntü üzerindeki köşe koordinatları ve ikinci

görüntüdeki eş köşe koordinatları aranan üç boyutlu noktaların görüntü düzlemlerindeki karşılıklarını vermektedir. İhtiyaç duyulan dönüşüm matrisleri ise, kalibrasyon sürecinde gerçekleştirilen kamera kalibrasyonu adımında elde edilmektedir. Elde edilen kamera kalibrasyon matrisi, uygulama sırasında alınan ilk görüntü için (kullanıcının ilgilenilen bölgeyi belirlediği görüntü) gerekli olan dönüşüm matrisine karşılık gelmektedir. İkinci matrisin elde edilebilmesi için ise, kalibrasyon nesnesini taşıyan döner tablanın, ikinci görüntüyü elde etmek için kullanılan ön tanımlı olan dönüş açısı kadar döndürülmesi ve kalibrasyon nesnesinin bu pozisyonu için kamera kalibrasyonunun tekrarlanması gerekmektedir. Elde edilen yeni matris, ilgilenilen bölgenin dünya koordinat sistemindeki üç boyutlu karşılığının bulunabilmesi için ihtiyaç duyulan ikinci matris olacaktır. Bu matrisin elde edilmesi, kamera kalibrasyon sürecinde bir kez yapılması gereken bir işlemdir ve aynı süreçte bahsedilen XML dosya formatında saklanmaktadır.

Üçgenleştirme işlemi için gerekli olan parametreler ve bu parametrelerin geometrik anlamları Şekil 2.14'da verilmiştir. Burada $\mathbf{w}$ dünya koordinat sisteminde karşılığı aranan noktaya, $\mathbf{u_1}$ ve $\mathbf{u_2}$ birinci ve ikinci görüntü düzlem koordinatlarındaki $\mathbf{w}$ noktasının homojen görüntü koordinat sitemindeki karşılıklarına, $\mathbf{M_1}$ ve $\mathbf{M_2}$ ise bu görüntü koordinat düzlemlerine ait dönüşüm matrislerine karşılık gelmektedir. Her iki görüntü ve her iki görüntü üzerinde konumlandırılmış noktalar ideal koşullarda Denklem (2.8)'de verilen ilişki ile elde edilmektedir ($\boldsymbol{\lambda \mathbf{u} = \mathbf{M} \mathbf{w}}$). Buna göre aşağıdaki ilişkiyi oluşturmak mümkün hale gelmektedir.

$$\mathbf{u} \times \mathbf{M}\mathbf{w} = \mathbf{0} \tag{2.27}$$

Burada; $\mathbf{u}$ kamera görüntü düzlemindeki homojen noktaya, $\mathbf{M}$ görüntü koordinat sistemi için tanımlı olan dönüşüm matrisine, $\mathbf{w}$ ise koordinatları aranan üç boyutlu homojen noktaya ve $\mathbf{0}$ üç boyutlu sıfır vektörüne karşılık gelmektedir.

Denklem (2.27), iki ayrı görüntü düzleminde aynı $\mathbf{w}$ noktası için tespit edilmiş iki homojen nokta için yazılır ise aşağıdaki denklem kümesi elde edilir.

$$\mathbf{u_1} \times \mathbf{M_1}\mathbf{w} = \mathbf{0} \tag{2.28}$$

$$\mathbf{u_2} \times \mathbf{M_2}\mathbf{w} = \mathbf{0} \tag{2.29}$$

Burada; $\mathbf{u_1}$ ve $\mathbf{u_2}$ iki ayrı görüntüde konumlandırışmış homojen noktalara $\mathbf{M_1}$ ve $\mathbf{M_2}$ bu görüntüler için elde edilen dönüşüm matrislerine ve $\mathbf{w}$ dünya koordinat sisteminde

tanımlı homojen üç boyutlu noktaya karşılık gelmektedir.

Denklem (2.28)'de verilmiş olan vektörel çarpım işlemi açık halde yeniden düzenlendiğinde:

$$\begin{bmatrix} u_1^x \\ u_1^y \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{m_1^1 w} \\ \mathbf{m_1^2 w} \\ \mathbf{m_1^3 w} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.30}$$

Burada; u_1^x ve u_1^y, $\mathbf{u_1}$ homojen noktasının x ve y koordinatlarına, $\mathbf{m_1^i}$ ise $\mathbf{M_1}$ dönüşüm matrisinin i'inci satırına karşılık gelmektedir.

Denklem (2.30)'da verilmiş olan vektörel çarpım işlemi gerçekleştirildiğinde ise aşağıdaki lineer denklem kümesi elde edilmektedir.

$$u_1^x(\mathbf{m_1^3 w}) - (\mathbf{m_1^1 w}) = 0 \tag{2.31}$$

$$u_1^y(\mathbf{m_1^3 w}) - (\mathbf{m_1^2 w}) = 0 \tag{2.32}$$

$$u_1^x(\mathbf{m_1^2 w}) - u_1^y(\mathbf{m_1^1 w}) = 0 \tag{2.33}$$

Bu denklem sisteminde (2.31) ve (2.32) lineer olarak birbirinden bağımsızdır. Aynı adımlar $\mathbf{u_2}$ homojen noktası ve $\mathbf{M_2}$ dönüşüm matrisi için uygulandığında dört denklem ve dört bilinmeyene sahip bir lineer sistem oluşacaktır.

$$\mathbf{Aw} = \mathbf{0} \tag{2.34}$$

$$\mathbf{A} = \begin{bmatrix} u_1^x \mathbf{m_1^3} - \mathbf{m_1^1} \\ u_1^y \mathbf{m_1^3} - \mathbf{m_1^2} \\ u_2^x \mathbf{m_2^3} - \mathbf{m_2^1} \\ u_2^y \mathbf{m_2^3} - \mathbf{m_2^2} \end{bmatrix} \tag{2.35}$$

Burada $\mathbf{A}$ matrisi elde edilen lineer sistemin çarpan matrisi $\mathbf{w}$ ise aynı sistemin bilinmeyen vektörüdür. Bu sistemin çözümü ile elde edilecek dört boyutlu vektör son elemanı ile normalizasyon işlemine sokulduğunda (homojen noktanın son elemanını birin haline getirme), bu vektörün ilk üç elemanı aranan üç boyutlu noktanın elde edilmesini sağlayacaktır.

Elde edilen üç boyutlu nokta birinci görüntü için gerçekleştirilen kamera kalibrasyon sürecinde kullanılan kalibrasyon nesnesi için tanımlanan koordinat sisteminde temsil edilmektedir.

Denklem (2.27) ve (2.35) arasında tanımlanan süreç, çevreleyen dörtgeni temsil

eden köşe koordinatlarının tamamı için uygulandığında ise, bu noktaların üç boyutlu sistemdeki karşılıkları elde edilmiş olacaktır.

2.2.3.4. *Çevreleyen Dörtgenini Tanımlayan Köşelerin Döner Tabla Koordinat Sistemindeki Karşılıklarının Bulunması*

Çevreleyen dörtgeni tanımlayan köşelerin bir önceki bölümde anlatılan üçgenleştirme yöntemi ile tespit edilmesinin ardından elde edilen üç boyutlu noktalar, kamera kalibrasyon sürecinde kullanılan kalibrasyon nesnesi tarafından tanımlanan koordinat sisteminde tanımlıdır. Fakat bu noktaların döner tabla koordinat sisteminde tanımlı hale getirilmesi gerekmektedir. Çünkü elde edilmiş olan bu noktalar kullanılarak, endüstriyel robot ucunda bulunan lazer profil sensör yardımıyla ilgilenen bölgenin üç boyutlu modeli oluşturulacaktır. Programlama süreci elde edilecek olan üç boyutlu modele bağımlıdır.

Çevreleyen dörtgene ait köşeler her ne kadar kalibrasyon nesnesi tarafından tanımlanan koordinat sisteminde tanımlı olacak şekilde elde edilmiş olsalar da kullanıcı tarafından gerçekleştirilen ilgilenen bölgenin kamera görüş alanına sokulması sürecinde, döner tablanın açısı değiştirilmiştir. Kalibrasyon sürecinde kullanılan kalibrasyon nesnesinin tanımladığı koordinat sistemi ise döner tablanın 0 pozisyonu için tanımlıdır. Bu nedenle elde edilmiş olan noktalar işlem görecek nesnenin bu kullanıcının gerçekleştirdiği dönüş miktarı kadar döndürülmüş halleridir.

Açı uyumsuzluğunun ortadan kaldırılması ve bir önceki adımda elde edilen noktaların, kalibrasyon nesnesinin tanımladığı gerçek koordinat sistemine dönüştürülmesi gerekmektedir. Kalibrasyon nesnesinin tanımladığı koordinat sisteminin z ekseni ile döner tablanın döner ekseni olan z ekseninin çakışık olması (kalibrasyon nesnesini bu şartı sağlayacak şekilde döner tablanın üzerine oturtulduğu varsayılmaktadır) sebebi ile kullanıcı tarafından gerçekleştirilmiş olan döndürme etkisini ortadan kaldıracak bir dönüşüm uygulanması gerekmektedir. Eğer kullanıcı tarafından gerçekleştirilen döndürme işleminden dolayı döner tablanın ilgili andaki açısı θ olacak şekilde değiştirilmiş ise, bir önceki adımda elde edilen noktalar z ekseni boyunca θ kadar döndürülmüş durumdadır:

$$T_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.36)$$

Burada; $T_z(\theta)$ z ekseni boyunca θ açısı ile döndürme işlemini gerçekleştirilen dönüşüm matrisine karşılık gelmektedir.

Eğer bir önceki adımda elde edilmiş olan üç boyutlu noktalar (çevreleyen dörtgenin köşeleri), kullanıcı tarafından gerçekleştirilmiş olan döner tabla dönüşünün etkisi ile Denklem *(2.36)*'da verilen dönüşüm matrisinin etkisi altında ise, bu noktalar homojen koordinat sisteminde şu şekilde tanımlanabilir.

$$\mathbf{w}_{\boldsymbol{\theta}}^{\mathbf{i}} = T_z(\theta)\, \mathbf{w}_{\mathbf{CALIB}}^{\mathbf{i}}, \mathbf{i} = 0,1,2,3 \quad (2.37)$$

Burada $\mathbf{w}_{\boldsymbol{\theta}}^{\mathbf{i}}$, bir önceki adımda elde edilen ve Denklem (2.36)'da verilen dönüşüm matrisinin etkisinde olan i'inci noktaya, $\mathbf{w}_{\mathbf{CALIB}}^{\mathbf{i}}$ ise aynı noktaların kalibrasyon nesnesi tarafından tanımlanan koordinat sistemindeki aranan gerçek değerlerine karşılık gelmektedir.

Elde edilmek istenen asıl noktaların $\mathbf{w}_{\mathbf{CALIB}}^{\mathbf{i}}$ olması ve elde bilinen noktaların z ekseni boyunca bilinen (döner tabla açısı) bir açıyla uygulanmış olan dönüşümün etkisi altında olması sebebi ile, aranan bu noktalar şu şekilde elde edilebilir.

$$T_z(\theta)^{-1}\mathbf{w}_{\boldsymbol{\theta}}^{\mathbf{i}} = T_z(\theta)^{-1}T_z(\theta)\, \mathbf{w}_{\mathbf{CALIB}}^{\mathbf{i}}, \mathbf{i} = 0,1,2,3 \quad (2.38)$$

Burada; $T_z(\theta)^{-1}$, $T_z(\theta)$ matrisinin tersini ifade etmektedir. $T_z(\theta)$ matrisi bir döndürme matrisini temsil ediyor olması sebebi ile aşağıdaki eşitlik elde edilebilir.

$$T_z(\theta)^{-1} = T_z(-\theta) \quad (2.39)$$

Denklem (2.38) kullanılarak,

$$T_z(-\theta)\mathbf{w}_{\boldsymbol{\theta}}^{\mathbf{i}} = T_z(-\theta)T_z(\theta)\, \mathbf{w}_{\mathbf{CALIB}}^{\mathbf{i}}, \mathbf{i} = 0,1,2,3 \quad (2.40)$$

Burada $T_z(-\theta)T_z(\theta) = I$ ise:

$$T_z(-\theta)\mathbf{w}_{\boldsymbol{\theta}}^{\mathbf{i}} = \mathbf{w}_{\mathbf{CALIB}}^{\mathbf{i}}, \mathbf{i} = 0,1,2,3 \quad (2.41)$$

Kamera kalibrasyon nesnesi koordinat sisteminde elde edilen yeni noktalar, kullanıcı tarafından gerçekleştirilen döner tabla dönüşünün etkisinden temizlenmiş haldedir. Bu noktalar ve kamera kalibrasyon adımı sırasında tanımlanmış olan döner tabla koordinat ekseni ve kalibrasyon nesnesi koordinat ekseni arasındaki ilişki (T_{CALIB}^{RT0})

kullanılarak, ilgilenilen bölgeyi temsil eden çevreleyen dikdörtgenin, döner tabla koordinat sistemindeki karşılığı olan noktalar bulunmaktadır.

$$\mathbf{w_{RT0}^{i}} = T_{CALIB}^{RT0}\mathbf{w_{CALIB}^{i}}\,,\mathbf{i} = 0,1,2,3 \tag{2.42}$$

Burada; $\mathbf{w_{RT0}^{i}}$, kullanıcı tarafından belirlenmiş olan çevreleyen dörtgenin köşe noktalarının döner tabla koordinat sistemindeki karşılıkları olan homojen üç boyutlu noktaları temsil etmektedir.

2.3. İlgilenilen Bölgenin Üç Boyutlu Modelinin Elde Edilmesi

Üç boyutlu modelin üretilmesi aşaması endüstriyel işlem gerçekleştirme ve bu işlem için programlanacak robotun gerçekleştireceği işlem için gerekli olan hareket planının çıkartılması için ihtiyaç duyulan bir adımdır. Bu süreçte izlenecek yol temel olarak bir önceki adımda kullanıcı arayüzü ve operatörün etkileşimi ile elde edilen ve bu tez çalışması kapsamında ilgilenilen bölge olarak tanımlanan alanın, endüstriyel robotun bağlantı ucuna bağlı bulunan lazer profil sensör yardımı ile taramasının gerçekleştirilerek elde edilecek nokta bulunun oluşturulmasıdır. Bu nokta bulutu işlem görecek bölgenin oldukça detaylı bir modelini içermekte olup, elde edilen veri endüstriyel robotun programlanması sürecinde temel kaynak olarak kullanılacaktır.

Endüstriyel robot koluna bağlı olan lazer profil sensör kullanılarak gerçekleştirilecek olan bu tarama işleminin yürütülebilmesi, tarama için kullanılacak olan lazer sensörün, kullanıcı tarafından belirlenmiş ve döner tabla koordinat ekseninde karşılığı elde edilmiş olan bölge için uygun ölçüm aralığı içerisinde kalacak şekilde robot tarafından hareket ettirilmesine bağlıdır. Bu durum, tanımlanan süreç için önce robot hareket planının oluşturulması gereksinimini beraberinde getirmektedir.

2.3.1. Robot Hareket Planının Oluşturulması

Lazer profil sensör ile gerçekleştirilecek olan tarama işlemi, bir önceki adımda elde edilmiş olan ilgilenilen bölgeyi tanımlayan, döner tabla koordinat sisteminde tanımlanmış olan üç boyutlu noktalar kullanılarak, ideal tarama işlemi için robot hareket planı çıkarılması gerekmektedir. Burada gerçekleştirilecek tarama işlemi, ilgilenilen bölgeyi

tanımlayan dört nokta için oluşturulan ideal düzlemsel bölgenin oluşturulup, bu bölgenin robot için erişilebilir bir yönelime sokulması ile mümkün olacaktır.

2.3.1.1. *Tarama Düzleminin Oluşturulması*

Kullanıcı tarafından belirlenmiş olan ilgilenilen bölge ve bu bölgeyi tanımlayan köşe noktaları (dört köşe), üç boyutlu uzayda, ideal bir düzlemsel bölge tanımlamama potansiyeli taşımaktadır. Bu durum, işlem görecek nesnenin üzerinde tanımlanan bölgenin düzlemsel bir bölge olmamasından kaynaklanmaktadır. Bu nedenle hesaplanmış olan bu noktalar için ideal bir düzlemsel bölge tanımlanıp, bu noktaların bu düzlemsel bölge üzerinde kalacak şekilde yeniden güncellenmesi gerekmektedir. Üç boyutlu uzayda herhangi bir düzlem şu şekilde tanımlanmaktadır.

$$ax + by + cz + d = 0 \tag{2.43}$$

Burada; a, b, c, düzlemin normal vektörünün elemanlarına ve x, y, z düzlem üzerinde tanımlı bir noktanın koordinatlarına karşılık gelmektedir. d ise düzlem denklemini sağlayan öteleme bileşenidir. Eğer bu d bileşenini 1 kabul edilebilir, fakat bu durumda normal vektörünün boyu birim uzunlukta olmayacaktır. Bu durumda Denklem (2.42), şu şekilde yeniden düzenlenebilir.

$$\mathbf{x} \cdot \mathbf{n} = 1 \tag{2.44}$$

Burada; **n** düzlemin normal vektörünü, **x** düzlem üzerinde bulunan bir noktayı ifade etmektedir. Burada dikkat edilmesi gereken kriter, düzlemin normal vektörünün birim uzunlukta olma zorunluluğu bulunmamasıdır.

Eğer ilgilenilen bölgeyi tanımlayan dört köşe noktası bir düzlem üzerine oturtulmak isteniyor ise, bu noktaların Denklem *(2.44)*'ü sağlaması gerekmektedir. Fakat bu noktaların üç boyutlu uzayda bir düzlem üzerinde konumlanmadığı ortadadır. Bu durumda bu noktalara en yakın uzaklıkta bulunan bir düzlem tespit edilmesi gerekmektedir. Bu noktalara en yakın uzaklıkta bulunan düzlemin bir normal vektörü bulunması gerekmektedir.

$$
\begin{aligned}
x_1^x n_x + x_1^y n_y + x_1^z n_z &= 1 \\
x_2^x n_x + x_2^y n_y + x_2^z n_z &= 1 \\
x_3^x n_x + x_3^y n_y + x_3^z n_z &= 1 \\
x_4^x n_x + x_4^y n_y + x_4^z n_z &= 1
\end{aligned}
\tag{2.45}
$$

Burada; x_i^x, x_i^y, x_i^z i'inci noktanın x, y ve z bileşenlerine n_x, n_y, n_z , bu dört noktaya en yakın olan düzlemin normal vektörünün x, y ve z bileşenlerine karşılık gelmektedir. Bu lineer sistem matris formunda şu şekilde yazılabilir.

$$
\begin{bmatrix} x_1^x & x_1^y & x_1^z \\ x_2^x & x_2^y & x_2^z \\ x_3^x & x_3^y & x_3^z \\ x_4^x & x_4^y & x_4^z \end{bmatrix} \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{2.46}
$$

Denklem (2.46)'da elde edilen lineer sistemin çözümü ile elde ($\mathbf{An} = \mathbf{1}$) edilecek normal vektör, ilgilenen bölgeyi tanımlayan dört noktaya en yakın uzaklıkta bulunan düzlemin normal vektörünü verecektir.

İlgilenen bölgeyi tanımlayan dört nokta bulunan normal vektör ile temsil edilen düzlem üzerinde olmadığı için, düzlem denklemini sağlamamaktadır. Bu nedenle bu noktaların düzlem denklemini sağlayacak şekilde güncellenmesi gerekmektedir. Burada izlenecek yol ise her bir noktanın ilgili düzleme en yakın olan noktaya eşitlenmesidir. Bunun öncelikli olarak Denklem (2.46)'da elde edilen normal vektörün birim vektörünün bulunması gerekmektedir.

$$
\mathbf{n_u} = \frac{\mathbf{n}}{|\mathbf{n}|} \tag{2.47}
$$

Burada; $\mathbf{n_u}$ birim normal vektöre, $|\mathbf{n}|$ ise normal vektörün büyüklüğüne karşılık gelmektedir.

Her bir noktanın düzeleme olanı uzaklığı aşağıdaki gibi hesaplanabilir.

$$
\begin{aligned}
\ell_i &= \frac{\mathbf{n} \cdot \mathbf{x_i} - 1}{|\mathbf{n}|} \\
\mathbf{x_i} &= \mathbf{x_i} - \ell_i \mathbf{n_u}
\end{aligned}
\tag{2.48}
$$

Burada; ℓ_i, i'inci noktanın düzleme olan uzaklığına karşılık gelmektedir.

Elde edilen yeni noktalar, $\mathbf{x_i}$, $i = 1,2,3,4$, tarama yapılacak düzlemi tanımlayan noktalardır.

2.3.1.2. *Taranacak Düzleminin Robot Erişimi İçin Döndürülmesi*

Tarama yapılacak düzlemin, döner tabla koordinat sistemindeki karşılığının tespit edilmesi, tarama işlemini gerçekleştirmek için tek başına yeterli değildir. Bu noktalar, endüstriyel robot kolunun erişemeyeceği bir bölgede (örneğin, cismin robot koluna uzak olan kenarında) konumlanmış olabilir. Bu nedenle, tarama yapılacak bölgenin robotun erişebileceği en ideal yönelime göre döndürülmesi gerekmektedir. Döndürme işlemi, döner tablanın dönüşü ile, taranacak bölgenin endüstriyel robota yaklaştırılmasını sağlamaktadır. Bu nedenle, ilk olarak bu döndürme işlemi için gerekli olan açının tespit edilmesi gerekmektedir.

Endüstriyel robotun gerçekleştireceği düzlemsel tarama işlemi için tarama işlemi gerçekleştirilen düzlemin merkez yöneliminin merkez noktasına ait pozisyon vektörü ($\mathbf{x_c}$) ile sistem kurulum ve kalibrasyonu sırasında belirlenmiş olan ideal tarama yönelim vektörü ($\mathbf{x_{is}}$) ile arasındaki açının hesaplanın, döner tablanın bu açı miktarı kadar döndürülmesi gerekmektedir.

$$\mathbf{x_c} = \frac{\mathbf{x_1} + \mathbf{x_2} + \mathbf{x_3} + \mathbf{x_4}}{4} \qquad x_c^z = 0; \tag{2.49}$$

$$\theta = \cos^{-1}\left(\frac{\mathbf{x_{is}} \cdot \mathbf{x_c}}{|\mathbf{x_{is}}| \cdot |\mathbf{x_c}|}\right) \tag{2.50}$$

$$\mathbf{\Omega} = \mathbf{x_c} \times \mathbf{x_{is}} \tag{2.51}$$

$$if(\Omega_z < 0)\ \theta = -\theta \tag{2.52}$$

Burada; $\mathbf{x_c}$, tarama işlemi gerçekleştirilecek düzlemin merkez noktasına ait pozisyon vektörü için, döner tabla koordinat sisteminin xy düzlemine iz düşümüne, $\mathbf{x_{is}}$ ise yine z bileşeni 0 olan döner tabla koordinat ekseninde tanımlı ideal tarama yönelim vektörüne, θ belirlenen tarama düzleminin ideal tarama yönelimine döndürülebilmesi için ihtiyaç duyulan döner tabla açısına karşılık gelmektedir. $\mathbf{\Omega}$, tarama düzleminin ideal tarama konumuna getirmek için döndürme gerçekleştirilecek, döndürme ekseni temsil etmektedir. Bu eksenin değeri, her iki vektör sadece xy düzleminde tanımlı olduğu için z ekseni boyunca çıkacaktır. Önemli olan bu döndürme ekseninin işaretidir. Bu işarete

bağımlı olarak döner tablanın saat yönündeki yoksa, saat yönünün tersinemi döndürülmesi gerektiği belirlenmektedir.

Taranacak bölgenin ideal tarama bölgesine getirilmesi için gerçekleştirilen döner tabla dönüşü, taranacak bölgeyi tanımlayan noktaların, bu dönüş açısına bağımlı olarak değiştirilmesi gerekliliğini doğurmaktadır.

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.53}$$

$$\mathbf{x_i^s} = R_z(\theta)\mathbf{x_i}\ ,\ i=1,2,3,4 \tag{2.54}$$

Burada; $R_z(\theta)$ döner tabla dönme hareketinden dolayı taranacak bölgeyi temsil eden noktalara uygulanması gereken döndürme matrisine, $\mathbf{x_i^s}$ ise i'inci noktanın döner tablanın tarama pozisyonuna getirildiğinde, alacağı yeni pozisyon değerine karşılık gelmektedir.

Tarama noktalarının döner tablanın tarama pozisyonuna getirilmesi ve bu hareketten dolayı oluşan yeni tarama noktaları ($\mathbf{x_i^s}\ , i = 1,2,3,4$) kullanılarak, tarama işlemi yapılacak yüzeyin normal vektörünün hesaplanması gerekmektedir. Bu normal vektörü aynı zamanda lazer profil sensörün oryantasyonunun hesaplanması sırasında ihtiyaç duyulacak olan doğrultu vektörü olacaktır.

$$\mathbf{n^s} = (\mathbf{x_1^s} - \mathbf{x_2^s}) \times (\mathbf{x_3^s} - \mathbf{x_2^s})$$

$$\mathbf{n_u^s} = \frac{\mathbf{n^s}}{|\mathbf{n^s}|} \tag{2.55}$$

Burada; $\mathbf{n^s}$ ideal tarama noktalarının oluşturduğu düzlemin normal vektörünü, $\mathbf{n_u^s}$ ise aynı düzlemin birim normal vektörünü ifade etmektedir.

Gerçekleştirilecek olan tarama işleminin, lazer profil sensörünün, tarama düzlemini ortalayacak ve yukarıdan aşağıya doğrusal hareket yapacak şekilde gerçekleştirileceği göz önünde bulundurulduğunda, tarama işlemi için bir en üst nokta ($\boldsymbol{x_u^s}$) ve en alt noktaya ($\boldsymbol{x_l^s}$) ihtiyaç duyulmaktadır. Sensörün bu iki nokta arasında oluşturulan pozisyon vektörünü takip edecek şekilde hareket etmesi beklenmektedir Bu hareket ekseni lazer sensörüne ait koordinat sisteminin y eksenini oluşturmaktadır.

$$\mathbf{x_u^s} = \frac{\mathbf{x_1} + \mathbf{x_2}}{2}\ , \qquad \mathbf{x_l^s} = \frac{\mathbf{x_3} + \mathbf{x_4}}{2} \tag{2.56}$$

Tarama yüzeyinin normal vektörü ve tarama işleminin en üst noktası ile en alt

noktası arasındaki pozisyon vektörleri kullanılarak, sensör oryantasyonunu döner tabla koordinat ekseninde hesaplamak mümkün hale gelmektedir.

$$\begin{aligned} \mathbf{n}_{\mathbf{z}}^{\mathbf{RT0}} &= -\mathbf{n}_{\mathbf{u}}^{\mathbf{s}} \\ \mathbf{n}_{\mathbf{y}}^{\mathbf{RT0}} &= \frac{\mathbf{x}_{\mathbf{u}}^{\mathbf{s}} - \mathbf{x}_{\mathbf{l}}^{\mathbf{s}}}{\left|\mathbf{x}_{\mathbf{u}}^{\mathbf{s}} - \mathbf{x}_{\mathbf{l}}^{\mathbf{s}}\right|} \\ \mathbf{n}_{\mathbf{x}}^{\mathbf{RT0}} &= \mathbf{n}_{\mathbf{y}}^{\mathbf{RT}} \times \mathbf{n}_{\mathbf{z}}^{\mathbf{RT}} \\ \boldsymbol{R}_{\boldsymbol{TOOL}}^{\boldsymbol{RT0}} &= \left[\mathbf{n}_{\mathbf{x}}^{\mathbf{RT}} \quad \mathbf{n}_{\mathbf{y}}^{\mathbf{RT}} \quad \mathbf{n}_{\mathbf{z}}^{\mathbf{RT}}\right] \end{aligned} \tag{2.57}$$

Burada; $\mathbf{n}_{\mathbf{x}}^{\mathbf{RT0}}, \mathbf{n}_{\mathbf{y}}^{\mathbf{RT0}}, \mathbf{n}_{\mathbf{z}}^{\mathbf{RT0}}$ sırası ile, lazer sensör koordinat sisteminin x, y, z eksenlerini göstermektedir. Ayrıca bu vektörler, lazer sensör oryantasyon matrisi olan $\mathbf{R}_{\boldsymbol{TOOL}}^{\mathbf{RT0}}$'nin sütunlarını oluşturmaktadır.

Hesaplanan tarama pozisyonları ($\mathbf{x}_{\mathbf{u}}^{\mathbf{s}}, \mathbf{x}_{\mathbf{l}}^{\mathbf{s}}$) ve oryantasyon matrisi ($\boldsymbol{R}_{\boldsymbol{TOOL}}^{\boldsymbol{RT0}}$) döner tabla koordinat sistemine göre tanımlanmaktadır. Fakat endüstriyel robot kolunun hareket edebilmesi için hareket pozisyonlarının oryantasyonlarının endüstriyel robot taban koordinat sisteminde tanımlanması gerekmektedir. Ayrıca, tarama işlemi için tanımlanan en üst nokta ve en alt noktada, tarama yüzeyinin üzerinde konumlanmış birer noktadır. Öncelikli olarak bu noktaların, sensörün ideal tarama mesafesinde tutulabilmesi için güncellenmesi gerekmektedir. Sensör ideal tarama mesafesi, sensörün tarama sınırlarının orta noktası olarak tanımlanır ise (ρ)

$$\boldsymbol{P}_{\boldsymbol{TOOL}_{\boldsymbol{u}}}^{\boldsymbol{RT0}} = \mathbf{x}_{\mathbf{u}}^{\mathbf{s}} + \rho\mathbf{n}_{\mathbf{u}}^{\mathbf{s}}, \qquad \boldsymbol{P}_{\boldsymbol{TOOL}_{\boldsymbol{l}}}^{\boldsymbol{RT0}} = \mathbf{x}_{\boldsymbol{l}}^{\mathbf{s}} + \rho\mathbf{n}_{\mathbf{u}}^{\mathbf{s}} \tag{2.58}$$

$$\boldsymbol{R}_{\boldsymbol{TCP}}^{\boldsymbol{B}} = \boldsymbol{R}_{\boldsymbol{RT0}}^{\boldsymbol{B}} \boldsymbol{R}_{\boldsymbol{TOOL}}^{\boldsymbol{RT0}} \boldsymbol{R}_{\boldsymbol{TCP}}^{\boldsymbol{TOOL}} \tag{2.59}$$

$$\boldsymbol{P}_{\boldsymbol{TCP}_{\boldsymbol{u}}}^{\boldsymbol{B}} = -\boldsymbol{R}_{\boldsymbol{TCP}}^{\boldsymbol{B}} \boldsymbol{P}_{\boldsymbol{TOOL}}^{\boldsymbol{TCP}} + \boldsymbol{R}_{\boldsymbol{RT0}}^{\boldsymbol{B}} (\boldsymbol{P}_{\boldsymbol{TOOL}_{\boldsymbol{u}}}^{\boldsymbol{RT0}} - \boldsymbol{P}_{\boldsymbol{BASE}}^{\boldsymbol{RT0}}) \tag{2.60}$$

$$\boldsymbol{P}_{\boldsymbol{TCP}_{\boldsymbol{l}}}^{\boldsymbol{B}} = -\boldsymbol{R}_{\boldsymbol{TCP}}^{\boldsymbol{B}} \boldsymbol{P}_{\boldsymbol{TOOL}}^{\boldsymbol{TCP}} + \boldsymbol{R}_{\boldsymbol{RT0}}^{\boldsymbol{B}} (\boldsymbol{P}_{\boldsymbol{TOOL}_{\boldsymbol{u}}}^{\boldsymbol{RT0}} - \boldsymbol{P}_{\boldsymbol{BASE}}^{\boldsymbol{RT0}}) \tag{2.61}$$

Burada; $\boldsymbol{P}_{\boldsymbol{TOOL}_{\boldsymbol{u}}}^{\boldsymbol{RT0}}$, $\boldsymbol{P}_{\boldsymbol{TOOL}_{\boldsymbol{l}}}^{\boldsymbol{RT0}}$ tarama yüzeyinin en üst ve en alt noktalarının sensör ideal tarama mesafesine bağımlı olarak, tarama yüzeyinin dışına doğru ötelenmiş pozisyon vektörlerinin döner tabla koordinat sistemindeki ifadeleridir. $\boldsymbol{P}_{\boldsymbol{TCP}_{\boldsymbol{u}}}^{\boldsymbol{B}}$ ve $\boldsymbol{P}_{\boldsymbol{TCP}_{\boldsymbol{l}}}^{\boldsymbol{B}}$ ise bu noktaların robot taban koordinat eksenindeki karşılıklarıdır. $\boldsymbol{R}_{\boldsymbol{TCP}}^{\boldsymbol{B}}$, endüstriyel robotun bağlantı noktasının tarama işlemini gerçekleştirmek için sahip olması gereken oryantasyon matrisine karşılık gelmektedir.

Ortaya çıkarılan nihai tarama noktaları ve oryantasyon matrisi, endüstriyel robot

tarafından taşınmakta olan lazer profil sensörünün belirlenen tarama yüzeyi için istenen pozisyonlar arasında, gerekli olan oryantasyonda hareket ettirilmesini sağlayacaktır.

2.3.2. Lazer Tarama İşleminin Gerçekleştirilmesi

Robot hareket planının oluşturulması sırasında elde edilmiş olan tarama işlemi için lazer sensör oryantasyonu veren oryantasyon matrisi ($\boldsymbol{R_{TCP}^{B}}$) ve tarama işlemine ait belirlenen en üst $\boldsymbol{P_{TCP_u}^{B}}$ ve en alt $\boldsymbol{P_{TCP_l}^{B}}$ noktaların endüstriyel robota sağlanması ile, lazer sensörü robot kolu bulunduğu pozisyondan, tarama işleminin başlangıç pozisyonuna ilerlemeye başlayacaktır ($\boldsymbol{P_{TCP_u}^{B}}$ veya $\boldsymbol{P_{TCP_l}^{B}}$). Başlangıç pozisyona gelmiş ve oryantasyonu ($\boldsymbol{R_{TCP}^{B}}$) oturmuş olan endüstriyel robot kolu, sahip olduğu oryantasyonu koruyarak, bitiş pozisyonuna lineer hareket yapacak şekilde ilerlemeye ve ilerleyişi sırasında lazer profil sensörüne ölçüm alma talebi olduğunu belirten elektriksel sinyali göndermeye başlayacaktır. Lazer profil sensör tarafından alınan ölçümler, geliştirilen yazılımın çalışmakta olduğu bilgisayara, endüstriyel robotun ilgili zaman anındaki pozisyonu ve oryantasyonu ile birlikte gönderilip kaydedilmektedir. Tarama işleminin sonunda, elde edilmiş olan profil sensör koordinat sisteminde tanımlı noktalar ve ilgili noktaların alındığı andaki robot pozisyonu ve oryantasyonuna ait veri kullanılarak, tarama işlemi ile elde edilmiş olan bütün noktalar endüstriyel robot taban koordinat sisteminde tanımlı noktalara dönüştürülmektedir.

$$P^{B} = T_{TCP}^{B}\, T_{TOOL}^{TCP}\, P^{TOOL} \tag{2.62}$$

Burada; P^{TOOL} lazer profil sensörü tarafından alınan ölçüme, T_{TOOL}^{TCP} lazer profil sensörü ve robot bağlantı noktası arasında dönüşüm matrisine, T_{TCP}^{B} lazer profil sensörü ölçüm verisi ile birlikte gönderilen robot pozisyon oryantasyon verisi kullanılarak elde edilen dönüşüm matrisine karşılık gelmektedir.

Lazer profil sensörü tarafından elde edilen noktaların endüstriyel robot taban koordinat sistemine dönüştürülmesi, bu noktaların sistem üzerinde tanımlı herhangi bir koordinat sistemine dönüştürülmesine izin vermektedir. Programlama sürecinde kullanılan ana koordinat sistemi döner tablanın 0 pozisyonunda tanımlı döner tabla koordinat sistemi olması sebebi ile bu noktaların döner tablanın ana koordinat sistemine dönüştürülmesi gerekmektedir.

$$P_{\theta}^{RT0} = T_{B}^{RT0} P^{B} \tag{2.63}$$

Burada; T_{B}^{RT0} endüstriyel robot taban koordinat sistemi ile döner tabla koordinat sistemi arasındaki dönüşüm matrisine, P_{θ}^{RT0} tarama işlemi ile elde edilmiş olan bir nokta için dönüşüm matrisi uygulanarak elde edilen yeni noktaya karşılık gelmektedir.

Döner tabla koordinat sistemine dönüştürülme işleminde dikkat edilmesi gereken önemli bir nokta, döner tabla hali hazırdaki pozisyonunun (dönüş açısının) 0° olmamasıdır. Tarama işleminin, endüstriyel robot açısından gerçekleştirilebilir hale getirilebilmesi için, taranacak bölgenin endüstriyel robotun erişilebileceği bir alana döndürülmesi gerekmektedir. Bu gereksinimin kaynağı olarak döner tabla tarama işlemi gerçekleştirildiği sırada 0 pozisyonunda bulunmamaktadır. Bu nedenle elde edilen noktaların döner tabla 0 pozisyonu için tanımlı olan koordinat sisteminde tanımlanması gerekmekte iken elde edilen noktalar (P_{θ}^{RT0}) döner tablanın, ana koordinat sistemine göre z ekseni boyunca gerçekleştirilmiş olan ve döner tablanın o anki pozisyonu olan θ birimlik bir dönüşümün etkisi altındadır.

$$R_{z}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.64}$$

Burada; $R_{z}(\theta)$ döner tabla dönüş açısından kaynaklanan, döner tabla 0 pozisyonunda tanımlanmış koordinat sistemine göre farklılığı oluşturan döndürme matrisini temsil etmektedir. Aşağıdaki denklem bu matrisin etkisini göstermektedir.

$$P_{\theta}^{RT0} = R_{z}(\theta) P^{RT0} \tag{2.65}$$

Burada; P^{RT0} , döner tabla 0 pozisyonunda tanımlı olan koordinat sisteminde tanımlı bir noktaya karşılık gelmektedir.

Sistem üzerinde gerçekleştirilecek işlemler için seçilen ana koordinat sisteminin döner tabla 0 pozisyonu için tanımlanmış olan koordinat sistemi olması sebebi ile bütün noktaların bu koordinat sistemine çevrilmesi gerekmektedir. Bu nedenle, Denklem (2.64)'de verilen döndürme matrisinin etkisini yok edilmelidir.

$$R_{z}(\theta)^{-1} P_{\theta}^{RT0} = R_{z}(\theta)^{-1} R_{z}(\theta) P^{RT0} \tag{2.66}$$

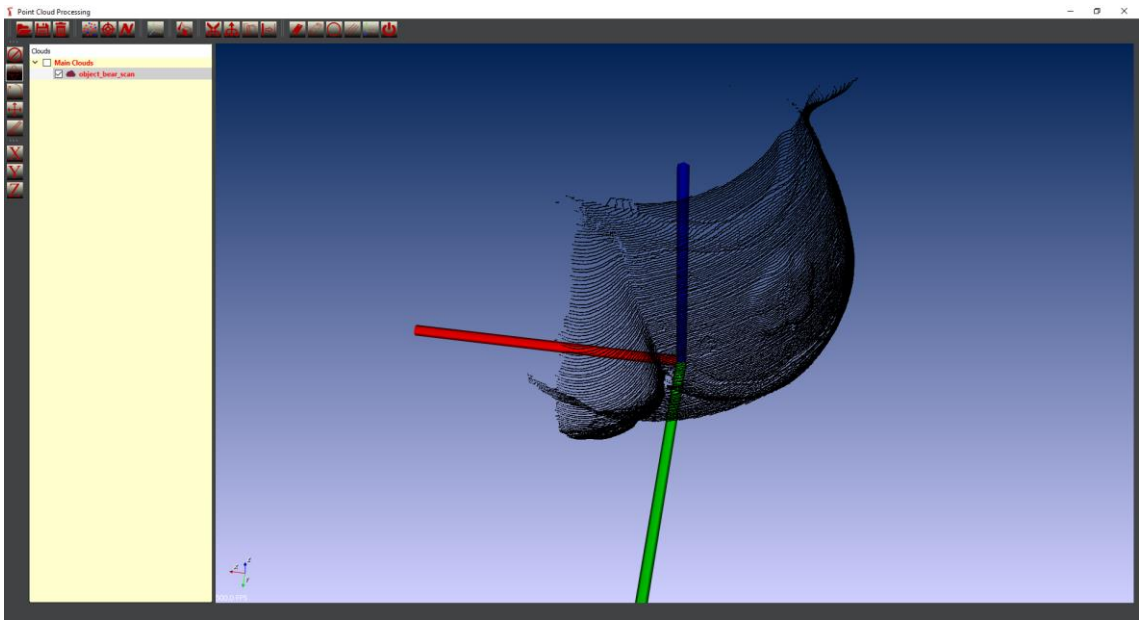
$$R_{z}(\theta)^{-1} = R_{z}(-\theta) \tag{2.67}$$

$$R_z(-\theta)P_\theta^{RT0} = R_z(-\theta)R_z(\theta)P^{RT0} \tag{2.68}$$

$$R_z(-\theta)R_z(\theta) = I \tag{2.69}$$

$$P^{RT0} = R_z(-\theta)P_\theta^{RT0} \tag{2.70}$$

Denklem (2.70)'de verilen dönüşüm, Denklem (2.63)'de elde edilen bütün noktalara uygulandığında, lazer profil sensör tarama işlemi elde edilmiş nokta bulutunun, döner tabla koordinat sistemindeki karşılığı elde edilmiş olacaktır. Görsel 2.19, gerçekleştirilmiş olan tarama işlemi ve elde edilen nokta bulutunun döner tabla koordinat sistemine dönüştürülmesi ile elde edilen yeni nokta bulutunu sergilemektedir. Görselde gösterilen koordinat sistemi döner tablanın 0 pozisyonu için tanımlı olan koordinat sistemine karşılık gelmektedir.



Görsel 2.19. *Nokta Bulutu ve Döner Tabla Koordinat Sistemi*

2.3.3. Nokta Bulutunun Kamera Görüntüsü ile Renklendirilmesi

Elde edilen nokta bulutu (Görsel 2.19), geliştirilen sistem için programlama sürecinde kullanılacak nokta bulutunu temsil etmektedir. Bu nokta bulutu ve operatör kullanıcı arayüzü etkileşimi ile operatörün gerçekleştirilmek istediği işlem için, operatör robot hareket planı oluşturulacaktır. Fakat verilen örnek görselde de görüldüğü üzere, bu

nokta bulutunun temsil ettiği nesneye ait detaylar yeterince belirgin değildir. Bunun temel nedeni nokta bulutunun tek renkte olmasıdır. Bu durum kullanıcı etkileşimini ve dolayısıyla programlama sürecini güçleştirecektir. Bu nedenle söz konusu nokta bulutunun işlem görecek nesnenin sahip olduğu renk bilgisi ile zenginleştirilmesi pozitif yönlü bir kullanıcı etkileşimi sağlayacaktır.

Günümüzde kullanılan lazer profil sensörleri ile gerçekleştirilen ölçümlerin renk bilgisi taşımaması, elde edilen nokta bulutunun renklendirilmesi sürecine farklı bir problem boyutu kazandırmaktadır. Endüstriyel Robot hücresinin bir kameraya sahip olması ve ilk kullanıcı etkileşiminin (ilgilenilen bölgenin belirlenmesi) bu kamera üzerinden alınan görüntüler yardımı ile gerçekleştiriliyor olması elde edilen nokta bulutunun, ilgilenilen bölgenin belirlenmesi sırasında elde edilmiş olan kamera görüntüsü kullanılarak renklendirilmesine olanak sağlamaktadır.

Nokta bulutunun renklendirilmesi süreci temel olarak, elde edilen nokta bulutunun, elde edilen kamera görüntüsüne ait kamera düzlemi koordinat sistemi üzerine geri yansıtılarak, her bir noktanın görüntü üzerindeki piksel karşılığının bulunması ve bu pikselin renk değerlerinin ilgili noktanın renk değerleri olarak atanmasıdır. Bu süreç iki basamaklı olarak gerçekleştirilmektedir.

İlk olarak, sahip olunan nokta bulutunun döner tabla koordinat sisteminde temsil ediliyor olması, üç boyutlu noktaların kamera düzlemine geri yansıtılmasını olanaksız kılmaktadır. Bu nedenle, nokta bulutunun döner tabla koordinat sistemi ve kamera kalibrasyon nesnesi koordinat sistemi arasındaki ilişki (T_{CALIB}^{RT0}) kullanılarak dönüştürülmesi gerekmektedir. Ayrıca, ilgili bölgenin kamera görüntüsü üzerinden belirlenebilmesi için gerçekleştirilen, ilgili bölgenin kamera görüş alanına sokulabilmesi için uygulanan döner tabla döndürme işleminin de yine aynı dönüşüm işlemine dahil edilmesi gerekmektedir.

$$P^{CALIB} = T_z(\theta) T_{RT0}^{CALIB} P^{RT0} \tag{2.71}$$

Burada; P^{RT0} döner tabla koordinat sisteminde tanımlı bir noktaya, T_{RT0}^{CALIB} döner tabla koordinat sistemi ve kamera kalibrasyon nesnesi tarafından tanımlanmış olan koordinat sistemi arasındaki dönüşüm matrisine, $T_z(\theta)$ ilgilenilen bölgenin kamera görüş alanına sokulması sırasında kullanıcı tarafından gerçekleştirilen döndürme miktarı (θ) için uygulanması gereken z ekseni boyunca döndürme matrisine, P^{CALIB} ise döner table koordinat sisteminde tanımlı olan noktanın, kamera kalibrasyon nesnesi koordinat

sistemindeki ifadesine karşılık gelmektedir.

Döner tabla koordinat sisteminde tanımlı olan nokta bulutunun kalibrasyon nesnesi koordinat sistemindeki karşılığının bulunması ile elde edilen yeni nokta bulutu üzerindeki her bir noktanın, kamera kalibrasyon nesnesi ve kamera görüntü düzlemi arasındaki ilişkiyi veren ve kamera kalibrasyon sürecinde elde edilmiş olan dönüşüm matrisi (**M**) kullanılarak, kamera görüntü düzlemindeki karşılığı bulunabilir.

$$p = \mathbf{M}P^{CALIB} \tag{2.72}$$

Burada; p kamera görüntü düzleminde tanımlı bir homojen noktaya karşılık gelmektedir.

Bütün noktalar için elde edilmiş olan her bir piksel değerinin, ilgili görüntü üzerinde bir renk karşılığı bulunmaktadır. Eğer bu renk bilgisi, ilgili piksele karşılık gelen nokta buluna ait olan noktanın renk bilgisine atanır ise, nokta bulutundaki noktaların renklendirilmesi gerçekleştirilmiş olacaktır.

$$P_{RGB}^{CALIB}{}_{i} = RGB(p_i) \tag{2.73}$$

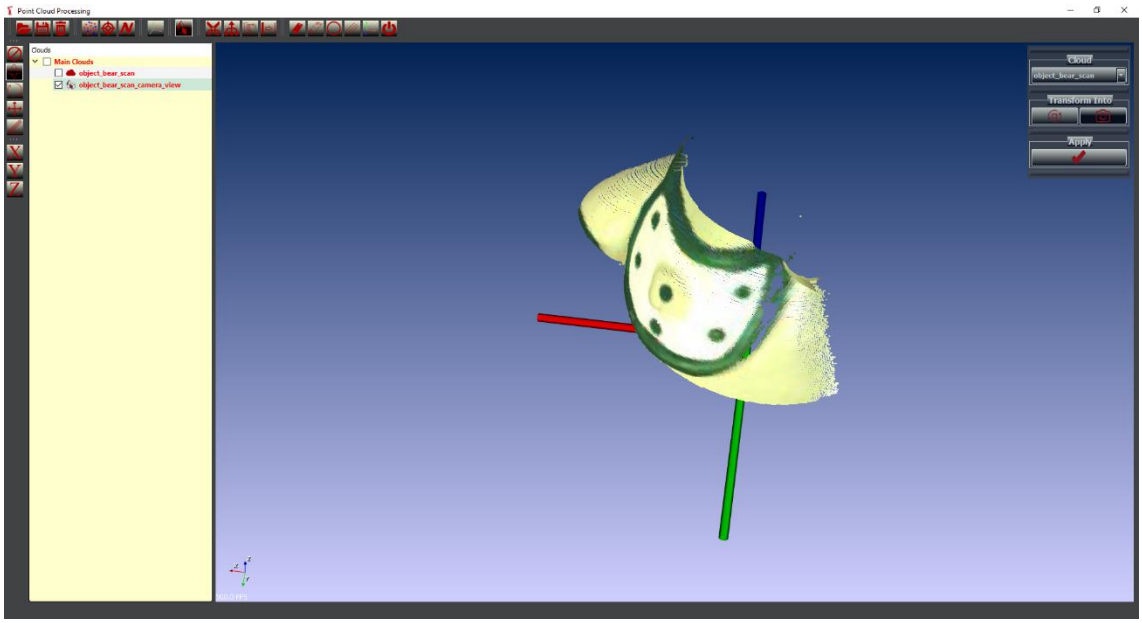
Burada; p_i nokta bulutu üzerindeki i'inci noktanın, kamera görüntü düzlemindeki koordinatlarına, $P_{RGB}^{CALIB}{}_{i}$, nokta bulutu üzerindeki i'inci noktanın renk bilgisini taşıyan alanlarına, $RGB(\dots)$ ise görüntü üzerindeki bir piksel koordinatının renk bilgisine erişilmesini sağlayan bir metoda karşılık gelmektedir.

Denklem (2.71) ve (2.73) arasında tarif edilen sürecin gerçekleştirilmesi ile elde edilen renklendirilmiş nokta bulutu örneği Görsel 2.20'de sergilenmektedir.

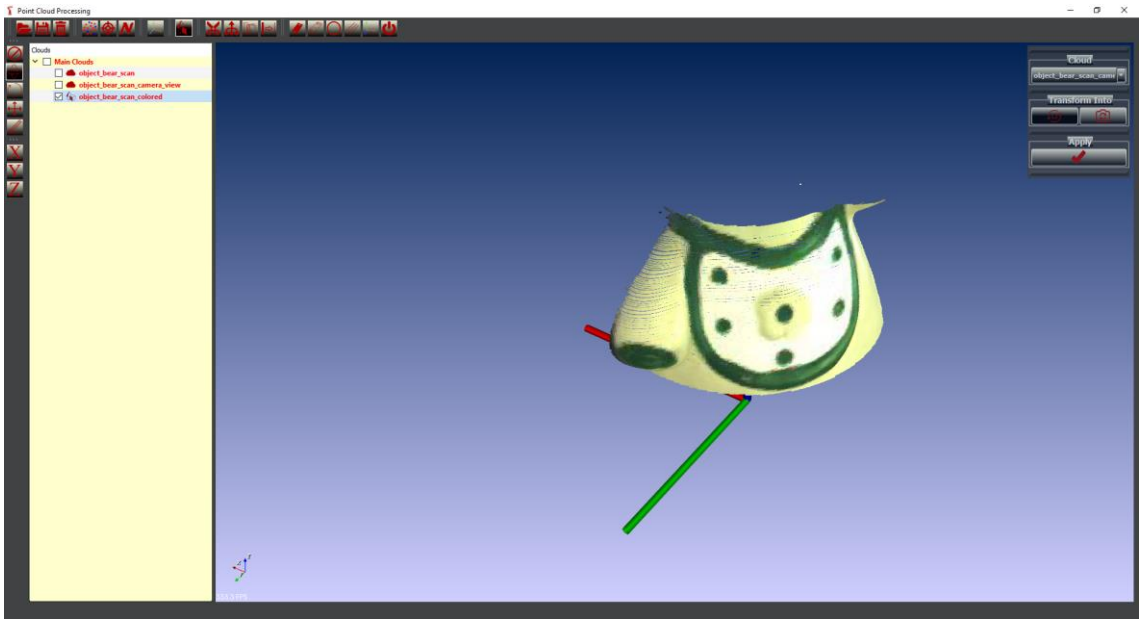
Renklendirilmiş olan nokta bulutunun kamera kalibrasyon nesnesi koordinat sisteminde tanımlı olması, bu nokta bulutunun programlama sürecinde kullanılmasına izin vermemektedir. Bu nedenle, renklendirilmiş bu nokta bulutunun yeniden döner tabla taban koordinat sistemine dönüştürülmesi gerekmektedir. Benzer şekilde kullanıcı tarafından ilgilenilen bölgenin kamera görüş alanına sokulması sürecinde uygulanan döndürme işleminin etkisi de göz önünde bulundurulmalıdır. Denklem *(2.74)* ün renklendirilmiş olan bütün noktalara uygulanması ile kamera kamera kalibrasyon nesnesi koordinat sisteminde tanımlı olan noktaların döner tabla koordinat sistemine dönüştürülmektedir. Görsel 2.21, elde edilen renklendirilmiş nokta bulutunun döner tabla koordinat sistemindeki karşılığını göstermektedir.

$$P_{XYZRGB}^{RT0} = T_z(-\theta)T_{CALIB}^{RT0}P_{XYZRGB}^{CALIB} \quad (2.74)$$

Burada; P_{XYZRGB}^{CALIB} kamera kalibrasyon nesnesine ait koordinat sisteminde tanımlı renklendirilmiş noktaya, T_{CALIB}^{RT0} döner tabla koordinat sistemi ile kamera kalibrasyon nesnesi koordinat sistemi arasındaki dönüşüme, $T_z(-\theta)$ operatör tarafından döner tabla ile gerçekleştirilen döndürme işleminin etkisine yok eden dönüşüm matrisine, P_{XYZRGB}^{RT0} ise elde edilmek istenen döner tabla koordinat sisteminde tanımlı renklendirilmiş noktalara karşılık gelmektedir.



Görsel 2.20. *Renklendirilmiş Nokta Bulutu ve Kalibrasyon Nesnesi Koordinat Sitemi*



Görsel 2.21. *Renklendirilmiş Nokta Bulutu ve Döner Tabla Koordinat Sitemi*

2.4. Robotik İşlem için Gerekli Yolun Belirlenmesi ve Hareket Planlaması

Endüstriyel robotik işlem uygulanacak olan nesneye ait ilgilenilen bölgenin üç boyutlu modelinin lazer profil sensor taraması ile edilmesinin ardından elde edilen model kullanıcı arayüzü üzerinde görüntülenmektedir (Bkz. Görsel 2.21). Görüntülenen model kullanılarak ortaya çıkarılacak olan endüstriyel robot hareket planı operatörün kullanıcı arayüzü ile etkileşimi ile gerçekleştirilmektedir.

Nokta bulutu yardımı ile gerçekleştirilecek olan programlama sürecinde, öncelikli olarak gerçekleştirilecek işleme ait başlangıç ve bitiş noktasının operatör tarafından belirlenmesi gerekmektedir. Belirlenen başlangıç ve bitiş noktasına bağımlı nokta bulutu üzerinde gerçekleştirilecek olan kırpma işlemi ile, elde edilmiş nokta bulutu üzerinde ilgilenilen bölgede detaylı analiz yapılabilir hale gelinmektedir.

Kırpılmış nokta bulutu üzerinde bulunan potansiyel kenar noktaları sistem tarafından tespit edilip, kullanıcı arayüzü üzerinde operatöre gösterilir. Bu kenar noktalarının yoğunluğunun fazla olması durumunda operatör talebi ile yoğunluk azaltma işlemi uygulanır. Yoğunluğu azaltılan potansiyel kenar noktaları, robotun izleyeceği yolu belirleyen hareket planının oluşturulması için temel oluşturacaktır.

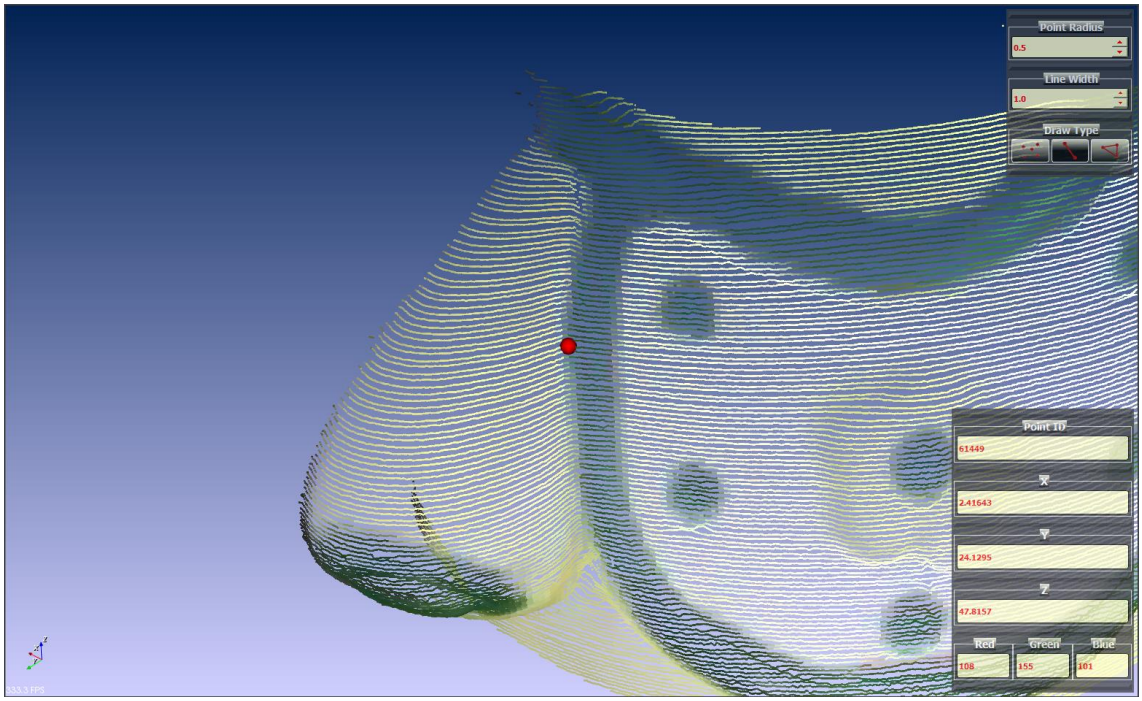
Elde edilen kenar noktaları robotun izleyeceği yolu ortaya çıkartmaktadır. Fakat bu noktalar için robotun hangi oryantasyon ile hareket edeceğinin belirlenmesi

gerekmektedir. Robot oryantasyonunun hesaplanması, izlenecek robot yolunu temsil eden noktalara ait normal vektörlerin belirlenmesi ile mümkün kılınmaktadır.

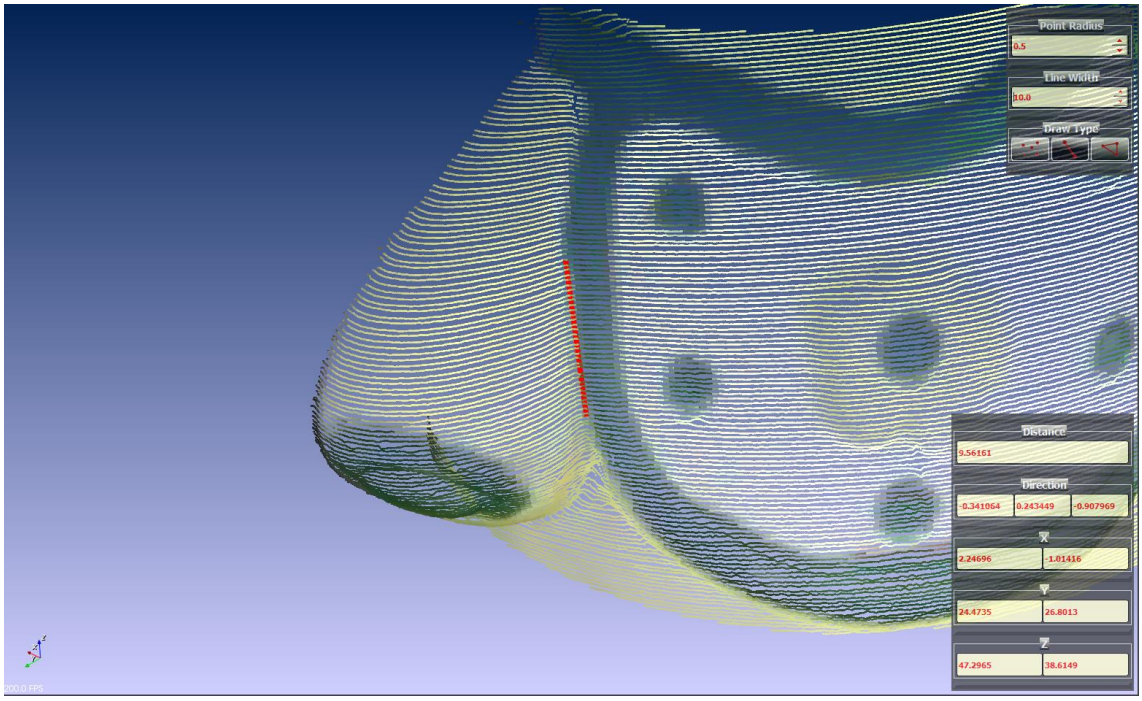
Robotun izleyeceği yol ve ilgili oryantasyonun hesaplanmasının ardından elde edilen pozisyon ve oryantasyon bilgileri robot kontrolcüsüne gönderilerek robotun hareket etmesi ve işlem görecek uç aparatının belirlenen yol ve oryantasyon değerlerinde hareket ettirilmesi sağlanmaktadır.

2.4.1. Başlangıç ve Bitiş Noktalarının Belirlenmesi

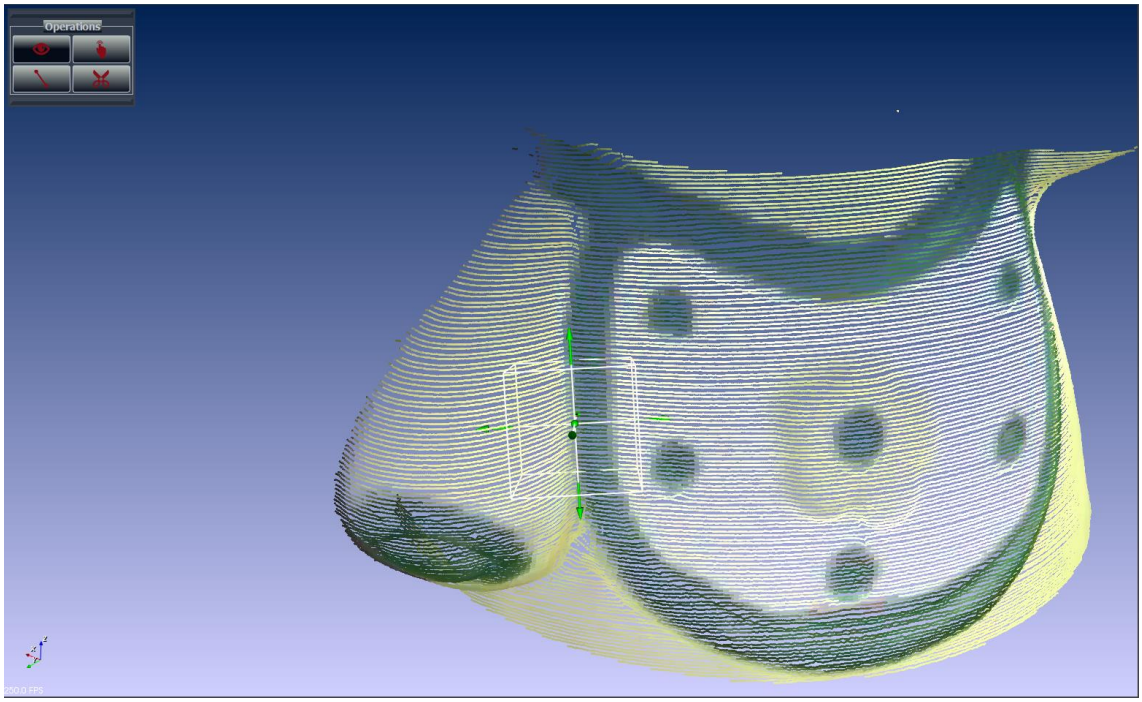
Gerçekleştirilecek olan endüstriyel işlem için robotun izleyeceği yola ait başlangıç ve bitiş noktalarının, lazer tarama ile elde edilmiş olan nokta bulutu kullanılarak, operatör tarafından belirlenmesi gerekmektedir. Belirlenen bu noktalar endüstriyel robotun ucunda bağlı olan işlem ucunun ilk temas edeceği ve son temas edeceği noktaya karşılık gelmektedir. Bu iki noktanın arasında kalan noktalar ise izlenecek yolu belirleyecektir. Ayrıca belirlenmiş olan bu noktalar, nokta bulutu üzerinde kapalı dikdörtgen prizması oluşturmaya olanak sağlamaktadır. Bu prizma aynı zamanda operatör kullanıcı arayüzü etkileşimi ile büyültülüp küçültülebilmekte ve/veya oryantasyonu değiştirilebilmektedir. Oluşturulan bu prizmanın içerisinde kalan noktalar ile ilerideki adımlarda nokta bulutuna ait geometrik özellikler tespit edilecektir. Bu sayede nokta bulutu üzerinde gerçekleştirilecek yüksek hesaplama yükü oluşturan işlemler için harcanan süre azaltılmış olacaktır (Bkz. Görsel 2.22-2.25).



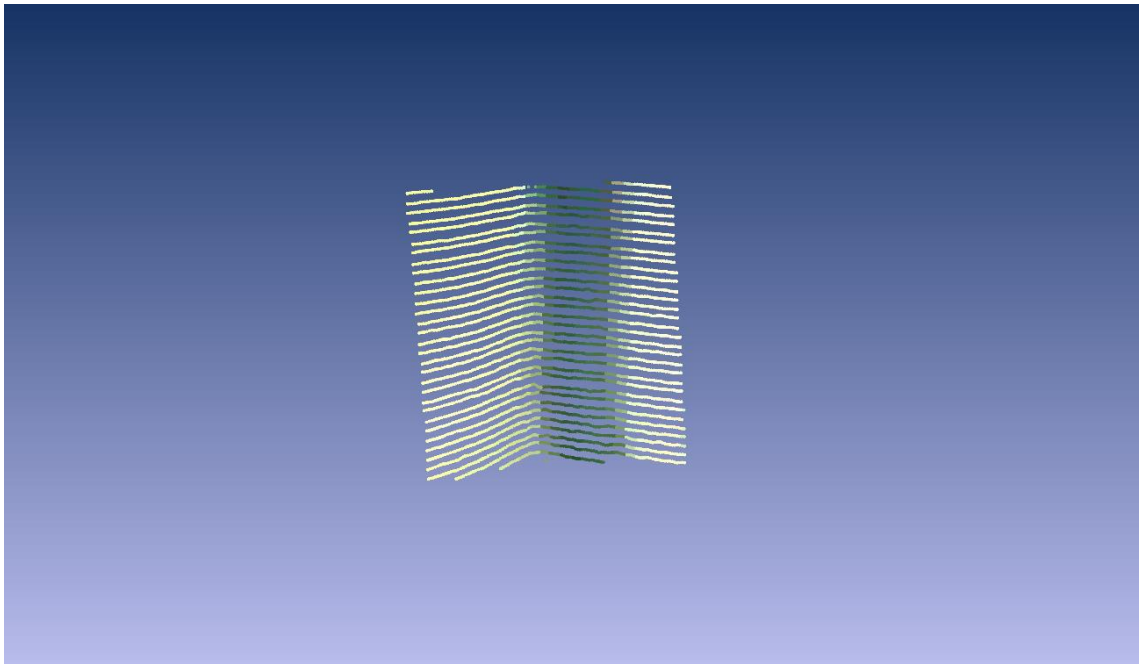
Görsel 2.22. *İlk İşlem Noktası*



Görsel 2.23. *İlk İşlem Noktası ve Son İşlem Noktasını Birleştiren Çizgi*



Görsel 2.24. *Kırpma Prizması*



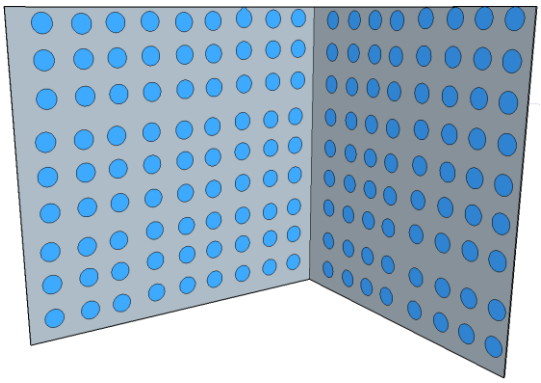
Görsel 2.25. *Belirlenen İlk ve Son İşlem Noktasına Göre Kırpılmış Nokta Bulutu*

2.4.2. Kenar Noktalarının Bulunması ve Seyreltilmesi

Operatör tarafından belirlenen ilk ve son noktaya dayalı olarak ortaya çıkartılan

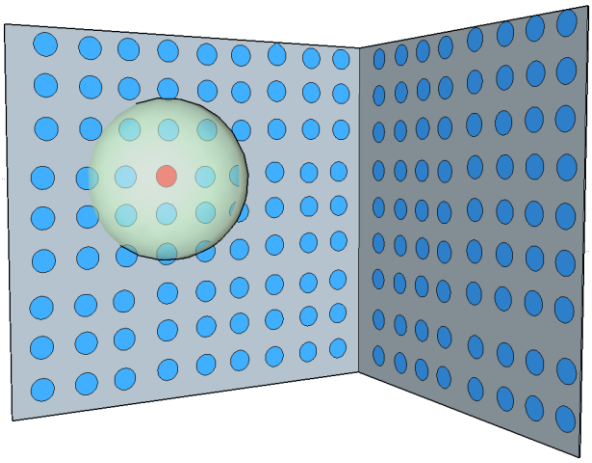
kırma prizması ve bu prizma kullanılarak kırpılan nokta bulutu üzerinde kenar nokta olma potansiyeli taşıyan noktaların tespit edilip operatöre görüntülenmektedir. Operatörün talep etmesi durumunda nokta bulutu üzerinde kenar potansiyeli taşıyan noktalar için seyreltilme işlemi yapılmaktadır. Bu bölümde gerçekleştirilen işlem, kullanıcı tarafından belirlenen ilk ve son noktanın yeterli hassasiyette seçilememesi durumunda ortaya çıkarılacak hareket planı hatalarının giderilmesini sağlamaktadır. Elde edilmiş olan nokta bulutunun geometrik özelliklerinden faydalanılarak hesaplanan potansiyel kenar noktaları, robot hareket yolunun hassas bir şekilde elde edilmesini sağlamakla beraber, kullanıcıya olan bağımlılığı ortadan kaldırmaktadır.

Elde edilen nokta bulutu üzerinde konumlanmış olan bir iç kenar bölgesinin tespit edilebilmesi için nokta bulutu üzerinde kenar noktası olma potansiyeli taşıyan noktaların ayırt edilebilmesi gerekmektedir. Şekil 2.16'de nokta bulutu üzerinde kırpma yapılmış ve iç kenar barındıran temsili bir nokta bulutu (mavi noktalar) gösterilmektedir.

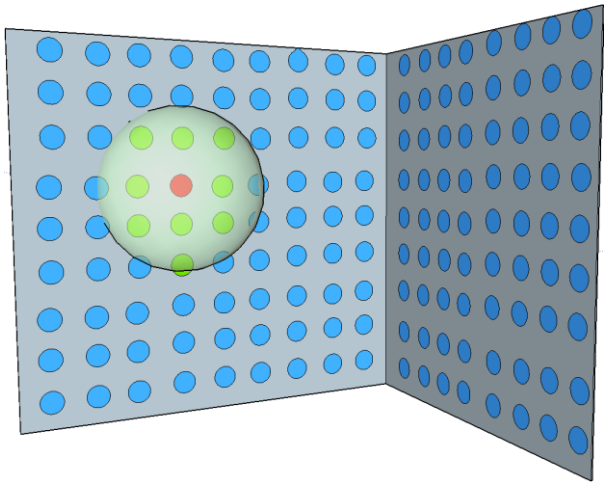


Şekil 2.16. *İç Kenar Barındıran Temsili Nokta Bulutu*

Herhangi bir noktanın kenar bölge civarında olup olmadığının tespit edilmesi ilgili noktanın belli bir yarıçapla tanımlı komşuluklarında bulunan noktaların analiz edilmesi ile belirlenebilir. Şekil 2.17'de gösterilen kırmızı renkli noktanın kenar bölge olup olmadığının belirlenmesi gerektiği varsayılır ise; bu noktayı merkez kabul eden belli yarıçaplı bir küre kullanılarak komşuluk noktalar bulunabilir. Kürenin içerisinde kalan noktalar komşulukları temsil etmektedir. Bu noktalar Şekil 2.18'de yeşil renk ile gösterilmektedir.

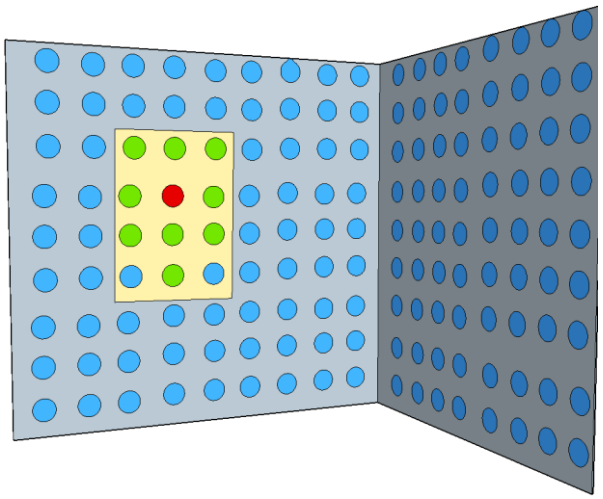


Şekil 2.17. *Analiz Edilen Nokta ve Bu Noktaya ait Arama Bölgesini Gösteren Küre*

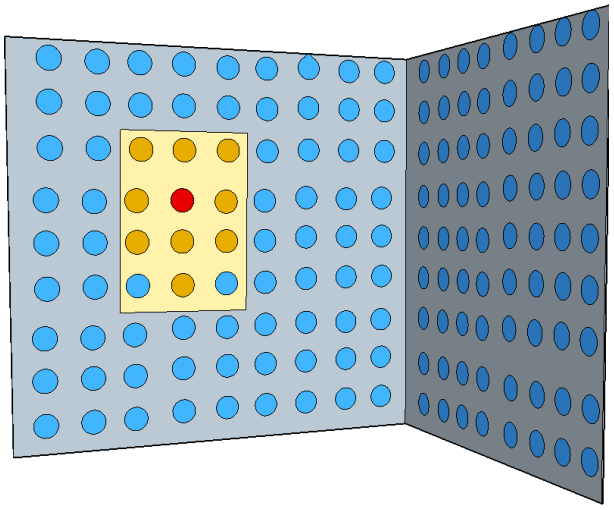


Şekil 2.18. *Tespit Edilen Komşuluk Noktaları*

Tespit edilen komşuluk noktaları (yeşil noktalar) üzerine bir düzlem uydurulabilir ve düşük bir hata payı ile bu düzlem üzerinde kalan noktalar belirlenebilir. Şekil 2.19'da komşuluk noktaları için uydurulan düzlem (sarı) gösterilmekte ve Şekil 2.20'de bu düzlem üzerinde olan noktalar (turuncu) gösterilmektedir. Burada ortaya çıkan ilk durum eğer nokta kenar bölgeye yakın bir nokta değil ise, komşuluk noktaları için uydurulan düzlemi oluşturan nokta kümesi komşuluk noktalarının tamamını ya da büyük çoğunluğunu içerecektir.

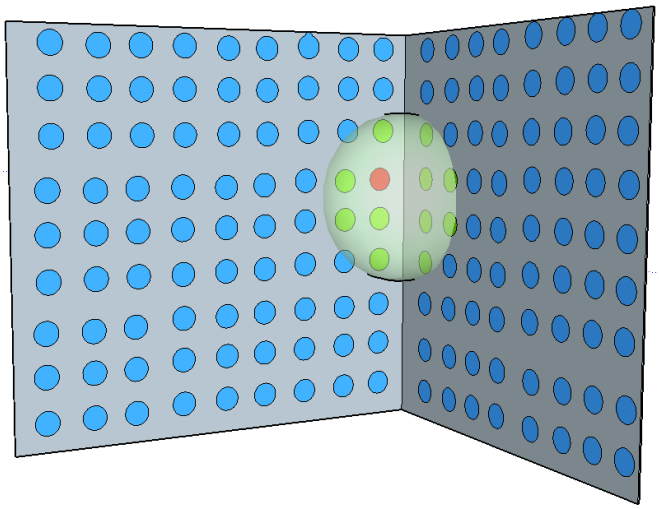


Şekil 2.19. *Komşuluk Noktalar için Uydurulan Düzlem*

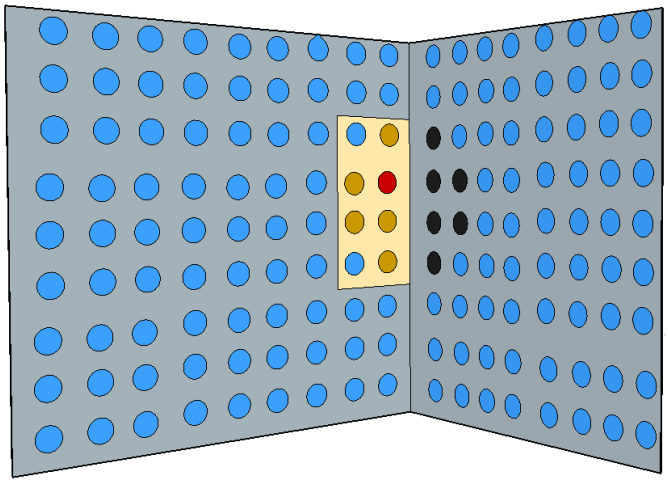


Şekil 2.20. *Uydurulan Düzlem ve Uygun Olan Noktalar*

Kenar bölgede bulunan bir nokta için komşuluk arama ve düzlem uydurma işlemleri uygulandığında elde edilecek durum bir önceki duruma göre farklılık gösterecektir. Şekil 2.21'de kenar bölgede bulunan bir nokta (kırmızı) için komşuluk küresi oturtulmuş ve komşuluk kümesini oluşturan noktalar (yeşil) belirlenmiştir. Şekil 2.22'de ise komşuluk noktaları kullanılarak uydurulan düzlem (sarı) ve bu düzlemi oluşturan noktalar (turuncu) ile komşuluk kümesinde var olmasına karşılık düzlemi oluşturan kümeye dahil edilmeyen noktalar (siyah) gösterilmektedir. Burada ortaya çıkan durum ise eğer nokta kenar bölgeye yakın bir nokta ise, komşuluk noktaları için uydurulan düzlemi oluşturan nokta kümesi ile komşuluk kümesini oluşturan nokta kümesi arasında önemli bir fark oluşacaktır.



Şekil 2.21. *Tespit Edilen Komşuluk Noktaları*



Şekil 2.22. *Uydurulan Düzlem, Uygun Olan ve Olmayan Noktalar*

Yukarıdaki şekillerde görüldüğü üzere, herhangi bir noktanın komşuluğunda bulunan noktalar için düzlem uydurma işlemi uygulandığında eğer nokta kenar bölgede değil ise komşulukların tamamına yakını bu düzleme uygun olan noktalar olacaktır. Kenar bölgede ise komşuluk noktaların bir bölümü uydurulan düzlemin dışarısında kalacaktır. Ortaya çıkan bu durum nokta bulutu üzerindeki herhangi bir noktanın kenar bölgesine ait olan bir nokta olup olmadığının tespit edilmesine izin vermektedir (Bkz. Algoritma 2.18).

Görsel 2.26'de yukarıda tarif edilen yöntem ile tespit edilen kenar noktaları (kırmızı) sunulmaktadır. Bu noktalar yoğun bir kenar nokta kümesini göstermekte olup

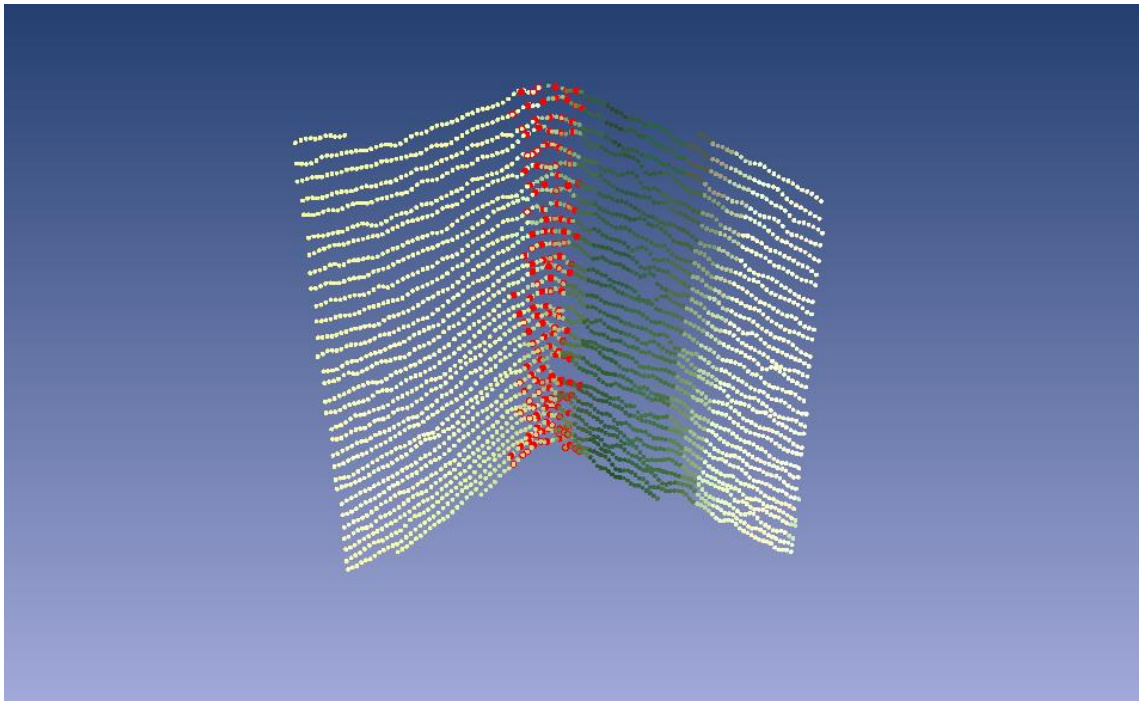
bu noktaların seyreltilmesi ilerleyen süreçte robot hareket planlaması için uydurulacak hareket çizgisinin kararlı olmasını sağlamak için önem taşımaktadır (Bkz. Algoritma 2.19). Görsel 2.27 ve Görsel 2.28 seyreltilmiş kenar noktaları ve bu kenar noktaları için uydurulmuş çizgi nokta kümesini göstermektedir.

Algoritma 2.18: Potansiyel Kenar Noktaların Tespiti

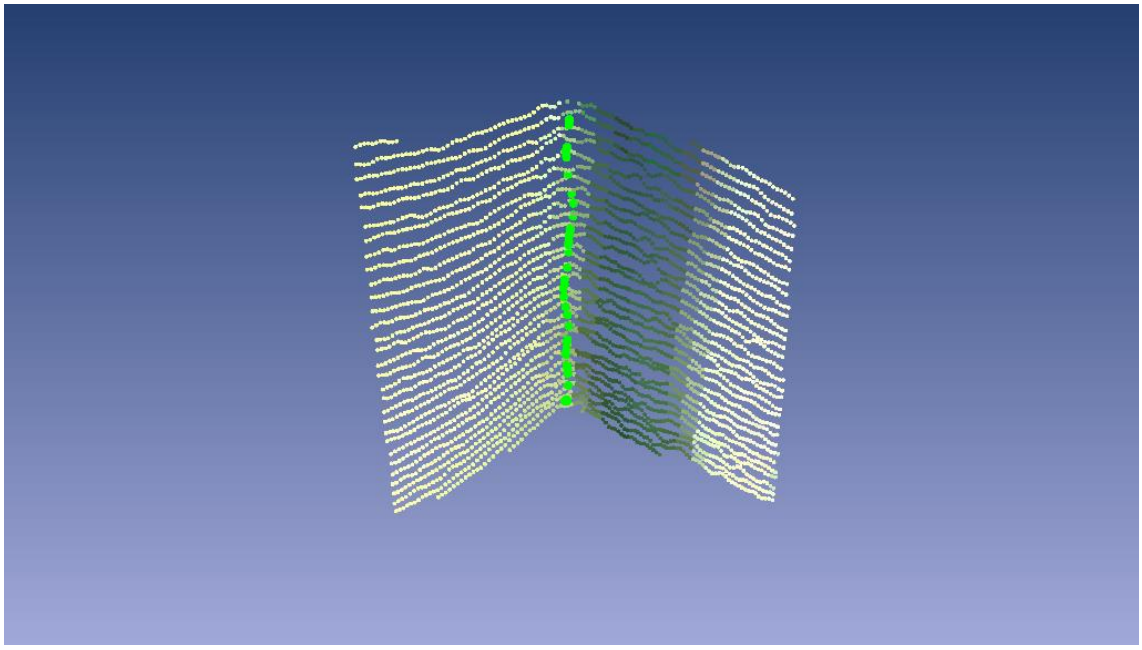
```
1: Fonksiyon PotansiyelKenarNoktalariBul( noktaBulutu,
   izgaraBoyutu, aramaYaricapi, duzlemEsigi, duzlemIcDisNoktaOrani )
2:     filtrelenmisNoktaBulutu ← IzgaraFiltreleme(noktaBulutu,
                                 izgaraBoyutu)
3:     kenarNoktalari ← {}
4:     aramaAgaci ← {filtrelenmisNoktaBulutu}
5:     for aramaNoktasi in filtrelenmisNoktaBulutu do
6:         komsulukNoktaBulutu ← YaricapKomsulukAramasi(
                                 aramaAgaci,
                                 filtrelenmisNoktaBulutu,
                                 aramaNoktasi, aramaYaricapi)
7:         duzlemModeli ← {komsulukNoktaBulutu}
8:         ransac ← {duzlemModeli, duzlemEsigi}
9:         if !ModelHesapla(ransac) then
10:            devam
11:        end if
12:        modeleUygunOlanlar ← ModeleUygunlariAl(ransac)
13:        NModel ← Boyut(modeleUygunOlanlar)
14:        UstSinir ← duzlemIcDisNoktaOrani *
                      Boyut(komsulukNoktaBulutu)
15:        if NModel < UstSinir then
16:            kenarNoktalari ← {kenarNoktalari, aramaNoktasi}
17:        end if
18:    end for
19:    return kenarNoktalari
20: end Fonksiyon
```

Algoritma 2.19: Potansiyel Kenar Noktaların Seyreltilmesi

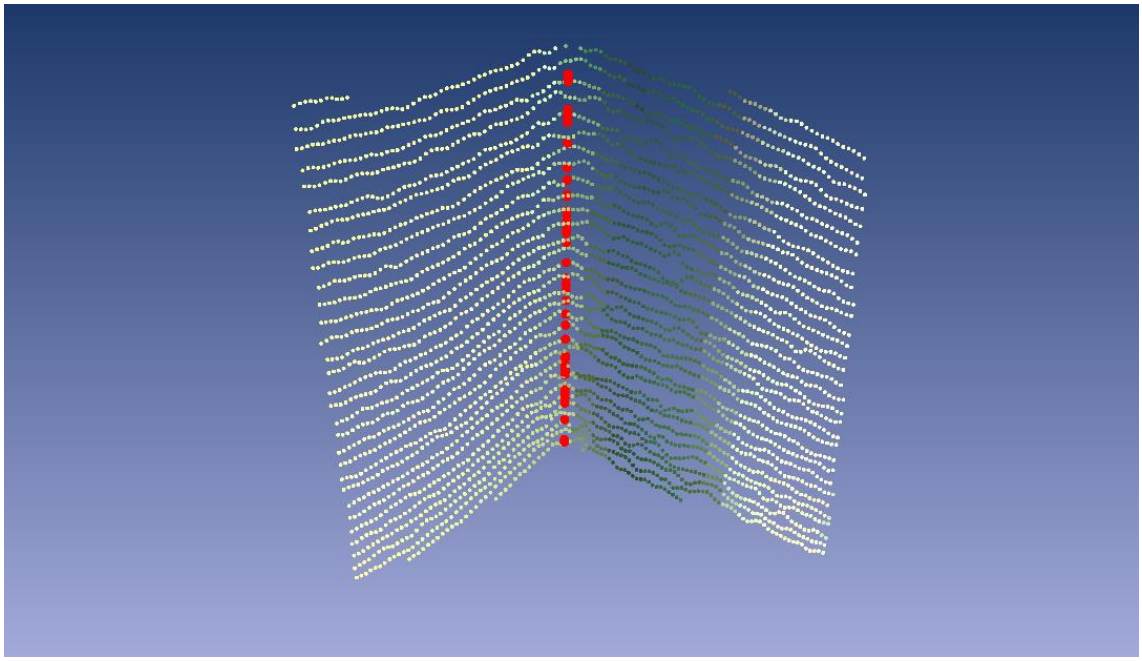
```
1: Fonksiyon PotansiyelKenarNoktalariniSeyrelt( noktaBulutu,
   aramaYaricapi, komsulukSayisiEsigi )
2:     aramaAgaci ← {noktaBulutu}
3:     seyreltilmisNoktaBulutu ← {}
4:     for aramaNoktasi in noktaBulutu do
5:         komsulukNoktaBulutu ← YaricapKomsulukAramasi(
                                 aramaAgaci, noktaBulutu,
                                 aramaNoktasi, aramaYaricapi)
6:         NBoyut ← Boyut(komsulukNoktaBulutu)
7:         if NBoyut > komsulukSayisiEsigi then
8:             merkez ← MerkezBul(komsulukNoktaBulutu)
9:             seyreltilmisNoktaBulutu ← {seyreltilmisNoktaBulutu,
                                         merkez}
10:        end if
11:    end for
12:    return seyreltilmisNoktaBulutu
13: end Fonksiyon
```



Görsel 2.26. *Potansiyel Kenar Noktaları Kümesi (Kırmızı)*



Görsel 2.27. *Potansiyel Kenar Noktaları Kümesinin Seyreltilmesi (Yeşil)*



Görsel 2.28. *Seyreltilmiş Kenar Noktaları İçin Uydurulan Çizgi Nokta Bulutu*

2.4.3. Kenar Noktalarına ait Normal Vektörlerinin Bulunması

Elde edilen kenar noktaları, gerçekleştirilecek olan işlem için oluşturulacak robot hareket planı için izlenecek yolun pozisyonlarının elde edilmesi için yeterli bilgiye sahiptir. Fakat robot kolunun, ortaya çıkacak hareket pozisyonlarına hangi oryantasyonda yanaşacağının da belirlenmesi gerekmektedir. Bu durumda, ortaya çıkan kenar noktalarına ait normal vektörlerin oryantasyon hesaplanmasında kullanılabileceği gözükmektedir. Fakat elde edilen kenar noktaları ve bu kenar noktalarının seyreltilmiş halini içeren nokta bulutları normal vektör hesaplamak için ihtiyaç duyulacak düzlemsel bölgeleri tanımlayamamaktadır. Oysaki ortaya çıkarılması gereken normal vektörler, kenar noktalarının üzerinde bulunduğu gerçek nesneye ait nokta bulutu ile aynı doğrultuda olmalıdır. Bu nedenle, öncelikli olarak, kenar nokta bulutlarının tespit edildiği ana nokta bulutu için normal vektörlerin bulunması gerekmektedir. Kenar noktalarına ait normal vektörlerin ise bulunan normal vektörlere bağımlı olarak türetilmesi, izlenecek olan doğru yoldur.

Ana nokta bulutunun normal vektörlerinin hesaplanması, ilgili noktanın yakınında bulunan komşuluk noktaları kullanılarak gerçekleştirilmektedir. Her bir noktaya ait komşuluk kümesi için bulunan normal vektör, ilgili noktanın normal vektörünü

oluşturmaktadır (Bkz. Algoritma 2.20).

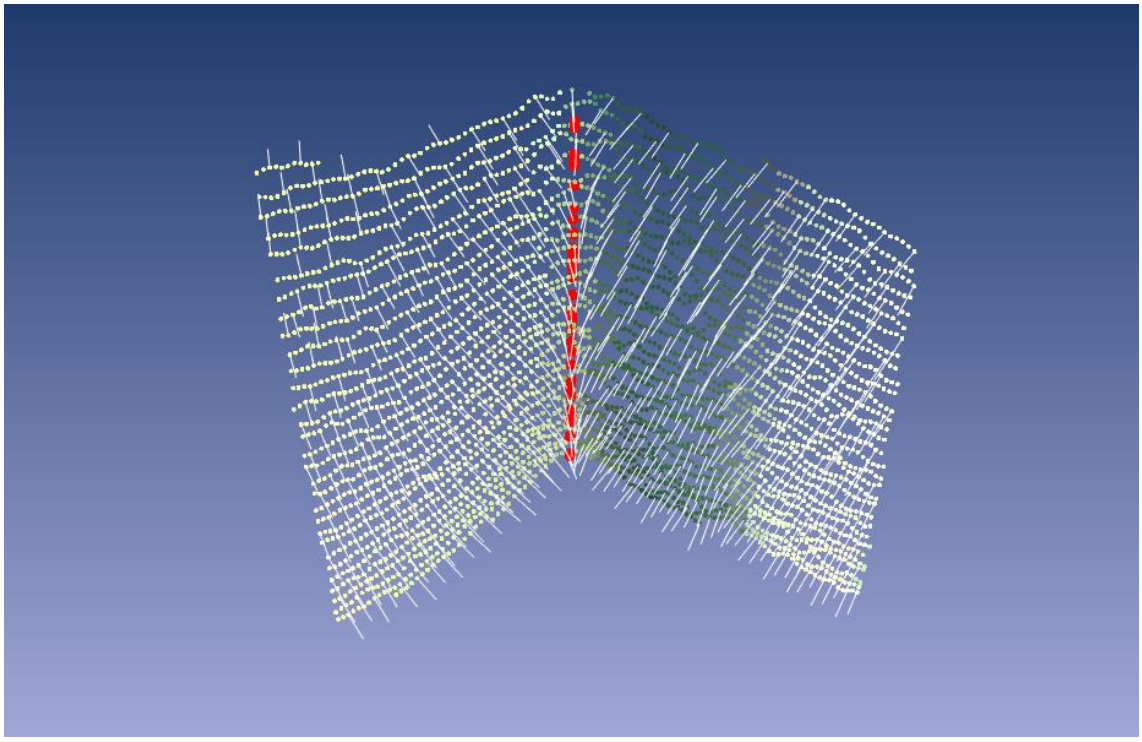
Kenar bölgesinin matematiksel modelini veren uydurulmuş kenar çizgi noktaları için gerekli olan normal vektörler ise bir önceki adımda bulunmuş olan ana nokta bulutunun normal vektörleri kullanılarak elde edilmektedir (Bkz. Algoritma 2.20). Burada çizgi nokta kümesi içerisinde bulunan her bir noktaya, ana nokta bulutu üzerindeki en yakın nokta tespit edilir. Tespit edilen bu noktaya ait nokta normal vektör, ilgili çizgi noktasının normal vektörünü oluşturacaktır (Bkz. Görsel 2.29 ve 2.30). Son olarak elde edilen normal vektörlerin ortalaması alınması ile robot oryantasyon hesabında ihtiyaç duyulan yönelim vektörü elde edilmiş olacaktır (Bkz. Görsel 2.31).

Algoritma 2.20: Nokta Bulutu Normal Vektörlerin Hesaplanması

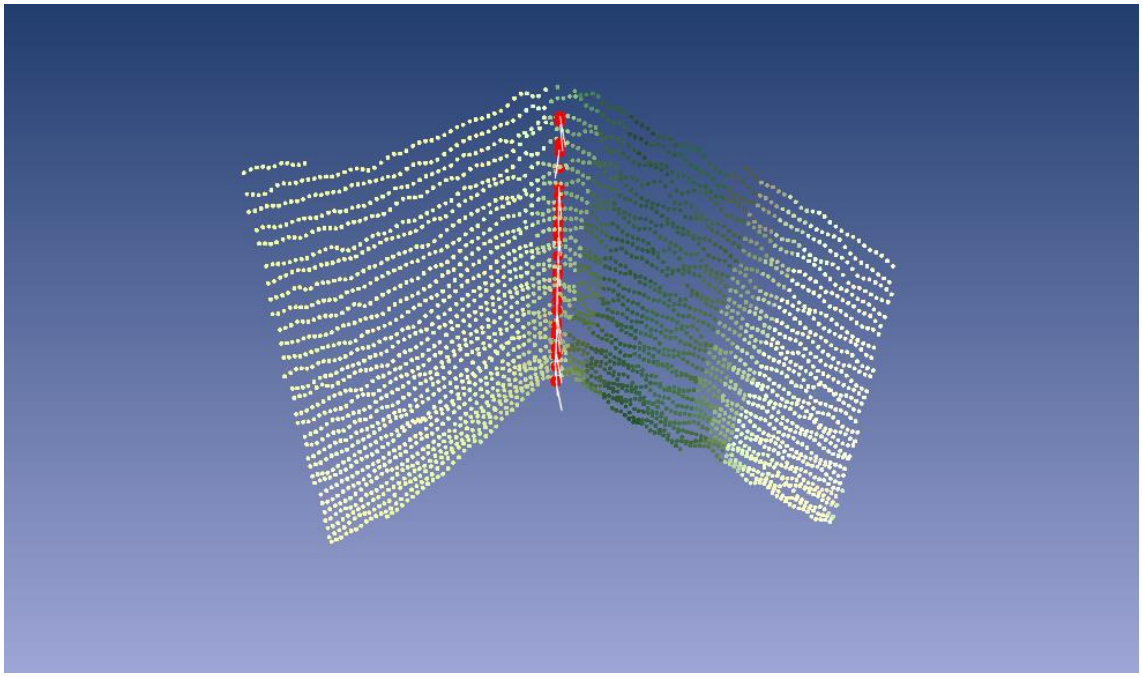
```
1:  Fonksiyon NormalVektorleriBul( noktaBulutu, aramaYaricapi )
2:      aramaAgaci ← {noktaBulutu}
3:      normalVektorler ← {}
4:      for aramaNoktasi in noktaBulutu do
5:          komsuNoktalar ← YaricapKomsulukAramasi(aramaAgaci,
                            noktaBulutu, aramaNoktasi,
                            aramaYaricapi)
6:          normalVektor ← NormalHesapla(komsuNoktalar)
7:          normalVektorler ← {normalVektorler, normalVektor}
8:      end for
9:      return normalVektorler
10: end Fonksiyon
```

Algoritma 2.21: Normal Vektörlerin Ana Nokta Bulutu Üzerinden Tespit Edilmesi

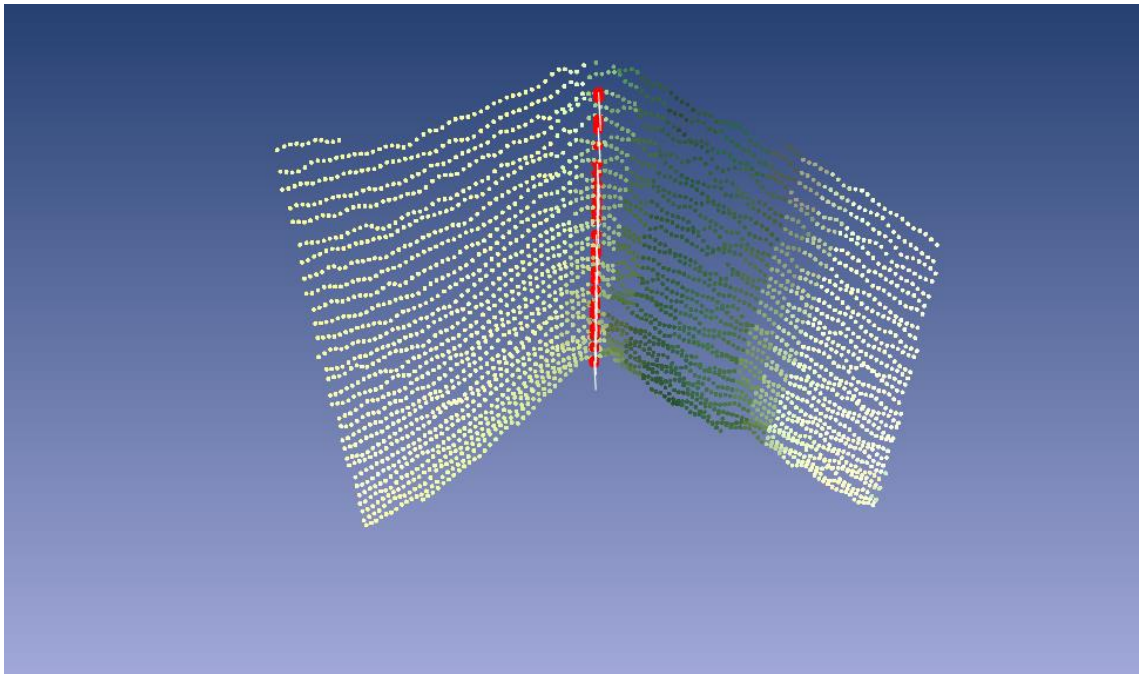
```
1:  Fonksiyon NormalVektorleriAnaNoktaBulutundanAta(
    noktaBulutu, anaNoktaBulutu )
2:      aramaAgaci ← {anaNoktaBulutu}
3:      normalVektorler ← {}
4:      for aramaNoktasi in noktaBulutu do
5:          enYakinNokta ← EnYakinKNoktaAramasi( aramaAgaci,
                           anaNoktaBulutu, aramaNoktasi, 1 )
6:          enYakinNoktaNormali ← NormaliAl(enYakinNokta)
7:          normalVektorler ← {normalVektorler,
                              enYakinNoktaNormalVektoru}
8:      end for
9:      return normalVektorler
10: end Fonksiyon
```



Görsel 2.29. *Cisim Normal Vektörleri*



Görsel 2.30. *Kenar Noktalarına ait Türetilmiş Normal Vektörler*



Görsel 2.31. *Ortalanmış Normal Vektörler*

2.4.4. Robot Hareket Planının Oluşturulması

Robot hareket planının oluşturulması önceki adımlarda elde edilen, robot hareket yolunu tanımlayan doğrusal çizgiye ve bu çizgi için bulunmuş ortalama normal vektöre bağlıdır. Elde edilmiş olan doğrusal çizgiye ait ilk nokta ve son nokta robotun izleyeceği yolun başlangıç (P_f^{RT0}) ve bitiş (P_l^{RT0}) noktasına karşılık gelmektedir. Bu noktaların elde edildiği nokta bulutunun döner tabla sıfır pozisyonuna ait koordinat sisteminde ($RT0$) tanımlı olması sebebi ile bu noktalarda aynı koordinat sisteminde tanımlıdır. Benzer şekilde elde edilen ortama normal vektörde n^{RT0} bu koordinat sisteminde tanımlıdır.

İşlem gerçekleştirilecek nesnenin lazer profil sensör tarama işlemi için döner tabla tarafından döndürülmüş olması nedeni ile döner tabla sıfır pozisyonunda konumlanmamaktadır. Eğer döner tablanın açısı θ ile ifade edilir ise nokta bulutu kullanılarak elde edilen başlangıç, bitiş noktaları ve ortalama normal vektörün robot hareket planlaması için bu açıya bağımlı olarak dönüştürülmesi gerekmektedir. Eğer bu dönüşüm uygulanarak oluşturulan koordinat sistemini RT olarak tanımlanır ise hareket planlaması için kullanılacak noktaların ve normal vektörün bu koordinat sistemine dönüştürülmesi gerekmektedir.

$$P_f^{RT} = T_z(\theta) P_f^{RT0} \tag{2.75}$$

$$P_l^{RT} = T_z(\theta) P_l^{RT0} \tag{2.76}$$

$$n^{RT} = R_z(\theta) n^{RT0} \tag{2.77}$$

Burada; $T_z(\theta)$ döner tabla dönüş açısından dolayı ortaya çıkan z ekseni boyunca döndürme işlemini gerçekleştirmek için kullanılan dönüşüm matrisine, P_f^{RT} ve P_l^{RT} robot ucunun takip edeceği noktaların döner tabla koordinat sistemindeki ifadelerine, $R_z(\theta)$ ise benzer şekilde aynı döndürme işleminden sorumlu döndürme matrisine ve n^{RT} izlenecek doğrusal çizgiye ait normal vektörün, döndürülmüş döner tabla koordinat sistemindeki ifadesine karşılık gelmektedir.

Döner tabla koordinat sistemi üzerinde tanımlı olan normal vektör, gerçekleştirilecek işleme ait işlem ucunun z ekseninin doğrultusuna karşılık gelmektedir. Robot işlem ucunun işlem uygulanacak parçanın üzerine doğru yönelebilmesi için bu vektörün yöneliminin endüstriyel robottan, döner tablaya doğru olması gerekmektedir. Fakat nokta bulutu üzerinden elde edilen bu vektörün yönelimi bu şartı sağlamak zorunda değildir. Bu nedenle bu vektörün gerekli olduğu durumlarda terslenmesi gerekmektedir. Eğer robot hareket yolu üzerindeki herhangi bir noktanın hesaplanmış olan normal vektör doğrultusunda ötelenmesi ile elde edilen yeni nokta endüstriyel robota yaklaşıyor ise normal vektör döner tabladan, endüstriyel robota doğru yönelmiş olduğu ortaya çıkmaktadır. Bu durumda bu vektörün terslenmesi gerekmektedir.

Hareket yolunu tanımlayan doğrusal çizgi nokta bulutu üzerinden alınan nokta hareket başlangıç noktası kabul edilir ise normal vektör doğrultusunda ötelenerek elde edilecek nokta şu şekilde elde edilebilir.

$$P^{RT} = P_f^{RT} + \delta n^{RT} \tag{2.78}$$

Burada; P^{RT} ilk noktanın normal vektör doğrultusunda ötelenmesi ile elde edilen yeni bir noktaya, δ ise öteleme miktarına karşılık gelmektedir.

Endüstriyel robot koordinat sisteminin merkez noktasının döner tabla koordinat sistemindeki ifadesi O_B^{RT} ile gösterilir ise, normal vektörün terslenip terslenmeyeceği belirlenebilir.

$$if\left(\|O_B^{RT} - P^{RT}\| < \left\|O_B^{RT} - P_f^{RT}\right\| \right)\ n^{RT} = -n^{RT} \tag{2.79}$$

Burada; $\|...\|$ vektör büyüklüğüne karşılık gelmektedir.

Robot işlem ucunun oryantasyonunun hesaplanabilmesi için, oryantasyon matrisini tanımlayan üç eksen vektörüne ihtiyaç duyulmaktadır. Bu vektörlerden z sütununa karşılık gelen vektör, hareket nokta bulutunun normal vektörü (ya da terslenmiş hali) ile oluşturulmaktadır. İkinci sütuna karşılık gelen y sütunu ise hareket sırasında izlenecek olan yolun doğrultusu tanımlayan vektör ile oluşturulmaktadır. Bu iki vektörün vektörel çarpımı ile birinci sütuna karşılık gelen x sütunu elde edilmektedir. Elde edilen bu oryantasyon matrisi aynı zamanda işlem ucu için tanımlı olan koordinat sistemi ile döner tabla koordinat sistemi arasında tanımlanan döndürme matrisini ifade etmektedir.

$$n_y^{RT} = \frac{P_f^{RT} - P_l^{RT}}{\left\| P_f^{RT} - P_l^{RT} \right\|} \tag{2.80}$$

$$n_z^{RT} = n^{RT} \tag{2.81}$$

$$n_x^{RT} = n_y^{RT} \times n_z^{RT} \tag{2.82}$$

$$R_{TOOL}^{RT} = \begin{bmatrix} n_x^{RT} & n_y^{RT} & n_z^{RT} \end{bmatrix} \tag{2.83}$$

Burada, n_x^{RT}, n_y^{RT}, n_z^{RT} sırası ile robot işlem ucunun oryantasyonunun döner tabla koordinat sistemindeki karşılığını oluşturan taban eksen vektörlerine ve R_{TOOL}^{RT} ise bu oryantasyonu tanımlayan döndürme matrisine karşılık gelmektedir.

Elde edilen bu döndürme matrisi kullanılarak robot hareket planı için ihtiyaç duyulan başlangıç ve bitiş noktalarının her ikisi içinde dönüşüm matrisleri oluşturulabilir.

$$T_{TOOL_f}^{RT} = \begin{bmatrix} R_{TOOL}^{RT} & P_f^{RT} \\ 0 & 1 \end{bmatrix} \tag{2.84}$$

$$T_{TOOL_l}^{RT} = \begin{bmatrix} R_{TOOL}^{RT} & P_l^{RT} \\ 0 & 1 \end{bmatrix} \tag{2.85}$$

Döner tabla ile endüstriyel robot taban koordinat sistemi arasındaki ilişki (T_B^{RT}) ve işlem ucu ile robot bağlantı ucu arasındaki ilişki (T_{TOOL}^{TCP}) bilinmektedir. Başlangıç ve bitiş noktası üzerinde konumlanacak olan işlem ucunun oryantasyonu ve pozisyonu ile oluşturulmuş olan dönüşüm matrisleri ($T_{TOOL_f}^{RT}$, $T_{TOOL_l}^{RT}$) ise yukarıda gösterildiği gibi hesaplanmaktadır. Bu durumda robot hareketini tanımlayacak olan robot taban ekseni bağlantı ucu arasındaki dönüşüm matrisi her iki nokta için şu şekilde elde edilebilir.

$$T^{B}_{TCP_f} = T^{B}_{RT} T^{RT}_{TOOL_f} T^{TOOL}_{TCP} \tag{2.86}$$

$$T^{B}_{TCP_l} = T^{B}_{RT} T^{RT}_{TOOL_l} T^{TOOL}_{TCP} \tag{2.87}$$

Burada; $T^{B}_{TCP_f}$, $T^{B}_{TCP_l}$ robot işlem ucunun gideceği ilk ve son noktalar için robot işlem ucu bağlantı nokrası ve robot taban koordinat sistemi arasındaki dönüşüm matrislerini ifade etmektedir. Bu matrisler ilgili oldukları noktaya ait oryantasyon matrisini ($R^{B}_{TCP_f}$, $R^{B}_{TCP_l}$) ve pozisyon vektörünü ($P^{B}_{TCP_f}$, $P^{B}_{TCP_l}$) barındırmaktadır.

$$T^{B}_{TCP_f} = \begin{bmatrix} R^{B}_{TCP_f} & P^{B}_{TCP_f} \\ 0 & 1 \end{bmatrix} \tag{2.88}$$

$$T^{B}_{TCP_l} = \begin{bmatrix} R^{B}_{TCP_l} & P^{B}_{TCP_l} \\ 0 & 1 \end{bmatrix} \tag{2.89}$$

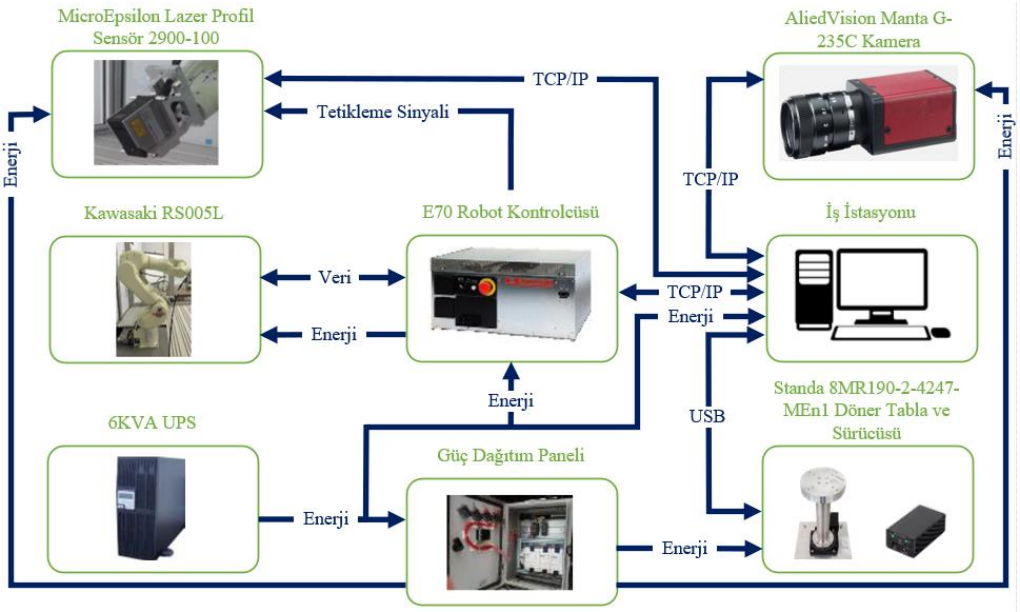
Elde robot işlem ucu bağlantı noktası ve robot taban koordinat sistemi arasında tanımlı olan döndürme matrisleri ve pozisyon vektörleri, ihtiyaç duyulan robot hareket planını tanımlamaktadır.

3. DENEYSEL ÇALIŞMALAR

Tez çalışması kapsamında geliştirilmiş olan sistemin gerçek bir uygulamada nasıl kullanılacağını ve bu kullanım sonucunda elde edilecek olan sonucun ne ölçüde başarılı olduğunun sergilenmesi gerekmektedir. Çalışma kapsamında iki endüstriyel parçanın birbirine kaynaklanması sürecinden sorumlu bir endüstriyel robot kolunun programlanması ele alınmıştır. Ayrıca, geliştirilen sistemin koşturulduğu donanım özellikleri ve sistemin detaylı yazılım tasarımı devam eden alt bölümlerde sunulmaktadır.

3.1. Donanım Yapısı

Deneysel çalışmalar için kullanılan endüstriyel robot hücresi ve bu hücre içerisinde yazılım tarafından kontrol edilen belli başlı donanımlar bulunmaktadır. Bu donanımlar endüstriyel işlemi gerçekleştirmekten sorumlu bir robot kolu, bu robot koluna bağlı bir lazer profil sensör, endüstriyel işlem görecek parçayı taşımaktan ve gerekli döndürme işlemlerini yapacak olan bir döner tabla, çalışma ortamını görüntüleyen ayrıca bu görüntüleri kullanıcı arayüzü vasıtası ile operatöre sunan bir kamera ve bu donanımların kontrol edilmesinden sorumlu bir bilgisayardan oluşmaktadır. Bu bilgisayar aynı zamanda, tez kapsamında geliştirilmiş olan sistem ve arayüzü çalıştırarak operatör etkileşimlerinin yürütülmesinden sorumludur. Deneysel çalışmalar kapsamında kullanılan sistem üzerinde bulunan donanımlar arasındaki haberleşme hatlarını ve enerji beslemelerini gösteren bağlantı şeması Şekil 3.1'de sunulmaktadır. Ayrıca bu sistemin genel görünümü Görsel 3.1'de gösterilmektedir.



Şekil 3.1. *Donanım Yapısı*

Endüstriyel işlemin gerçekleştirilmesi ve işlem gerçekleştirilecek bölge için üç boyutlu modelin elde edilmesinde kullanılan lazer profil sensörünün hareket ettirilmesi için kullanılan sistemde bir adet 6-döner eksenli Kawasaki RS005L endüstriyel robot bulunmaktadır. Bu robot en fazla 5 kg yük taşıma kapasına sahiptir. En fazla erişebileceği uzaklık 903 mm olan bu endüstriyel robotun istenilen pozisyon için takip mesafesi 0.03 mm'dir. Bilgisayar ile arasındaki haberleşme robot kontrolcüsü üzerinden eternet hattı üzerinden sağlanmaktadır. Ayrıca, robotun bilgisayardan gelen komutlara cevap vermesini sağlamak amacı ile robota ait programlama dili ile yazılmış olan bir yazılım çalıştırılmaktadır [57]. Bu yazılım sayesinde, bilgisayar yardımı ile kullanıcı (operatör) tarafından gönderilen pozisyon ve oryantasyon komutları robot tarafından gerçekleştirilebilmekte ve robotun anlık pozisyonu ve oryantasyonu takip edilebilmektedir.

Yüzey taraması ile elde edilen nokta bulutu Micro-Epsilon Inc tarafından üretilen scanControl 2900-100 serisi lazer profil sensörü kullanılarak elde edilmektedir. Bu sensör, optik üçgenleştirme yöntemini temel alarak çalışmaktadır. Tarama işlemi sırasında sensörün kontrolünde bulunan bir lineer optik sistem tarafından bir lazer çizgi, taranacak yüzeye yansıtılır. Yansıtılan lazer çizgi için geri yansıyan lazer ışını sensörün yüksek kaliteli optik algılayıcı sistemi ile iki boyutta geri oluşturulur. Oluşturulan geri yansıma verisi ve elde edilen yansıma noktaları ile ölçüm işlemi gerçekleştirilir. Ölçüm

işleminde elde edilen uzaklık bilgisinin yanı sıra (Z-ekseni), ilgili lazer çizgi üzerindeki her bir noktaya ait net pozisyon verisi de (X-ekseni) sistem tarafından sağlanmaktadır. Elde edilen ölçümler için en fazla nokta sayısı, her bir lazer çizgi (profil) için 1280 nokta olup bu noktalara ait ölçüm hassasiyeti 12 μmdir, ve etkili ölçüm mesafesi 190 mm ile 290 mm arasındadır. Sensör ayrıca dahili eternet arayüzü ile bilgisayar haberleşmesini desteklemektedir.

Endüstriyel robot işlem ucunun, nesnenin bütün pozisyonlarına erişememesinden dolayı ilgili nesneyi robota doğru çevirecek (robot kolunun nesne üzerinde ilgilenilen pozisyonlara erişebilmesini sağlayacak) Standa 8MR190-2-4247 model bir döner tabla, robot hücresi içerisinde konumlandırılmaktadır. Bu döner tabla, adım motor tahrikli olup, yüksek kararlılıkta ve 0,01 derece hassasiyette çalışmaktadır. Ayrıca döner tabla dönüşünü sağlayan adım motoru süren bir adet sürücü modülü bulunmaktadır. Bu modül kullanılarak USB bağlantısı üzerinden bilgisayar ile döner tabla haberleşmesi gerçekleştirilmektedir.

İşlem görecek nesneye ait ilgilenilen bölgenin, dünya koordinat sistemine göre nerede olduğunun tespit edilebilmesi için sistem içerisinde bir adet Alied Vision Manta G238C endüstriyel kamera kullanılmaktadır. Bu kamera sahip olduğu Sony IMX172, 1/1.2” CMOS sensör ile 1936x1216 piksel çözünürlükte saniyede 50 kare görüntü sağlayabilmektedir. Bilgisayar ile kamera iletişimi eternet bağlantısı ile gerçekleştirilmektedir.

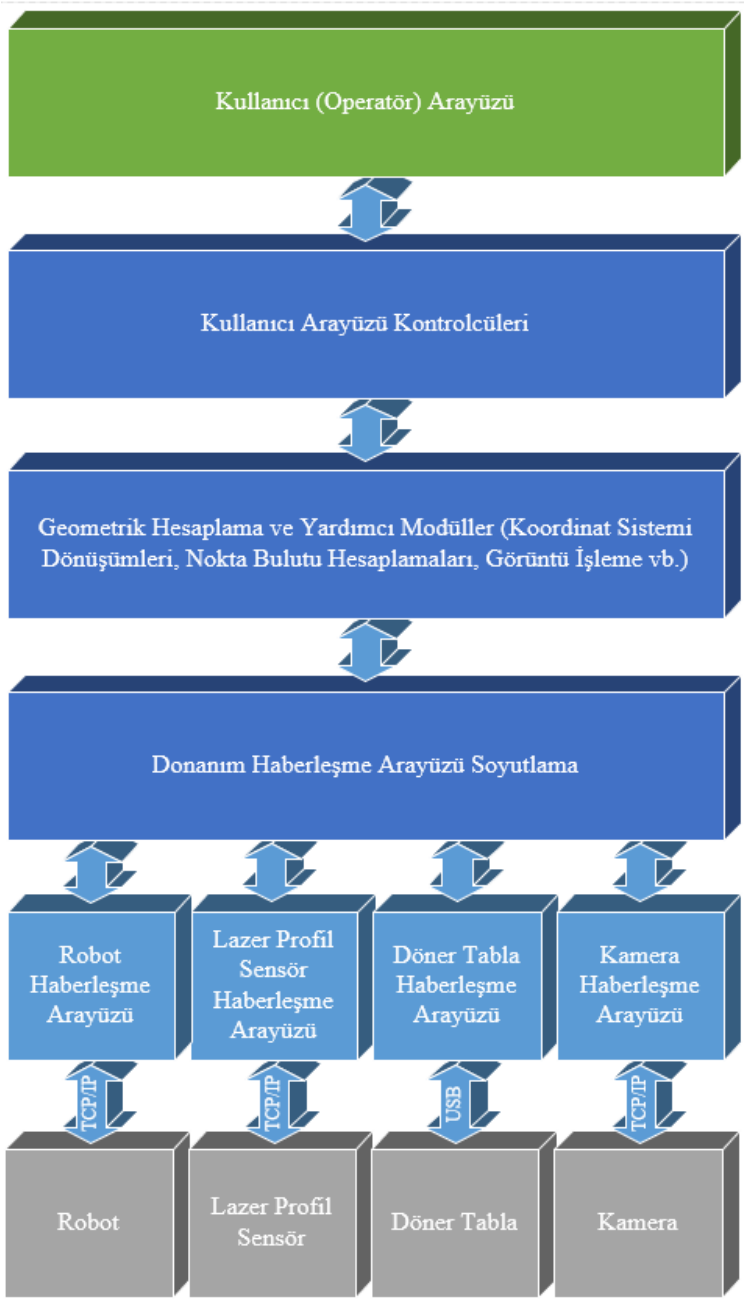
Geliştirilen sistem içerisinde kullanılan donanımlar arasındaki haberleşmeyi kontrol altında tutan ve operatör bilgisayar etkileşimini sağlayan arayüzü ve bu arayüzün kullandığı alt programlama paketlerini çalıştıran yüksek hesaplama gücüne sahip bir bilgisayara ihtiyaç duyulmaktadır. Tez çalışması için gerçekleştirilen deneysel çalışmalarda kullanılan bilgisayar HPZ840 modelinde bir iş istasyonu olup, üzerinde iki adet Xeon E5-2630 barındırmaktadır.

(a) (b)

Görsel 3.1. *Robot Hücresi Genel Görünümü (a): Uzak Çekim (b): Yakın Çekim*

3.2. Yazılım Mimarisi

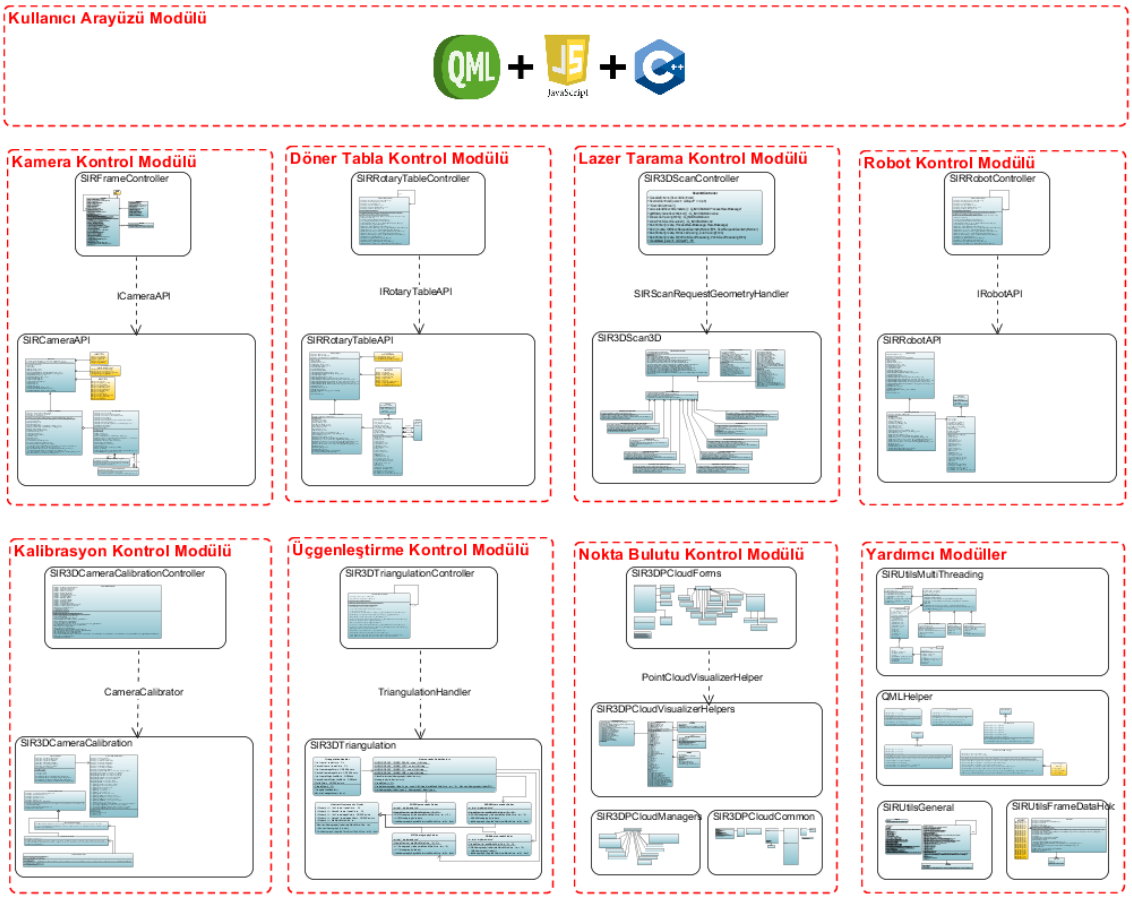
Endüstriyel robot hücresinde ihtiyaç duyulan donanımların (robot kolu, lazer profil sensör, döner tabla, kamera) verimli bir şekilde kontrol edilebilmesi ve bu donanımlar için geliştirilen yazılım paketlerinin yönetilebilir olması gerekmektedir. Bununla birlikte endüstriyel robot hücresi ve operatör arasındaki tek etkileşim aracı olan kullanıcı arayüzünün etkin bir şekilde kullanılabilmesi bu tez kapsamında önerilen yöntemin başarısını etkileyen hayati bir etkendir. Bu nedenle, geliştirilen yazılımın iyi tasarlanmış, geliştirmeye açık ve yönetilebilir olması gerekmektedir. Bu gereksinimin karşılanması geliştirilen alt paketlerde meydana gelen değişikliklerin bu paketleri kullanan katmanlara olan etkisini azaltılarak, modüler bir geliştirme yapısı ortaya çıkarılmasını sağlayacaktır. Şekil 3.2 bu tez kapsamında önerilen kolay endüstriyel robot programlama sisteminin yazılım mimarisinin katmanlara ayrılmış halini göstermektedir.



Şekil 3.2. *Yazılım Mimarisi*

Tez çalışması kapsamında geliştirilen sistemin Şekil 3.2'de verilen katmanlı yazılım mimarisinin gerçekleştirebilmesi için, bütün yazılımın modüllere ayrılması gerekmektedir. Bu modüllerin her biri kendi içerisinde tanımlanan işlemleri gerçekleştirmekle sorumlu olmakla birlikte diğer modüllerden gelen isteklere cevap verebilmelidir. Aynı zamanda herhangi bir modülün içyapısında gerçekleştirilen bir değişikliğin diğer modüllere olan etkisinin en aza indirilmesi gerekmektedir. Şekil 3.3'de

verilen genel yazılım diyagramı, tez kapsamında geliştirilen sistem içerisinde bulunan modülleri göstermektedir.



Şekil 3.3. *Genel Yazılım Diyagramı*

Tez kapsamında oluşturulan kullanıcı arayüzü dışındaki bütün modüllerin geliştirilmesinde C++ programlama dili tercih edilmiştir. Yazılımın yürütülmesi sırasında ihtiyaç duyulan hız ve harici donanım ile haberleşme gereksinimleri bu dilin seçilmesinde belirleyici rol oynamaktadır. Bununla birlikte C++ programlama dili için geliştirilmiş pek çok harici kütüphane bulunması da bu dilin tercih edilmesinde etkili olmaktadır. Aynı zamanda bu programlama dilinin kullanılması ile geliştirilen sistemin işletim sistemi bağımlılığı da genel olarak ortadan kaldırılmaktadır. Kullanım kolaylığı ve C++ programlama dili için standartta sunulmayan kullanıcı arayüzü geliştirme desteği sağlaması sebebi ile geliştirme ortamı olarak QtCreator [58] kullanılmaktadır. Kullanıcı arayüzü geliştirme sürecinde Qt yazılım geliştirme platformu tarafından yaratılmış olan QML [59] tercih edilmiştir. Bu dilin tercih edilmesinde belirleyici olan etken ise

geliştirilen arayüzlerin oldukça zengin görsellere ve animasyonlara sahip olabilmesi ve kendi içerisinde JavaScript kullanılabilmesi sayesinde operatör ile etkileşim sürecini modern bir sürece dönüştürebilmesidir. Şekil 3.2'de gösterilen katmanlı yazılım mimarisinin oluşturulabilmesi için ortaya çıkarılan modüller şu şekilde özetlenebilir:

Kamera Kontrol Modülü: Geliştirilen sistem için ihtiyaç duyulan kameranın bilgisayar ile haberleşme işleminin gerçekleştirilmesinden sorumlu modüldür. Bu modül yardımı ile geliştirilen sistemin operatöre sunacağı kamera görüntüleri elde edilmektedir.

Döner Tabla Kontrol Modülü: Endüstriyel robot hücresi içerisinde konumlandırılmış olan döner tablanın kontrolünü ve bilgisayar ile haberleşmesini sağlayan modüldür. Bu modül yardımı ile operatöre, döner tablayı istediği herhangi bir açı ve hız ile döndürebilmesi sağlanmaktadır.

Lazer Tarama Kontrol Modülü: Robot programlama işlemi için gerekli olan işlem görecek bölgeye ait nokta bulutunun elde edilmesi için gerçekleştirilecek lazer tarama işleminden sorumlu modüldür. Bu modül ile operatör tarafından belirlenen ilgilenen bölgeyi tanımlayan köşe noktaları için lazer tarama yörüngesi oluşturulup, lazer tarama işlemi gerçekleştirilir.

Robot Kontrol Modülü: Endüstriyel robot hücresi içerisindeki robot kolunun kontrolünden ve bilgisayar ile arasındaki haberleşme sürecinden sorumlu modüldür. Bu modül ile sistem içerisinde hesaplanan hareket planları robota aktarılarak, robotun belirlen bu yörüngeler boyunca hareket etmesi sağlanmaktadır.

Kalibrasyon Kontrol Modülü: Sistem kurulumu sırasında bir kez gerçekleştirilecek olan kalibrasyon sürecine ait matematiksel hesaplama işlemlerini gerçekleştirilen modüldür. Bu modül yardımı ile kalibrasyon işleminin yürütülmesi sağlanmakla birlikte elde edilen kalibrasyon verisi kayıt altına alınmakta olup, sistem içerisinde ihtiyaç duyulduğu anda sisteme sunulmaktadır.

Üçgenleştirme Kontrol Modülü: İlgilenilen bölgenin belirlenmesi sürecinde operatör tarafından belirlenen bölgenin üç boyutlu uzayda nerde olduğunun bulunmasından sorumlu modüldür. Bu modül ile bulunan bölge lazer taraması yapılacak bölgenin elde edilmesini sağlanmaktadır.

Nokta Bulutu Kontrol Modülü: Taranan bölgenin görüntülenmesinden ve robot programlama işlemi için ihtiyaç duyulan hareket yolunun oluşturulması için gerekli olan nokta bulutu işlemlerinin gerçekleştirilmesinden sorumlu modüldür. Bu modül ile

operatöre nesne üzerindeki ilgilenen bölgeye ait nokta bulutu üzerinde çalışma fırsatı sunulmaktadır.

Yardımcı Modüller: Geliştirilen yazılıma çoklu izlek desteği sağlanması, koordinat sistemleri arasındaki dönüşümlerin yapılması, elde edilen görüntüye ait hafıza alanının etkili bir veri tipinde saklanması gibi bütün yazılım sistemi içerisinde ihtiyaç duyulan destekleyici modüllerdir.

3.2.1. Donanım Haberleşme Arayüzü

Tez çalışması kapsamında oluşturulmuş ve Şekil 3.2'de verilmiş olan mimarinin en alt iki kaptanında, endüstriyel robot hücresinde kullanılan donanımlar (gri) ve donanımlar ile bilgisayar arasında iletişimi sağlayan haberleşme arayüzleri (açık mavi) bulunmaktadır. Burada gösterilmekte olan haberleşme arayüzlerinin her biri donanım üreticileri tarafından sağlanan farklı iç mimariye sahip yazılım geliştirme kütüphaneleridir. Tez kapsamında kullanılmakta olan bütün donanımlara ait haberleşme modülleri C++ programlama dilinde donanım üreticisi tarafından sağlanmaktadır.

3.2.2. Donanım Haberleşme Arayüzü Soyutlama

Endüstriyel robot hücresinin içerisinde bulunan donanımlar ve bilgisayar arasındaki haberleşmenin sağlanabilmesi için her bir donanıma ait uygulama programlama arayüzleri, donanım üreticileri tarafından sağlanmaktadır. Bu programlama arayüzleri ilgili donanım ile bilgisayar haberleşmesini sağlanması ve donanımın bilgisayardan gönderilen talepler ile istenilen işlemi gerçekleştirilmesini sağlamayı hedeflemektedir. Fakat her bir donanımın kendine özgü bir programlama arayüzü barındırması, genel kapsamlı birbirinden farklı donanımların bulunduğu karmaşık bir sistem için uygulama geliştirme sürecini oldukça kapsamlı ve yönetilmesi zor bir yazılım geliştirme sürecine dönüştürmektedir. Bununla birlikte, kullanılan herhangi bir donanımın değiştirilmesi veya bu donanımın üreticisi tarafından sağlanan programlama geliştirme arayüzünün güncellenmesi durumunda, bu haberleşme arayüzlerinin doğrudan kullanılarak geliştirilmiş kapsamlı bir sistemin yeniden düzenlemesi ihtiyacı ortaya çıkmaktadır. Karşılaşılan bu sorunun üstesinden gelmek için donanıma bağımlı olan ve üretici

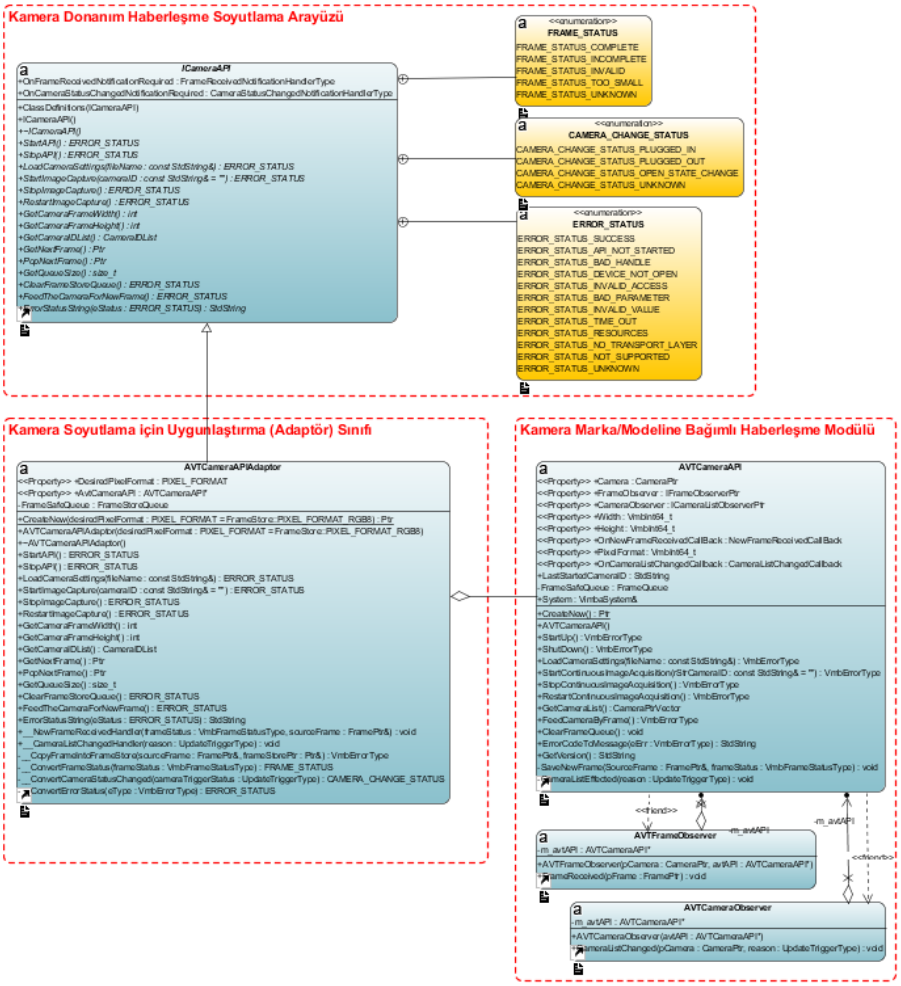
tarafından sağlanan bu programlama paketlerinin, ana uygulama yazılım sisteminden soyutlanması uygun bir çözüm olacaktır. Soyutlama işlemi ile donanıma bağımlı programlama arayüzlerinin yönetimi, geliştirilmesi veya değiştirilmesi durumlarında ana yazılım sisteminde değişiklik yapılması ihtiyacı ortadan kaldıracaktır.

Robot hücresi içerisinde kullanılan donanımların (kamera, döner tabla, lazer profil sensör, robot kolu) bilgisayar kontrolü ile yönetilmesi işleminin ana sistem yazılımından soyutlanabilmesi için, donanıma bağımlı olan modülün, bir başka uyumlaştırıcı modül altında saklanması ve bu saklama işlemini gerçekleştiren modülün, ana yazılım sistemi için ortak erişime sahip olan bir başka modülle bağdaşlaştırılması gerekmektedir. Ortak erişime sahip olan bu modül sayesinde, ana yazılım sistemi, kullanılan donanımın ve/veya ne olduğunu bilmeksizin, donanım üzerinde işlem yürütme yeteneğine sahip olacaktır.

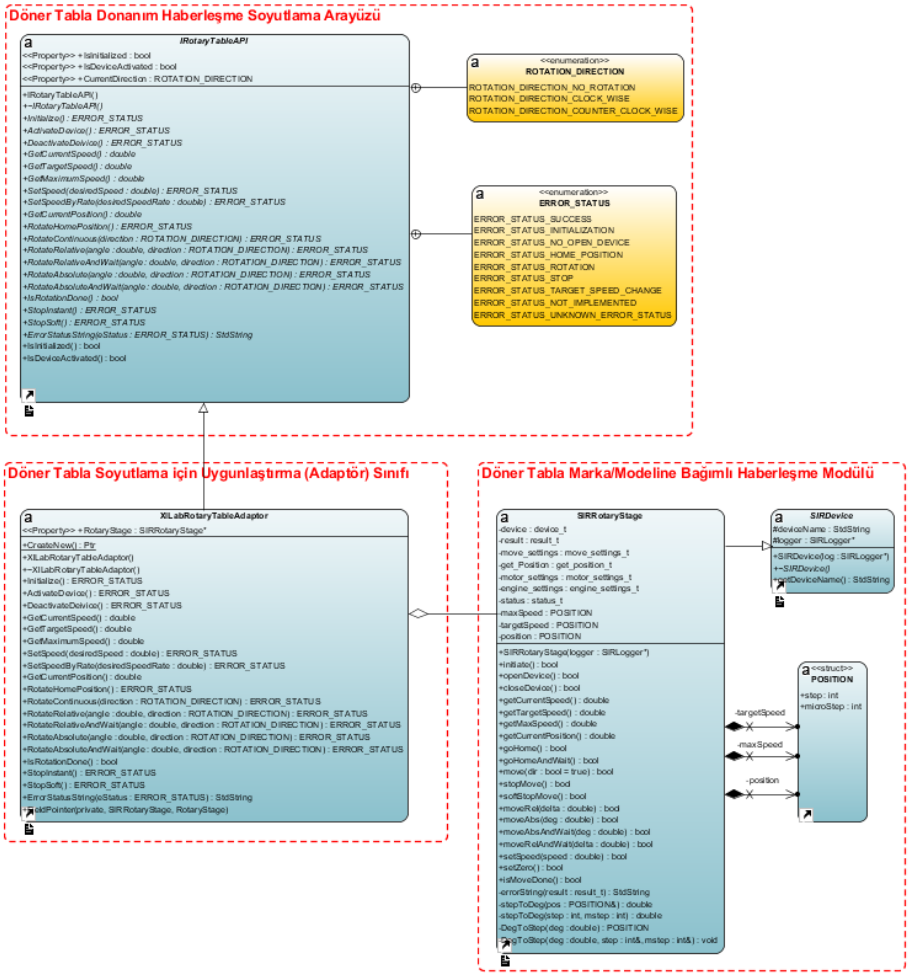
Şekil 3.4 ve Şekil 3.5 kamera ve döner tablanın ana sistem yazılımından soyutlanabilmesi için uygulanan tasarımı göstermektedir. Burada ana sistemden soyutlanmak istenen donanıma bağımlı haberleşme modülü ve soyutlama işlemi için gerekli olan uygunlaştırma (adaptör) sınıf ve bu sınıfın türetildiği ve ana sistem tarafından kullanılan soyutlama sınıfı ve ilgili modülü gösterilmektedir.

Kameraya ait donanıma bağımlı haberleşme modülü içerisinde bulunan bütün sınıflar ve metotlar, kameranın bilgisayar üzerinden kontrol edilebilmesini ve kameradan görüntü alınmasını sağlamaktadır. Fakat bu modül içerisindeki bütün yazılım sistemi, kullanılan kameraya özgüdür. Bu modül doğrudan ana sistem tarafından kullanılabilir. Fakat kameranın değişmesi veya haberleşme için kullanılan ve kamera üreticisi tarafından sağlanan düşük seviyeli programlama arayüzünün güncellenmesi durumunda, ana sistem içerisinde bu modülü kullanan her yerin yeniden yazılması gerekmektedir. Bu durumun ortadan kaldırılması için donanıma bağımlı olan bu modül, ana sistem için doğrudan erişilebilir olan bir soyut sınıftan türetilmiş olan bir uygunlaştırma sınıfı altında saklanmaktadır. Ana sistem yalnızca soyutlama arayüzünü kullanarak yürütülmek istenen işlem ile ilgili talepte bulunmaktadır. İşlem talebine verilen cevap ise adaptör sınıfı tarafından oluşturulmaktadır. Adaptör sınıfı ise gerçekleştirilen bu talebe verilen cevabı, donanıma bağımlı olan modülü kullanarak elde etmektedir. Böyle bir yapıda, kamera modülüne ait haberleşme yazılımında ortaya çıkan değişikliklerden yalnızca uygunlaştırma modülü sorumlu olacaktır. Ana sistem yazılımının bağımlılığı soyutlama arayüzü ile sınırlıdır. Benzer şekilde farklı bir marka/model kamera kullanılması durumunda ise farklı bir donanım haberleşme modülü kullanılması gerekliliği ortaya

çıkacaktır. Bu durumda ise, yine aynı soyutlama arayüz sınfından türetilmiş ve yeni kamera için sunulan haberleşme modülünü kendi içinde saklayan yeni bir adaptör sınıfı yazılması yeni kameranın kolaylıkla kullanılabilmesine olanak sağlayacaktır. Ana sistem yazılımı bu donanım değişikliğinden etkilenmeyecektir. Aynı sınıf tasarım yapısı diğer donanımlar (döner tabla, lazer profil sensör, endüstriyel robot kolu) içinde uygulanmaktadır.



Şekil 3.4. *Kamera Donanımı Haberleşme Soyutlama Tasarımı*



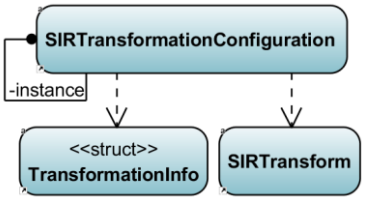
Şekil 3.5. *Döner Tabla Donanımı Haberleşme Soyutlama Tasarımı*

3.2.3. Geometrik Hesaplama ve Yardımcı Modülleri

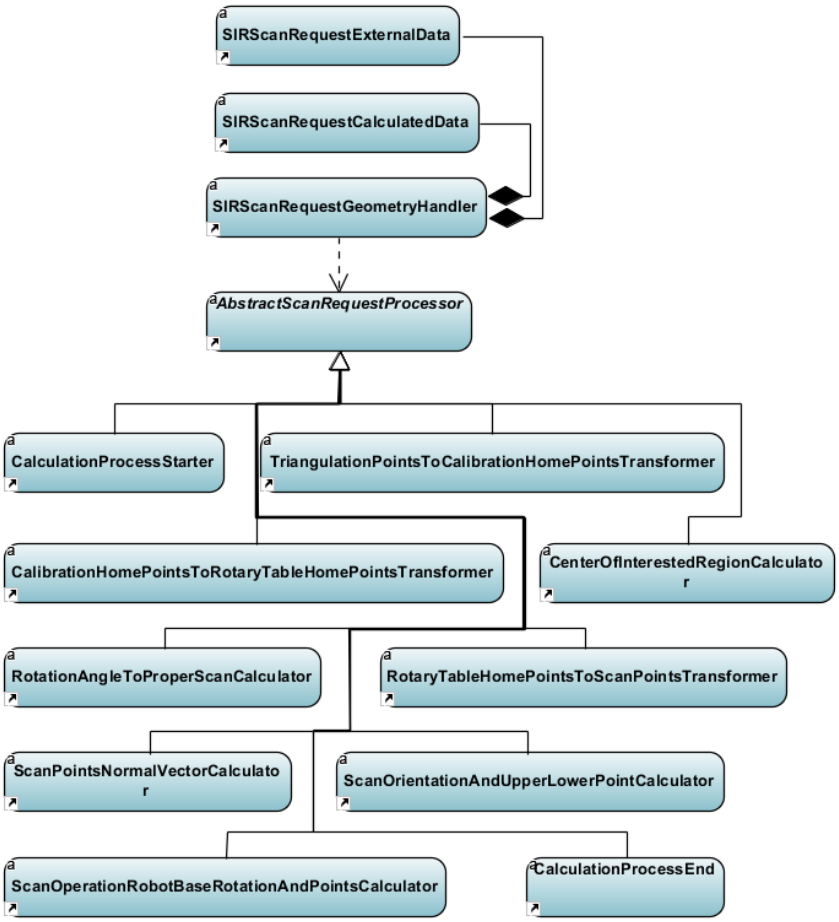
Tez çalışması kapsamında oluşturulmuş olan yazılımın, donanımdan alınan ham veriyi analiz ederek operatöre sunulabilecek hale getiren yazılım geliştirme paketlerinin bulunduğu ya da operatörün, kullanıcı arayüzü üzerinden sağladığı veriler için hesaplama taleplerinin karşılandığı katmandır. Bu katmanda bulunan yazılım geliştirme paketleri, yöntem bölümünde matematiksel modeller ve sahte kodlarla verilmiş olan adımların gerçek kodlamalarını barındırmaktadır. Bu katmanın diğer katmanlardan ayrılması ile, bu katmanda bulunan paketleri kullanan kullanıcı arayüzü kontrolcüleri katmanının bu katmanda gerçekleştirilen değişikliklerden en az seviyede etkilenmesi sağlanmaktadır. Kullanıcı arayüzü kontrolcü katmanından bu katmana ait bir pakete yapılan veri işleme

talebinin, ilgili paketin içeriğinin güncellenmesinden (yöntemin değiştirilmesi, yeni yöntem eklenmesi vb.) dolayı oluşacak değişikliklerden en az seviyede etkileneceğinden kullanıcı arayüzü kontrolcü katmanında önemli bir değişiklik yapılmasına gerek kalmayacaktır. Böyle bir yapı, bütün sistemi katmanlara ayırma ile hedeflenen yazılımın geliştirilmeye açık ve yönetilebilir olması hedefini doğrudan etkilemektedir

Geometrik hesaplama modülleri genel olarak sistem içerisinde kullanılan koordinat sistemlerinin birbirleri arasındaki dönüşümlerini gerçekleştirmekten sorumludur. Bu modüllerin yardımı ile yöntem sürecinde değinilen pek çok dönüşüm işlemi, modüler bir yapıda gerçekleştirilmektedir. Şekil 3.6'da verilen sınıf diyagramı koordinat sistemleri dönüşümlerini gerçekleştirmekten sorumlu modüldür. Bununla birlikte operatör tarafından belirlenen ilgilenilen bölgeye ait lazer tarama işlemi için gerekli olan robot hareket planının oluşturulmasından sorumlu lazer tarama robot hareket planı hesaplama modülleri de kullanıcı arayüzü kontrolcüleri tarafından kullanılan geometrik hesaplama modüllerindendir. (Bkz. Şekil 3.7).



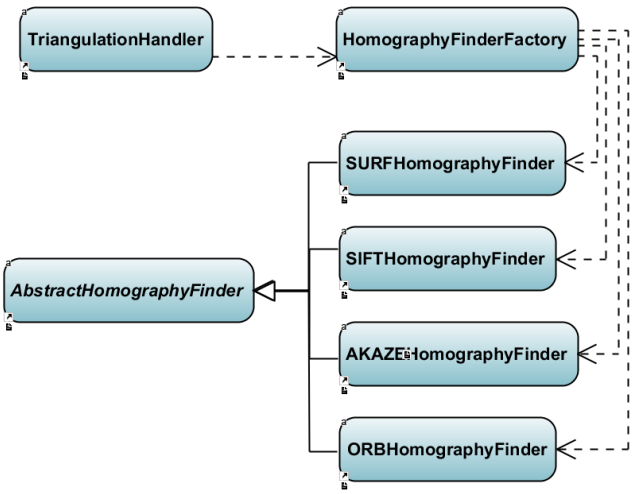
Şekil 3.6. *Koordinat Sistemleri Dönüşüm Modülü*



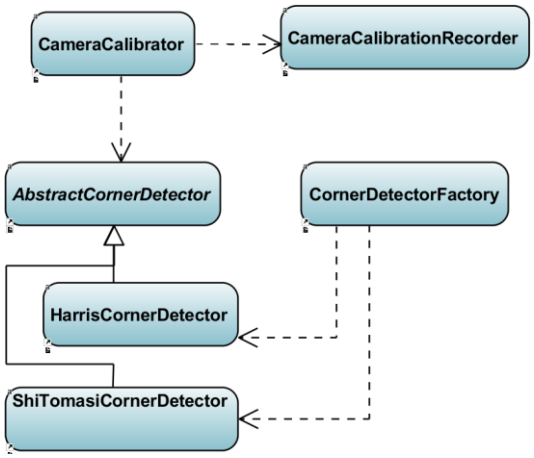
Şekil 3.7. *Lazer Tarama Robot Hareket Planı Oluşturma Modülü*

Tez kapsamında geliştirilen sisteme ait pek çok yardımcı modül ise kullanıcı arayüzü ve operatör etkileşimi ile tetiklenen hesaplama süreçlerinin yürütülmesinden sorumludur. Bu modüllerden bazıları şöyle sıralanabilir:

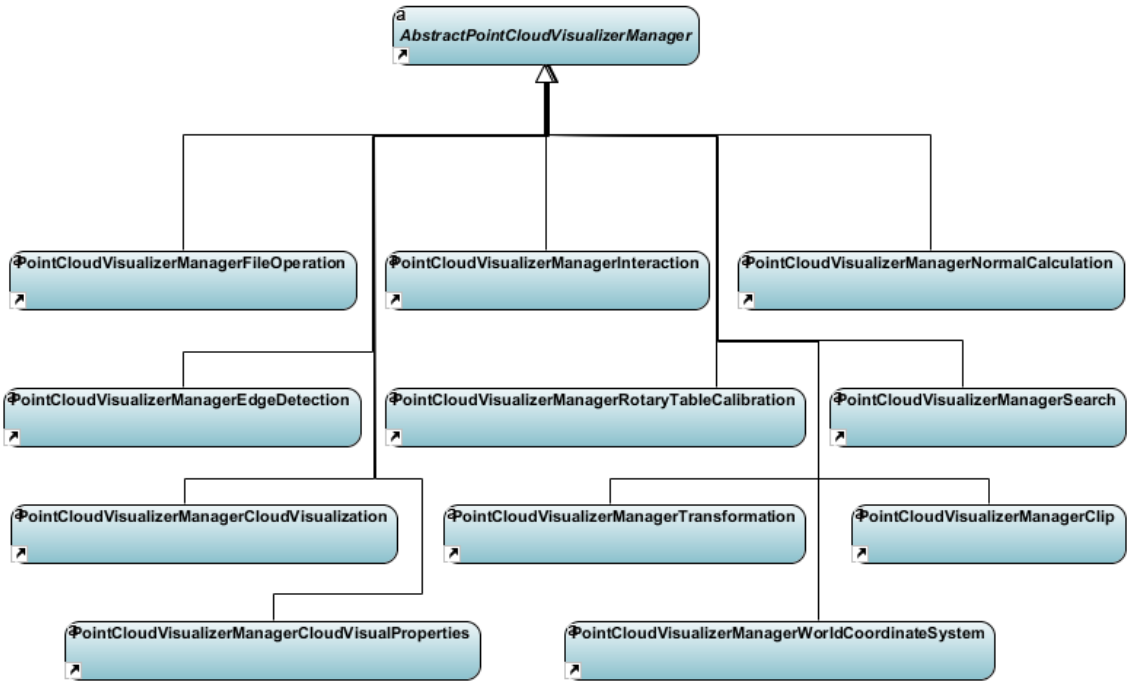
- Operatör tarafından belirlenen ilgilenilen bölgenin üç boyutlu uzaydaki konumunun tespit edilmesinden sorumlu olan üçgenleştirme modülü (Bkz. Şekil 3.8)
- Kamera kalibrasyon modülü (Bkz. Şekil 3.9)
- Elde edilen nokta bulutu üzerinde gerçekleştirilecek olan nokta bulutu işlemlerini yürütmekten sorumlu olan nokta bulutu hesaplama ve yönetme modülü (Bkz. Şekil 3.10)



Şekil 3.8. *Üçgenleştirme Modülü*

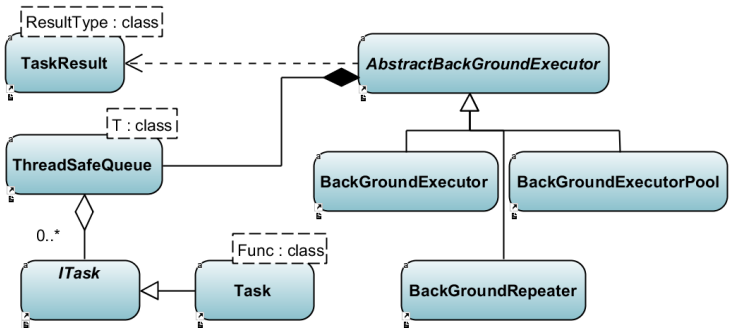


Şekil 3.9. *Kamera Kalibrasyon Modülü*



Şekil 3.10. *Nokta Bulutu Hesaplama ve Yönetim Modülü*

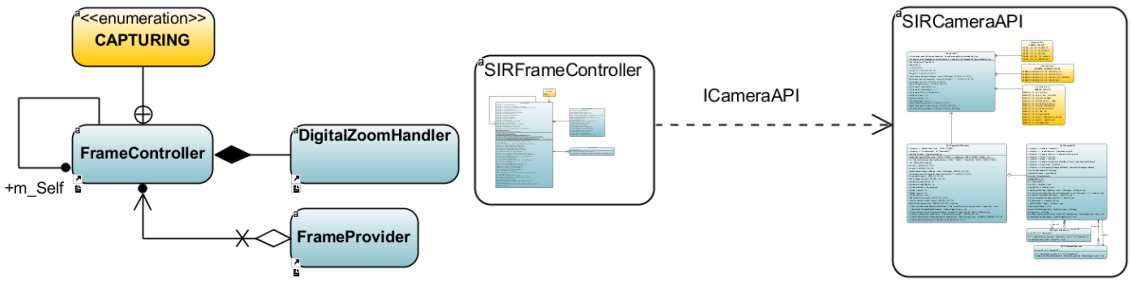
Sistem üzerinde yürütülen pek çok işlem ağır hesaplama yüküne sahip işlemlerdir. Bu işlemlerin kullanıcı arayüzünü yürüten ana izlek üzerinde çalıştırılması, kullanıcı arayüzünün geçici süreliğine cevap veremez hale gelmesine sebep olacaktır. Bununla birlikte yürütülen işlemlerin farklı izleklerde paralel olarak yürütülmesi geliştirilen yazılımın çalışma performansını önemli ölçüde artıracaktır. Bu nedenle yönetilmesi kolay bir izlek yönetim modülüne ihtiyaç duyulmaktadır. Tez kapsamında tasarlanan ve kullanılan izlek yönetim modülü Şekil 3.11'de gösterilmektedir.



Şekil 3.11. *İzlek Yönetim Modülü*

3.2.4. Kullanıcı Arayüzü Kontrolcüleri

Operatör ve sistem etkileşimini sağlayan kullanıcı arayüzü ile geometrik hesaplama ve yardımcı modüller arasına yerleştirilmiş olan bu katman, kullanıcı arayüzü etkileşim taleplerini karşılayarak operatör tarafından sağlanan veriyi ilgili alt pakete aktarmaktan ve gereken hesaplama işleminin gerçekleştirilmesini sağlamaktan sorumludur. Bu katman ile kullanıcı arayüzü üzerinde gerçekleştirilen görsel tasarıma dayalı güncelleme işlemlerinden kaynaklanan değişikliklerden alt katmanların soyutlanması sağlanmaktadır. Kamera kontrolü ve kameradan alınan görüntünün kullanıcı arayüzünde görüntülenmesi sürecini yönetmekten sorumlu olan görüntüleme kontrolcüsü Şekil 3.12'de görüntülenmektedir. Bu kontrolcü donanım ile haberleşme işlemini, Şekil 3.4'de gösterilen *Kamera Donanım Haberleşme Soyutlama Arayüzü*nü kullanarak gerçekleştirmektedir. Bu sayede bu kontrolcü üzerinde gerçekleştirilen görüntüleme işleminin donanım bağımlılığı, ilgili bölümde anlatıldığı üzere ortadan kaldırılmaktadır. Benzer yapı sistem üzerinde kullanılan diğer kontrolcü modüllerde de kullanılmaktadır.



(a) (b)

Şekil 3.12. *Görüntüleme Kontrolcü Modülü*

3.2.5. Kullanıcı Arayüzü

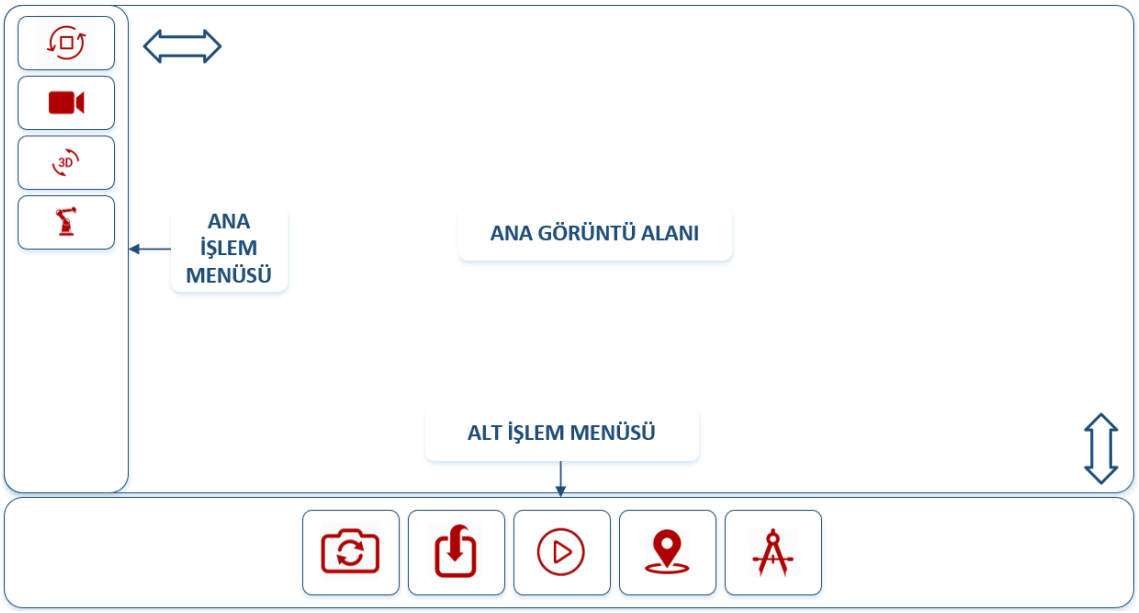
Operatör ile sistem etkileşimini sağlayan görsel katmandır. Bu katman ile operatör tarafından sağlanan girdilerin ya da komutların ilgili kullanıcı arayüzü kontrolcüsüne aktarılması sağlanır. Ayrıca donanımdan alınan verinin nasıl ve nereden elde edileceği ya da işlem uygulanmış veri sonucunda oluşan çıktının kullanıcıya nasıl sunulacağını bu katman ile belirlenmektedir. Bu katman yalnızca ilgili kontrolcü ile etkileşim halindedir. Bu sayede, kullanıcı arayüzünün, ana sistem yazılımının diğer katmanları ile olan ilişkisi

ortadan kaldırılmaktadır. Şekil 3.13 tez kapsamında tasarlanan QML tabanlı kullanıcı arayüzünün genel tasarımını göstermektedir. Yöntem bölümünde verilen pek çok örnek görsel aynı zamanda kullanıcı arayüzünü sergilemektedir.

Ana işlem menüsü operatör tarafından gerçekleştirilecek işlemin ait olduğu ana bölümü temsil etmektedir. Örneğin, operatör kamera ile ilgili bir işlem yürütecek ise öncelikle kameraya ait ana işlem menüsünü seçmesi gerekmektedir.

Alt işlem menüsü, ana işlem menüsünde gerçekleştirilen seçimle ilintili olan alt işlem seçeneklerini görüntülemektedir. Kamera operasyonları ile ilgili alt işlem menüsünün açılmış olduğu durumda, operatör açılan alt işlem menüsünü kullanarak, kamerayı başlatma, durdurma, kamera ayarlarını yükleme, kamera kalibrasyonu gibi işlemleri yürütebilmektedir.

Kullanıcı arayüzünün merkezinde konumlanmış olan ana görüntü alanı ise kameradan alınan anlık görüntülerin sergilendiği alandır. Bu alan üzerinde operatör ilgilenilen bölgeyi belirleme, kamera kalibrasyonu vb. işlemleri yürütmektedir. Ayrıca sistemin çalışması sırasında oluşan hatalar ya da bilgilendirme mesajları bu alanda gösterilmektedir.

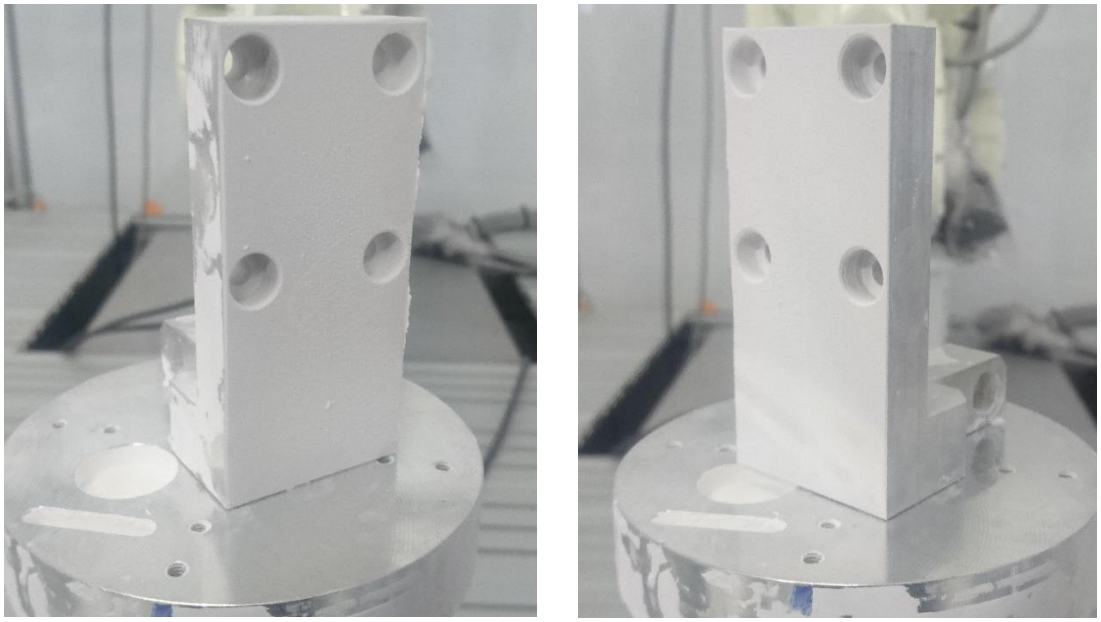


Şekil 3.13. *Kullanıcı Arayüzü Genel Tasarımı*

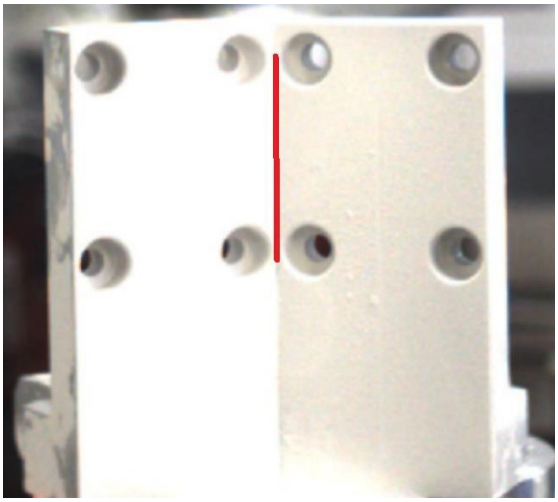
3.3. Deneysel Çalışma ve Değerlendirme

Tez kapsamında geliştirilen sistem kullanılarak gerçekleştirilen endüstriyel robot programlama sürecinin tamamı yöntem bölümünde anlatıldığı gibi kullanıcı arayüzü üzerinden yürütülmektedir.

İki endüstriyel parçayı düz bir çizgi hattı boyunca birbirine sabitlemek için kaynak işlemi gerçekleştirmekten sorumlu bir endüstriyel robotun programlanması süreci, geliştirilen yöntemin akış sürecini özetlemek için oldukça uygun bir çalışmadır. Deneysel çalışma kapsamında endüstriyel robotun bağlantı ucuna lazer profil sensör ile birlikte monte edilmiş işlem aparatının, programlama sürecinde elde edilen yolu takip ettirilmesi hedeflenmiştir. Bu bağlamda kaynak uygulanacak endüstriyel parçalar Görsel 3.2'de sergilenmektedir. Birbirinin aynısı olan bu iki parça kenar bölgelerinden birbirine tutturularak, vida deliklerinin bulunduğu hat boyunca kaynak işlemine tabi tutulacaktır (Bkz. Görsel 3.3).

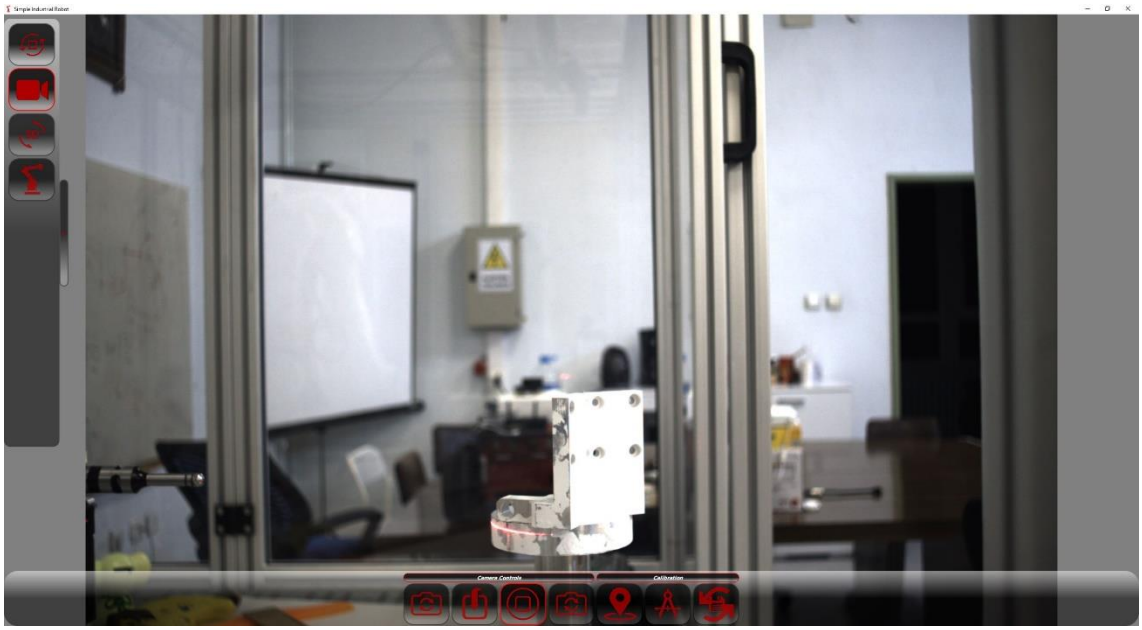


Görsel 3.2. *Kaynak Uygulanacak Parçalar*

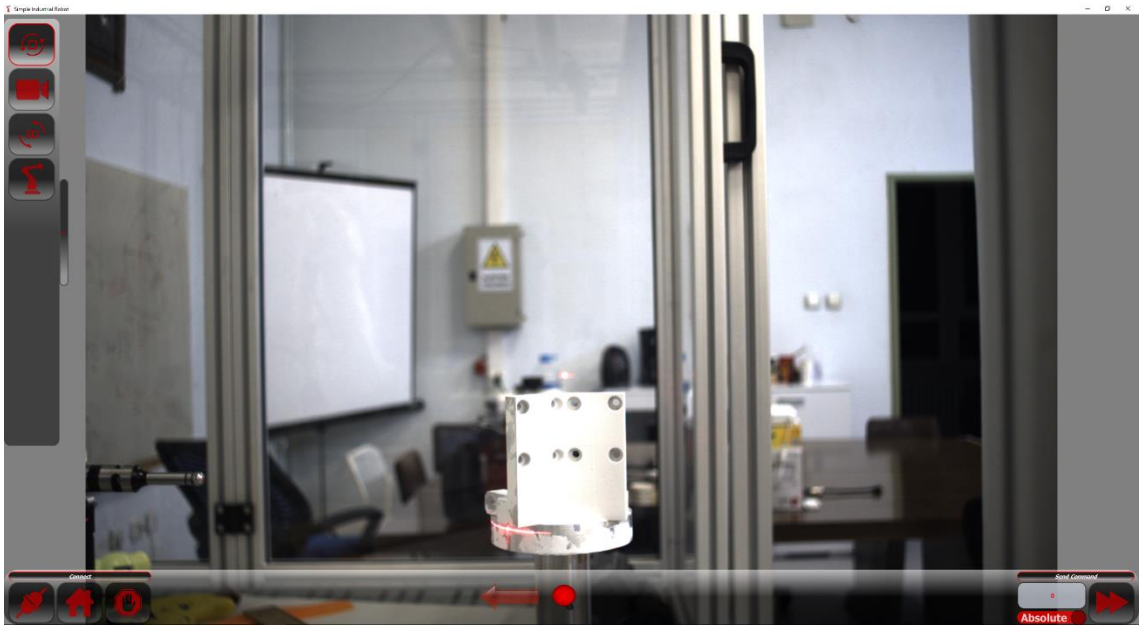


Görsel 3.3. *Parçaların Kaynaklanma Pozisyonu ve Kaynak Uygulanacak Hat*

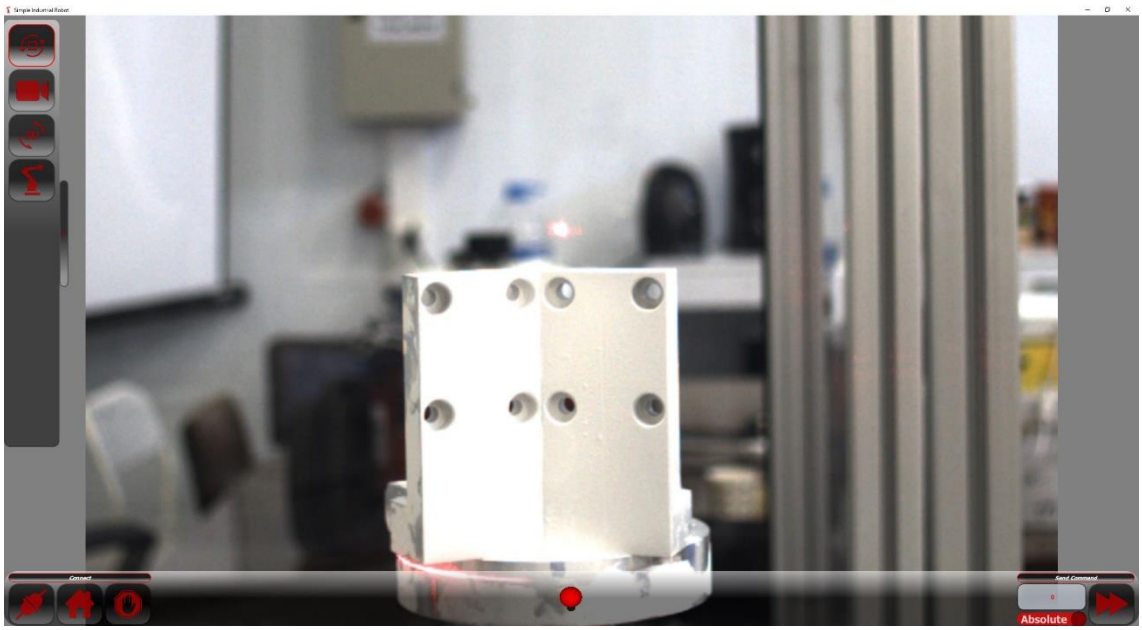
Kaynak yapılacak parçaların döner tabla üzerine işlem gerçekleştirilmek üzere yerleştirilmesinin ardında programlama süreci başlatılmaktadır. Başlangıç olarak bu parçaların sistem üzerinde kullanılan arayüz üzerinde gerçek zamanlı olarak görüntüleme sürecinin başlatılması gerekmektedir (Bkz. Görsel 3.4). Gerçek zamanlı görüntüleme işleminin başlatılmasının ardından kaynak işlemi için ilgilenilen bölgenin kamera görüş alanına sokulması sağlanır. Bunun için döner tablanın operatör tarafından kullanıcı arayüzü kullanılarak döndürülmesi (Bkz. Görsel 3.5) ve dönüş sonrası elde edilen durum için görüntü üzerinde yakınlaştırma işlemi uygulanması gerekmektedir (Bkz. Görsel 3.6).



Görsel 3.4. *Sürecin Gerçek Zamanlı Görüntülenmeye Başlatılması*

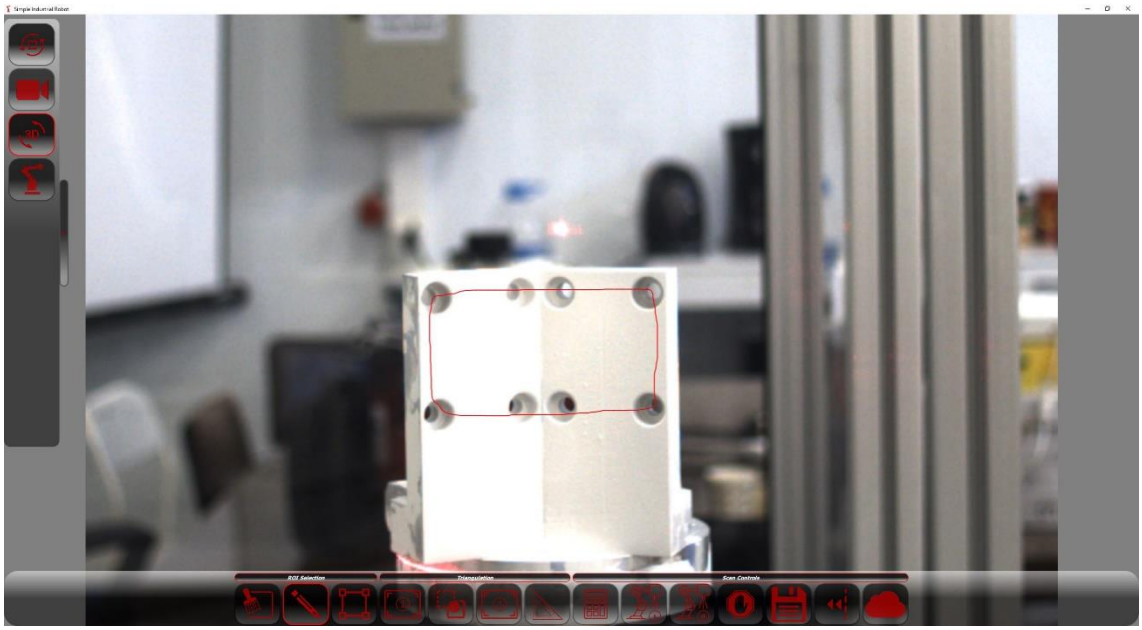


Görsel 3.5. *İlgilenilen Bölgenin Görüntülenmesi için Döner Tablanın Döndürülmesi*

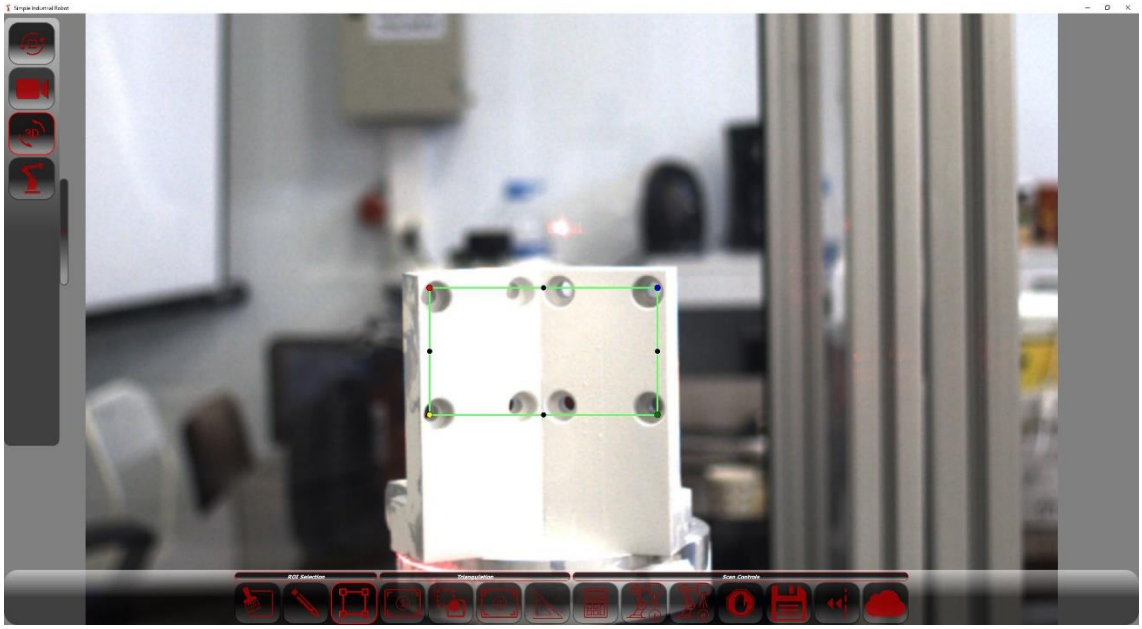


Görsel 3.6. *Yakınlaştırılmış Görüntü*

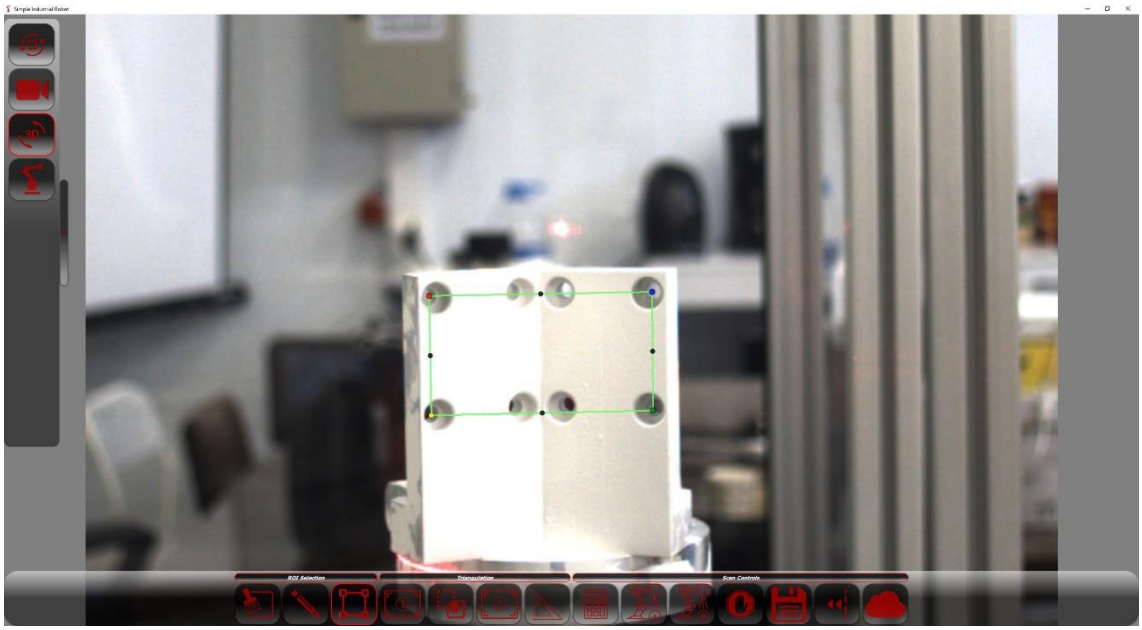
Kamera görüş alanına sokulmuş ve gerekli yakınlaştırma işlemi uygulanmış görüntü kullanılarak kaynak işlemi uygulanacak bölge belirlenir (Bkz. Görsel 3.7, 3.8, 3.9).



Görsel 3.7. *Serbest Çizim İşlemi ile İşlem Görecek Bölgenin Çizimi*

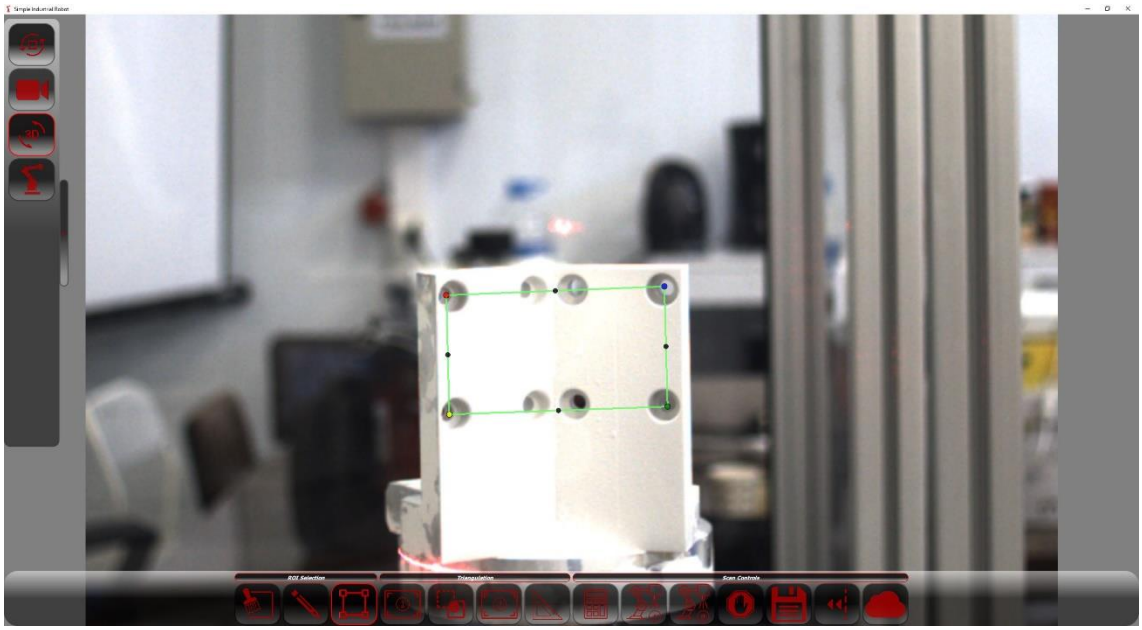


Görsel 3.8. *Çizim İşlemi ile Elde Edilen Bölgeye Ait Çevreleyen Dörtgen*

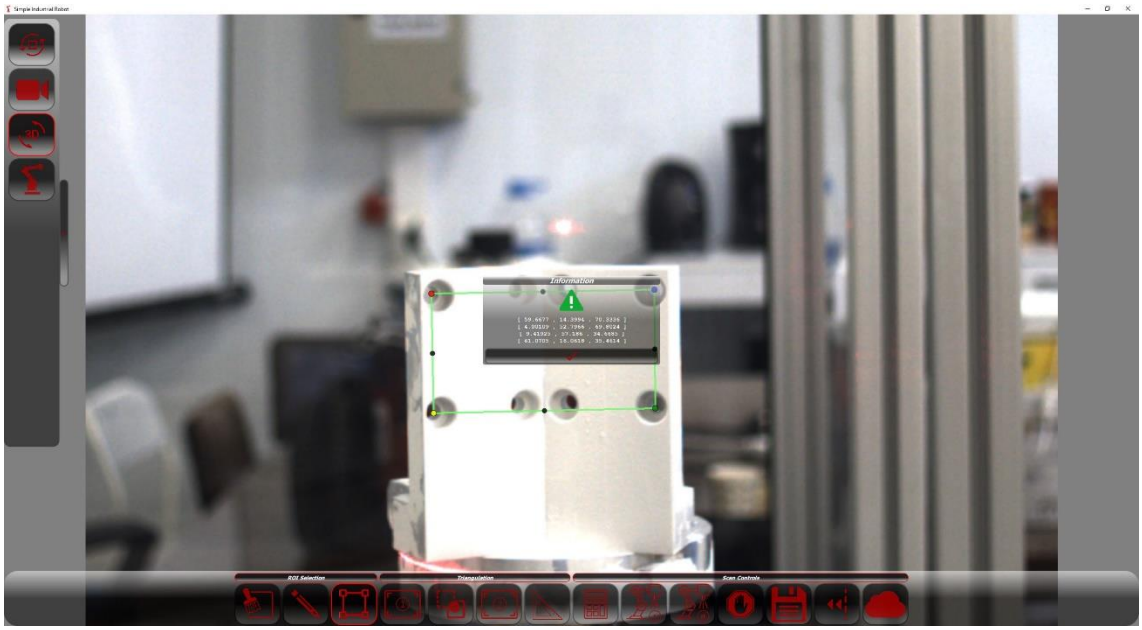


Görsel 3.9. *Çevreleyen Dörtgen için Operatör Müdahalesi*

Elde edilen ilgilenilen bölge dörtgenin üç boyutlu uzaydaki karşılığının bulunması için ikinci bir görüntü üzerinde bu dörtgenin sistem tarafından yeniden oluşturulması gerekmektedir. (Bkz. Görsel 3.10, 3.11).

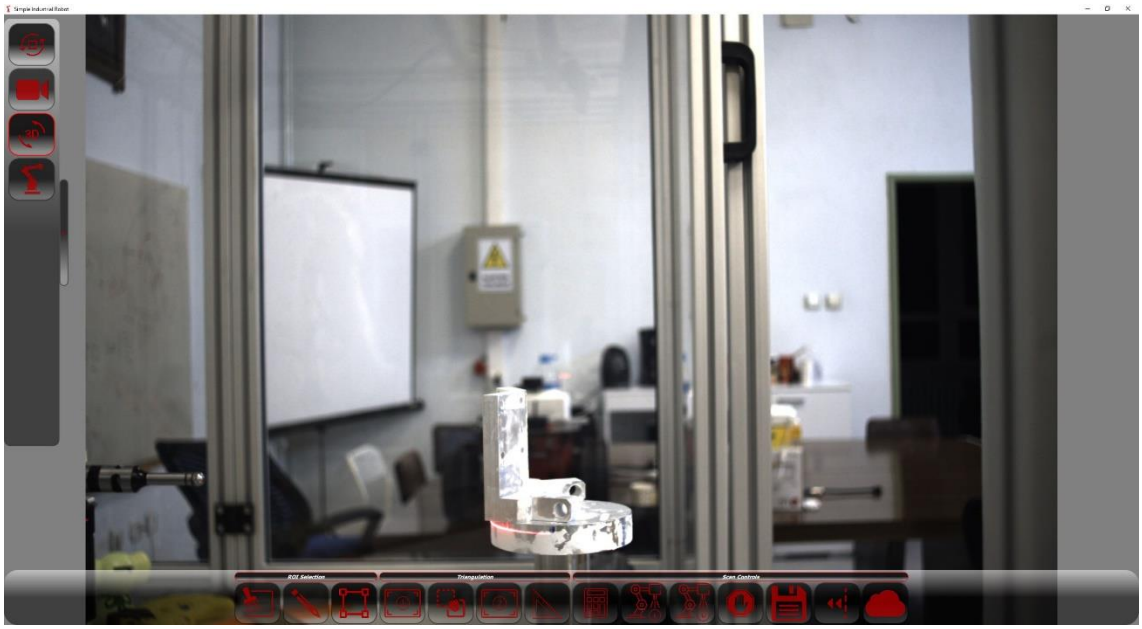


Görsel 3.10. *İkinci Görüntü Üzerinde Elde Edilen Çevreleyen Dörtgen*

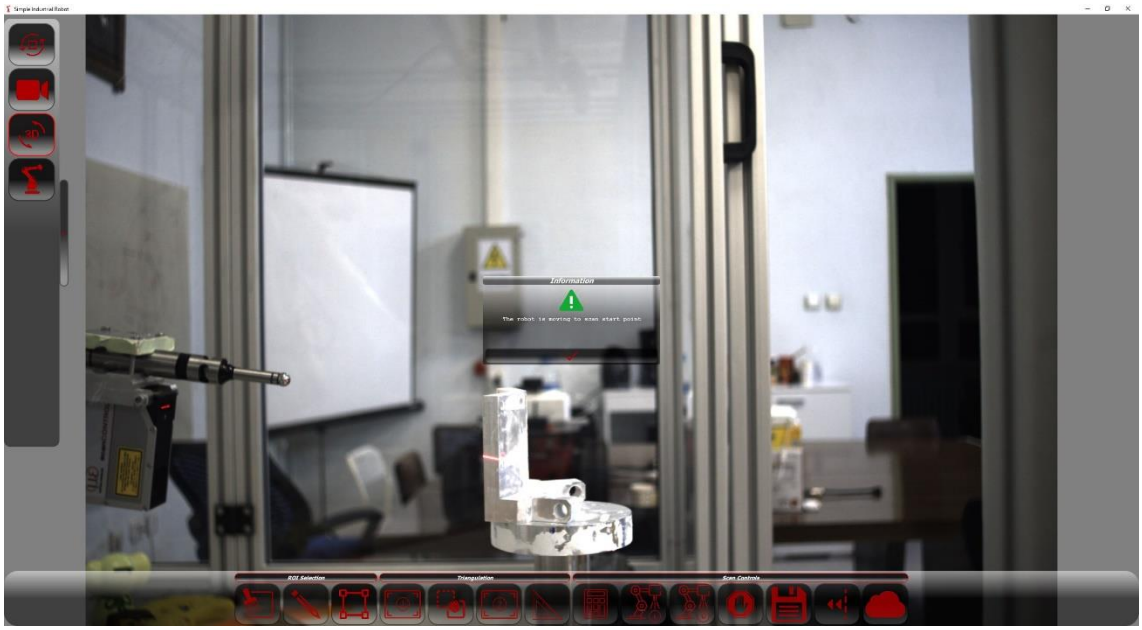


Görsel 3.11. *Üçgenleştirme Yöntemi ile Elde Edilen 3B Köşe Koordinatları*

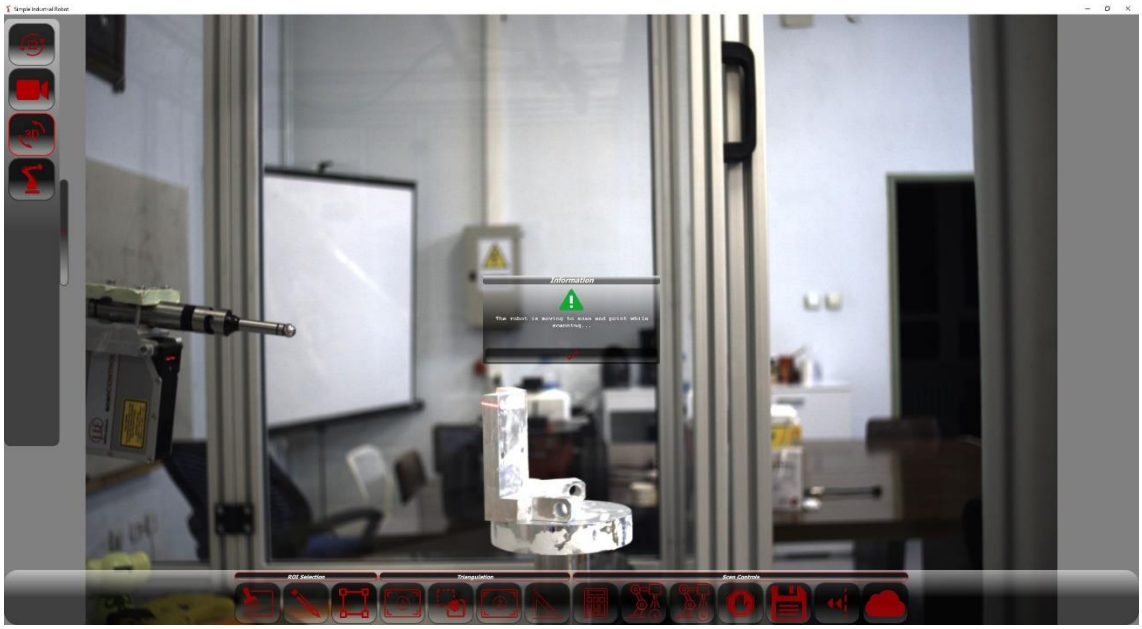
Üç boyutlu uzaydaki karşılığı tespit edilen ilgilenilen bölgenin üç boyutlu modelinin oluşturulması için lazer tarama işlemi gerçekleştirilecektir. Bu işlem için endüstriyel parçanın ilgili bölgesinin robota doğru yönlendirilmesi ve robot hareket planının oluşturulması gerekmektedir (Bkz. Görsel 3.12, 3.13, 3.14).



Görsel 3.12. *İlgilenilen Bölgenin Robota Döndürülmesi*

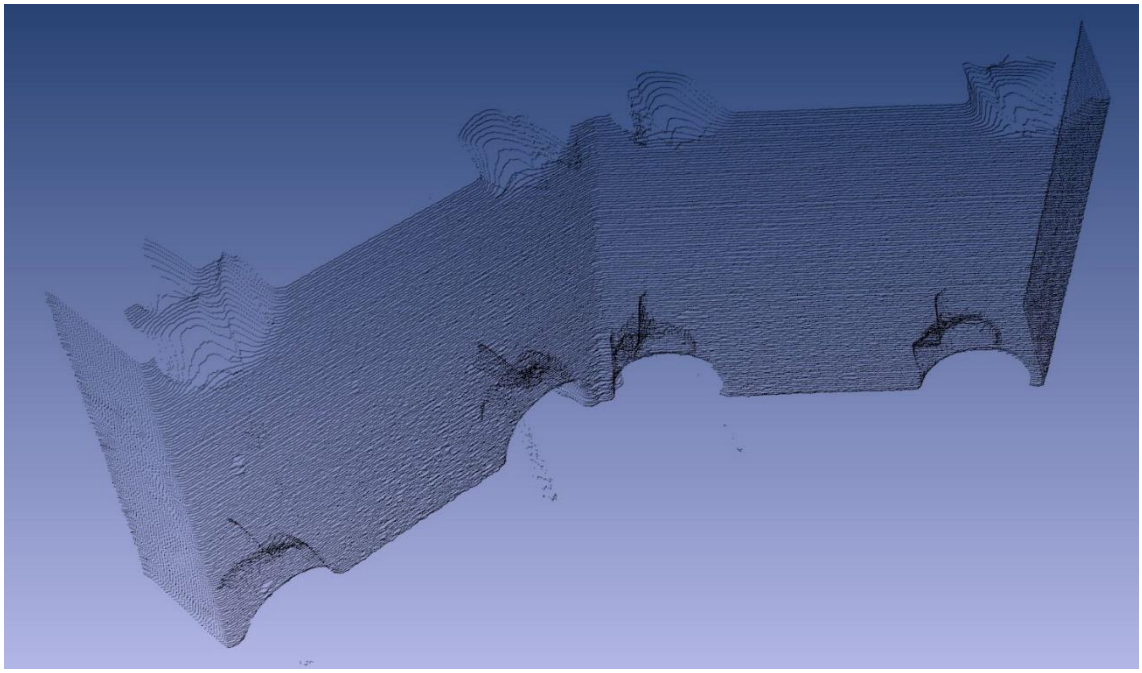


Görsel 3.13. *Lazer Profil Tarama İlk Pozisyonuna Robotun Gönderilmesi*



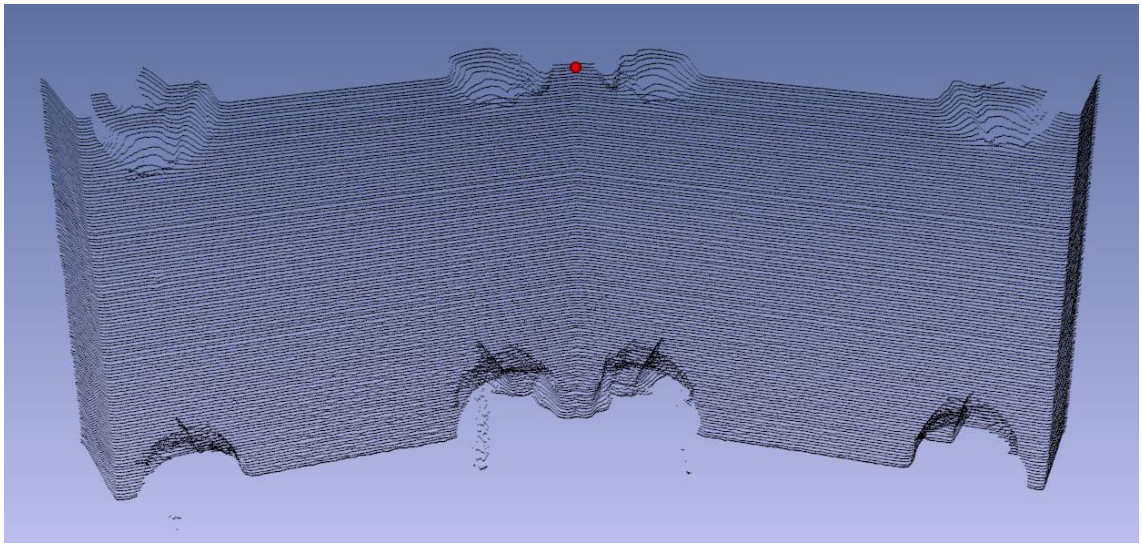
Görsel 3.14. *Lazer Tarama Son Pozisyonuna Robotun Gönderilmesi*

Lazer profil sensörü ile gerçekleştirilen tarama işlemi sonrası elde edilen nokta bulutu dosyaya kaydedilmektedir. Kaydedilen bu nokta bulutu ilgilenilen bölgenin üç boyutlu modelini oluşturmaktadır (Bkz. Görsel 3.15).

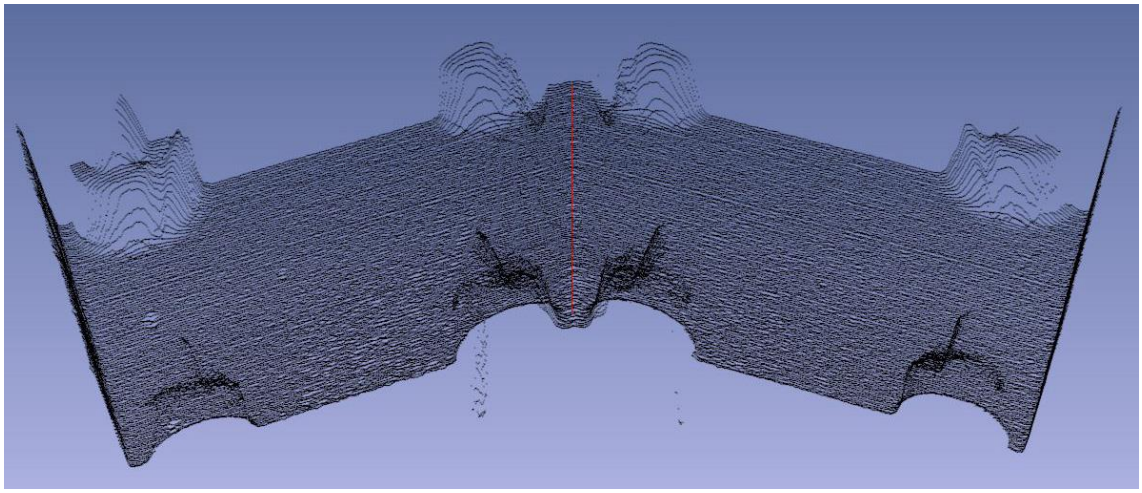


Görsel 3.15. *Lazer Profil Tarama İşlemi ile Elde Edilen Nokta Bulutu*

Elde edilen nokta bulutu ile operatör gerçekleştirilecek kaynak işleminin başlangıç ve bitiş noktaları operatör tarafından tanımlanmaktadır (Bkz. Görsel 3.16, 3.17).

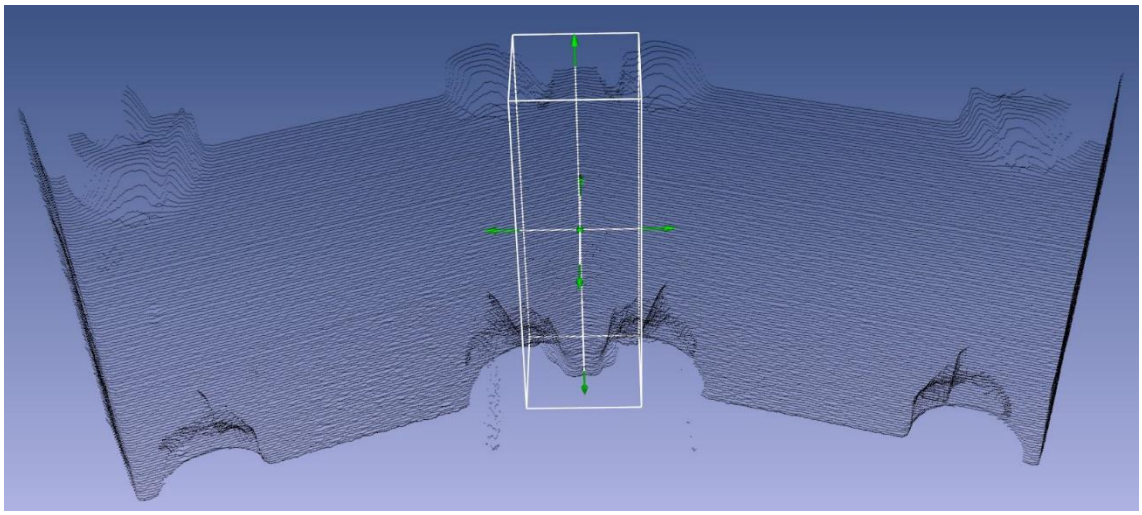


Görsel 3.16. *Kaynak İşlemi Başlangıç Noktası*

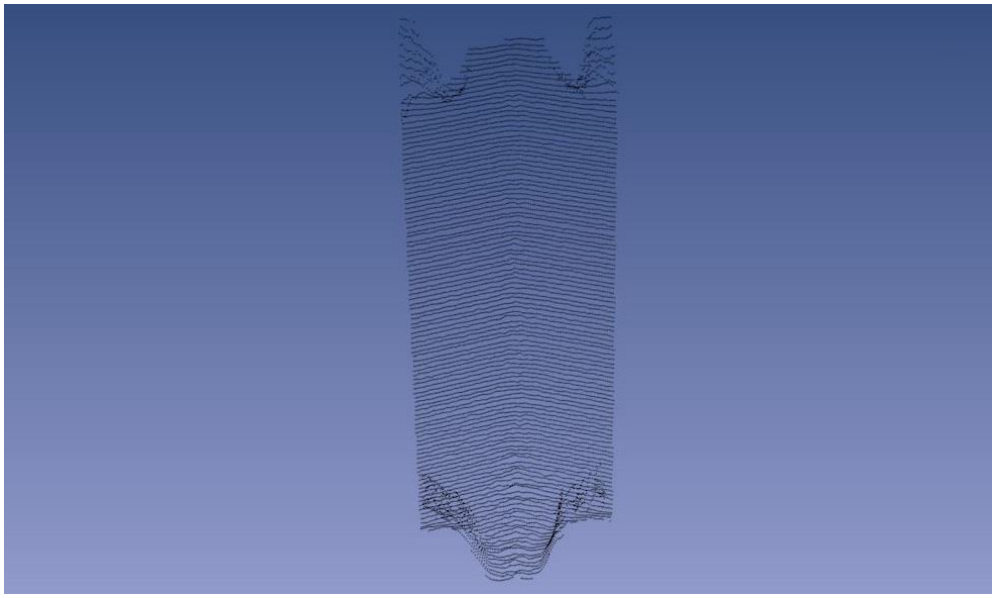


Görsel 3.17. *Kaynak İşlemi Bitiş Noktası ile Edilen Kullanıcı Tanımlı Kaynak Hattı*

Kullanıcı (Operatör) tarafından tanımlanan kaynak hattının hataya açık olması sebebi ile belirlenen bu hattın nokta bulutunun geometrik özellikleri kullanılarak düzeltilmesi gerekmektedir. Bu işlem için harcanan zamanın azaltılması için nokta bulutu üzerinde işlem gören nokta sayısının azaltılması gerekmektedir. Bu nedenle tanımlanan hat üzerinde sistem tarafından bir kırma kutusu oluşturulup nokta bulutunun kırpılması sağlanır (Bkz. Görsel 3.18, 3.19).

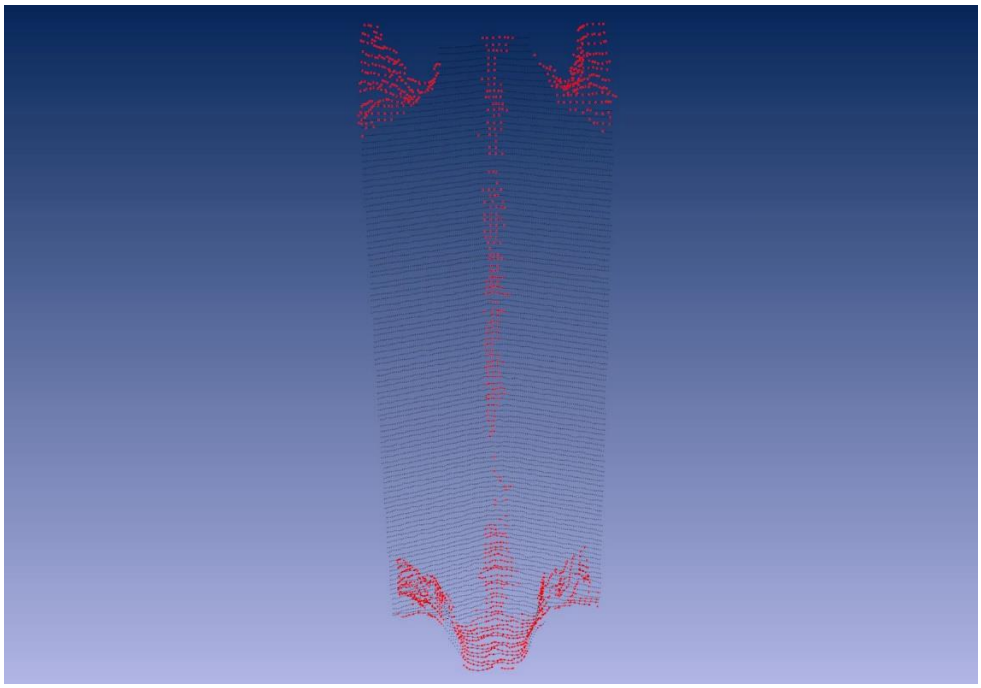


Görsel 3.18. *Kullanıcı Tanımlı Kaynak Hattı Üzerinde Oluşturulan Kırma Kutusu*



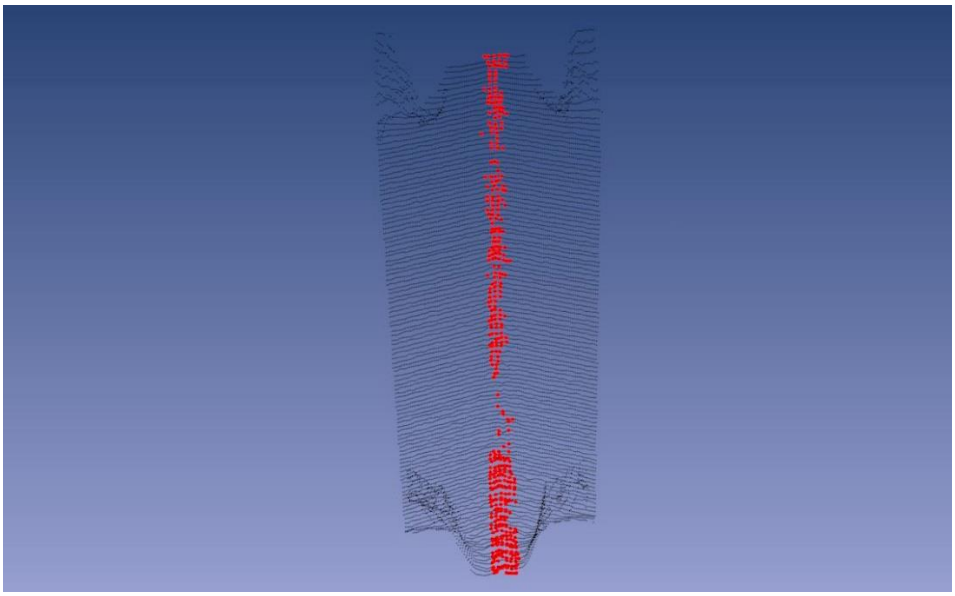
Görsel 3.19. *Kırpılmış Nokta Bulutu*

Kırpılmış olan bu nokta bulutu üzerinde bulunan kenar noktaları ilgilenilen kaynak hattını tanımlayan nokta kümesini barındırmaktadır. Bu kenar noktalarının sistem tarafından tespit edilip kullanıcıya sunulmaktadır (Bkz. Görsel 3.20).



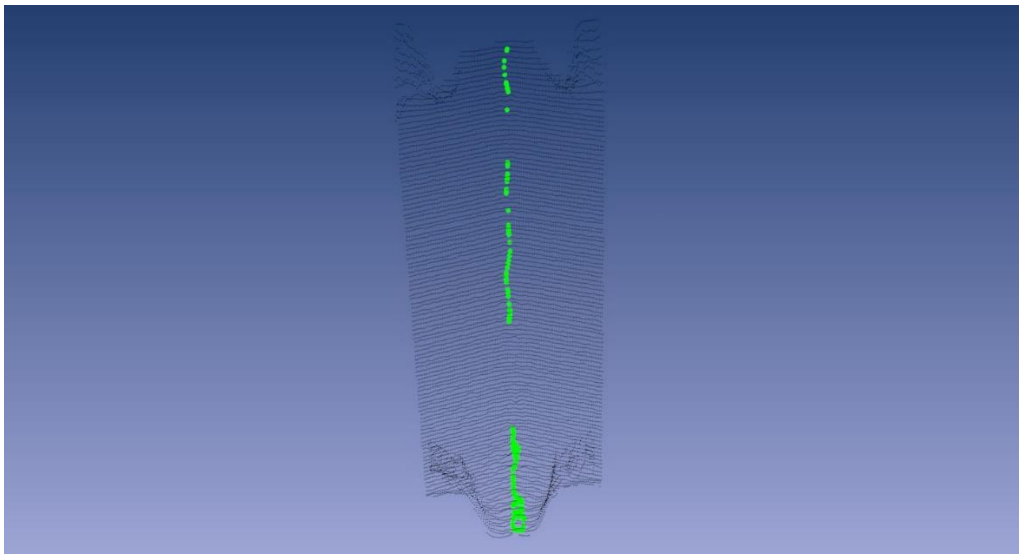
Görsel 3.20. *Nokta Bulutu Üzerindeki Kenar Noktaları*

Elde edilen kenar noktaları içerisinde tespit edilecek kaynak hattının yanı sıra, endüstriyel parçaların üzerinde bulunan vida deliklerine ait kenar noktaları da tespit edilmiştir. Bu noktaların kırpılarak atılması gerekmektedir (Bkz. Görsel 3.21).



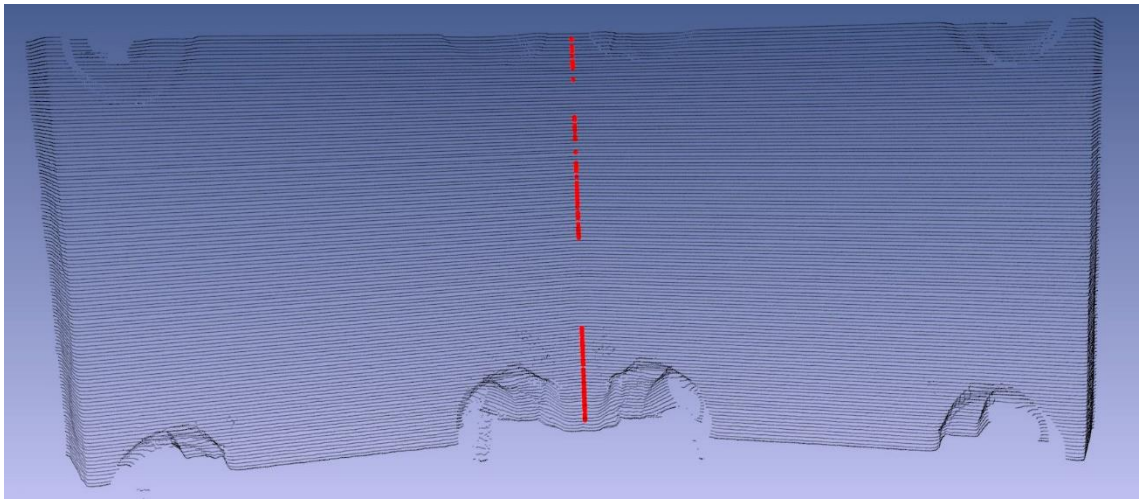
Görsel 3.21. *Kırpılmış Kenar Noktaları*

Kaynak hattını içerisinde barındıran kenar noktaları ihtiyaç duyulandan çok daha fazla yoğunlukta nokta barındırmaktadır. Bu nedenle bu noktaların seyreltilmesi sağlanmaktadır (Görsel 3.22).

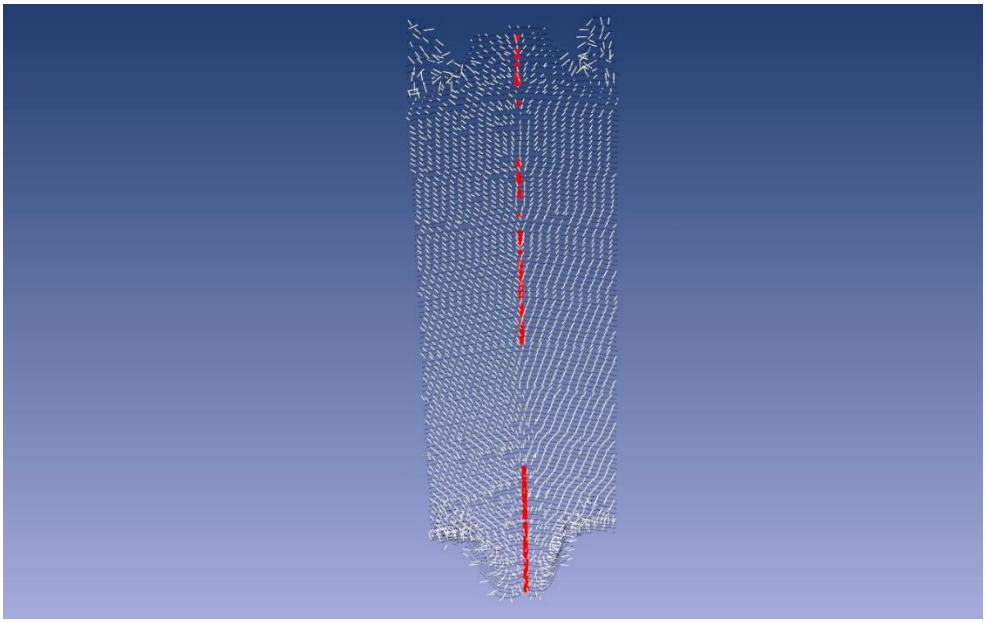


Görsel 3.22. *Seyreltilmiş Kenar Noktaları*

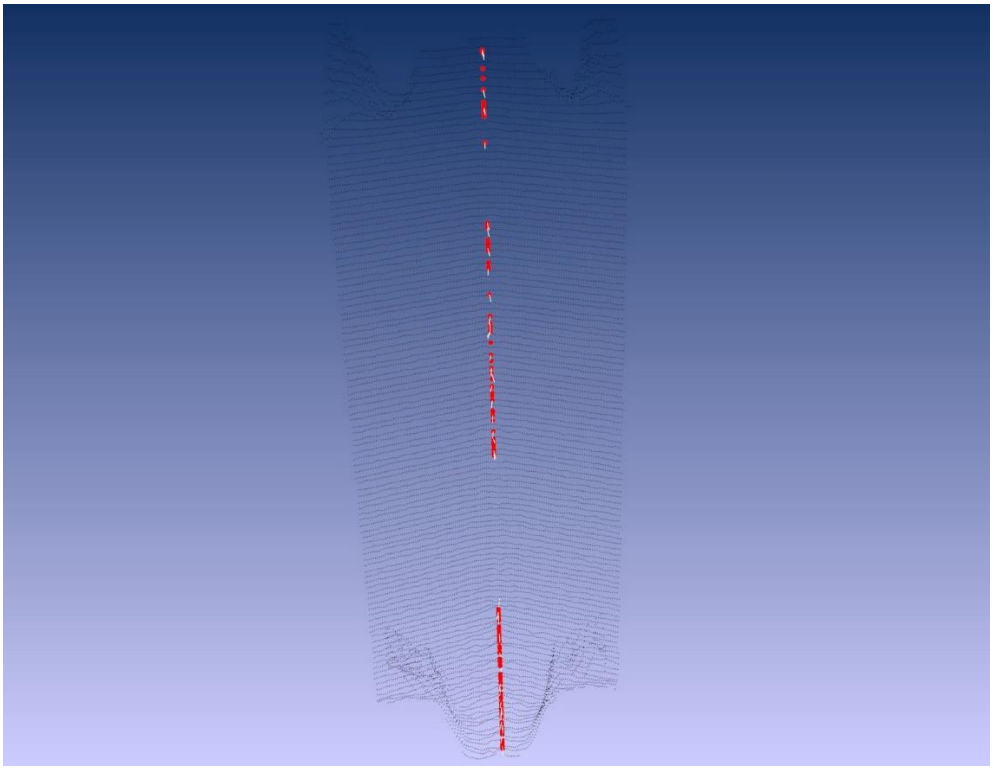
Seyreltilen kenar noktalar kullanılarak robotun gerçekleştireceği kaynak işlemi için izleyeceği yolu tanımlayan çizgisel hat elde edilebilir. Bunun için bu noktalar üzerinde çizgi uydurma işlemi uygulanır (Bkz. Görsel 3.23). Elde edilen bu çizginin matematiksel modelinin bilinmesi ile kullanıcı tarafından belirlenen başlangıç ve bitiş noktalarının iyileştirilmesi sağlanmaktadır. Bu sayede kullanıcı seçimi sırasında ortaya çıkabilecek seçim hataları ortadan kaldırılıp, robot hareket planı için ihtiyaç duyulan pozisyon bilgisi elde edilmektedir. Robot hareket planı için ihtiyaç duyulan bir diğer parametre ise robot yönelim bilgisidir. Bu bilgi ise kaynak işlemi için izlenecek yola ait normal vektörler kullanılarak elde edilmektedir. Bu nedenle kaynak hattını tanımlayan kenar noktaları için uydurulmuş olan çizgisel nokta bulutuna ait bir normal vektörünün sistem tarafından tespit edilmesi gerekmektedir (Bkz. Görsel 3.24, 3.25, 3.26).



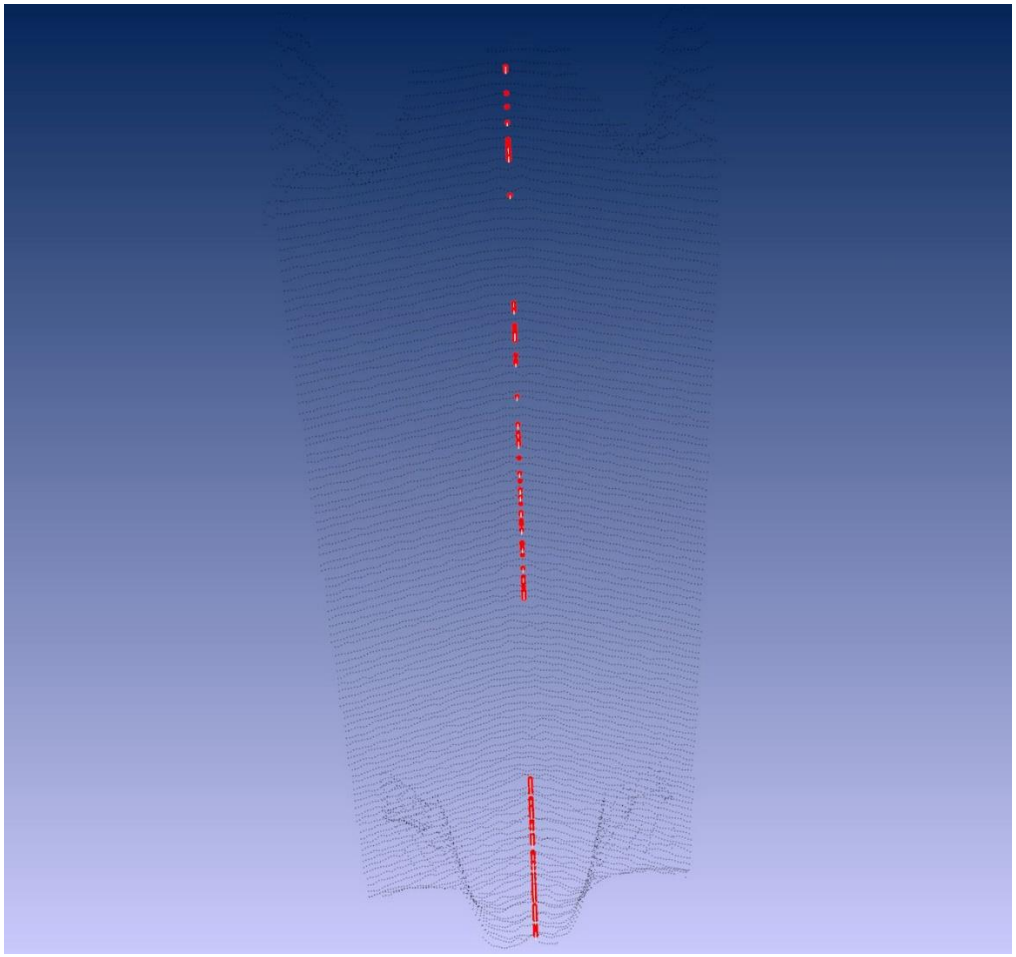
Görsel 3.23. *Çizgi Uydurma İşlemi ile Elde Edilen Robot Hareket Yolu*



Görsel 3.24. *Yüzey Nokta Bulutuna ait Normal Vektörler*

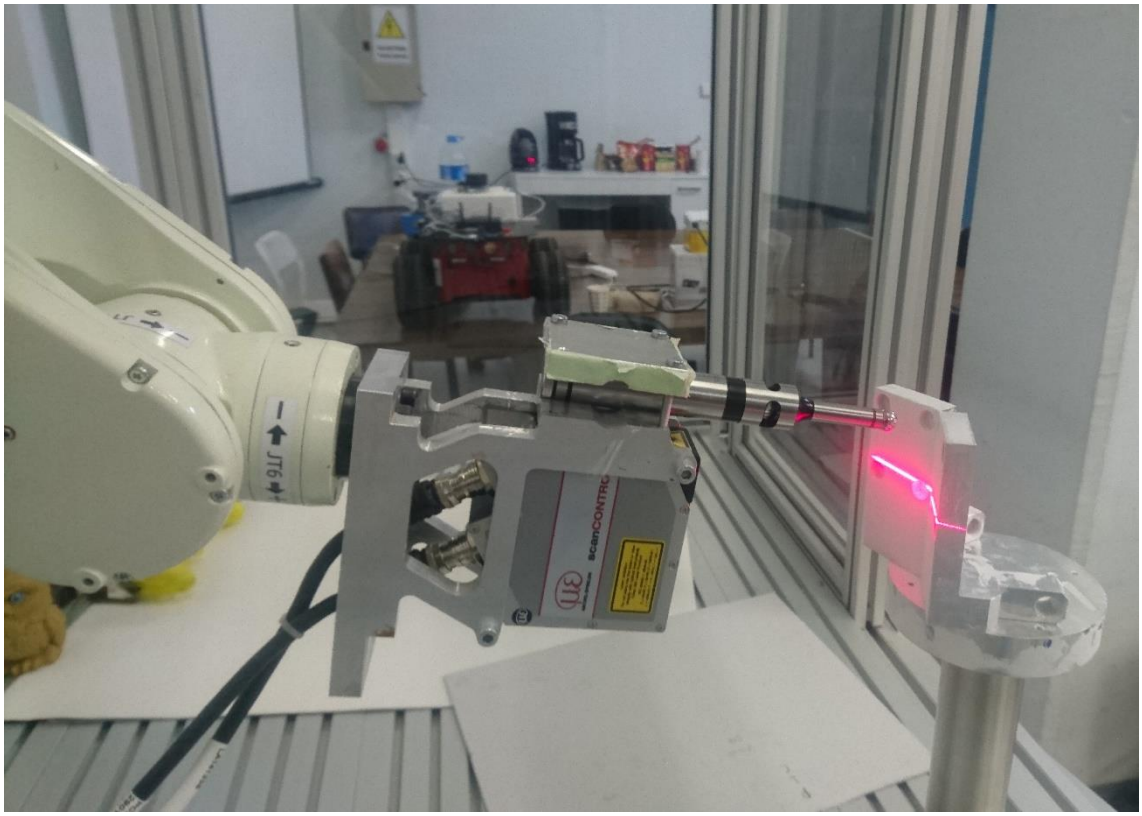


Görsel 3.25. *Kaynak Hattı Nokta Bulutuna ait Normal Vektörler*

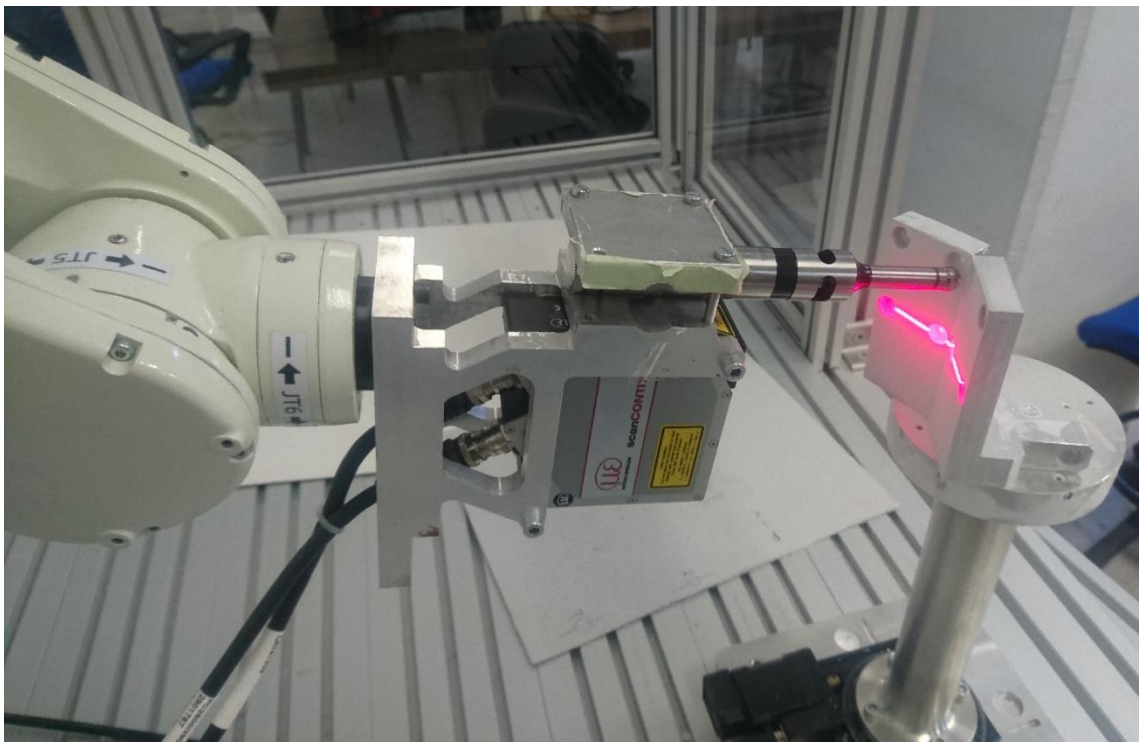


Görsel 3.26. *Kaynak Hattı Nokta Bulutuna ait Ortalanmış Normal Vektörler*

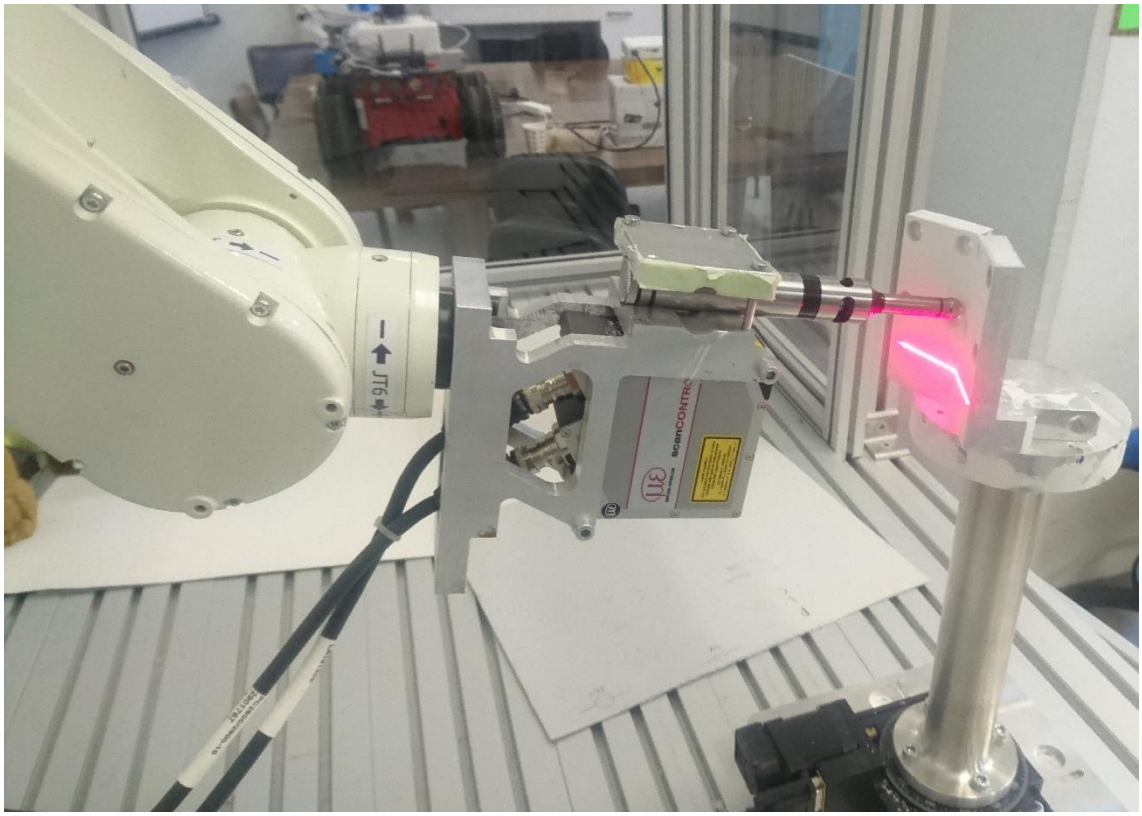
Normal vektörlerin ve hareket pozisyonlarının elde edilmesi robot hareket planının oluşturulmasına olanak sağlamaktadır. Robot hareket planının oluşturulması ile robot programlama süreci tamamlanmış olacaktır. Sürecin son adımı olarak elde edilen robot hareket planını sistem tarafından robota gönderilip kaynak işlem ucunun kaynak yapılacak hat boyunca hareket ettirilmesi sağlanır (Bkz. Görsel 3.27-3.30).



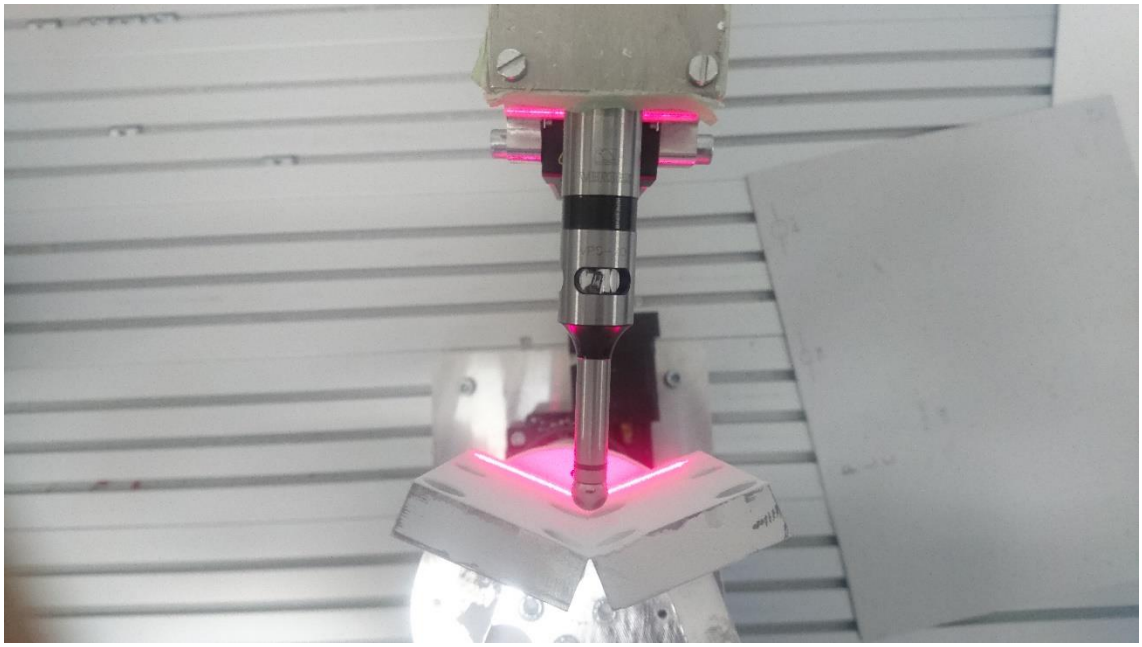
Görsel 3.27. *Robot Kaynak Ucunun Başlangıç Noktasına Yaklaşması*



Görsel 3.28. *Kaynak İşlemi Başlangıcı*



Görsel 3.29. *Kaynak İşlemi Bitişi*



Görsel 3.30. *Kaynak İşlemi Bitiş Noktası Üst Görünüm*

4. SONUÇLAR VE YORUMLAR

Gerçekleştirilen bu çalışma ile üretim tesislerin aktif olarak kullanılan en önemli donanımlardan olan endüstriyel robotların kullanımında karşılaşılan en büyük sorunlardan biri olan programlama sürecinin iyileştirilmesi hedeflenmiştir. Bu hedefin gerçekleştirilebilmesi doğrudan programlama sürecini gerçekleştirecek olan operatörün ihtiyaç duyacağı teknik bilgi ve beceri gereksiniminin en aza indirilmesi ile mümkün kılınmaktadır. Bunu sağlamak için ise, robot ve operatör arasında bir etkileşim katmanı olan ve operatörün ihtiyaç duyacağı teknik bilgileri içerisinde barındıran ve operatör adına gerekli hesaplamaları gerçekleştiren bir sistem geliştirilmiştir. Bu sistem aynı zamanda programlama ve endüstriyel işlem sırasında kontrol edilmesi gereken donanımı da kontrol altında tutması ile operatörün gözetmesi gereken pek çok durumu ortadan kaldırmaktadır. Bu nedenle, endüstriyel robot programlama sürecini karmaşık, uzun deneme yanılma yöntemlerine dayalı ve ilkel dillere bağımlı zorlu bir süreç olmaktan, basit kullanıcı bilgisayar etkileşimlerine dayanan görsel zenginliğe sahip bir sürece dönüştürülmüştür.

Ortaya çıkarılan sistemin tasarımında gözetilen çok yüzlülük ile farklı donanımlarında kolayca sisteme adapte edilebilmesine olanak sağlanmaktadır. Bu durum hali hazırda benzer donanıma sahip kullanıcıların da geliştirilen sistemi kullanabilmesine izin verilmektedir.

Kullanılan ara basamakların tamamı bu tez kapsamında yöntem bölümünde, detaylı bir şekilde, örnek görsellerle desteklenerek sunulmuştur. Bu bölümlerde kullanılan yöntemlerin pek çoğunun ayrıntılı sahte kodları yine ilgili bölümlerde sunulmaktadır. Ortaya çıkarılmış olan sistem kullanılarak uygulanan gerçek bir uygulamaya ait görseller ayrıca yorumlanarak sunulmaktadır.

Tez çalışmasının ana hedefi doğrultusunda küçük ve orta ölçekli üretim tesislerinde ürün çeşitliliğinin fazla olması sebebi ile karşılaşılan sık aralıklar ile programlama ihtiyacının ortaya çıkması dolayısı ile ortaya çıkan kaldırılamaz programlama maliyetlerinin düşürülmesi ön görülmektedir. Bu maliyetlerin düşmesi söz konusu üretim tesisleri için endüstriyel robot kullanımının önünü açma potansiyeli yüksektir. Bu durumda bu işletmeler için üretim miktarı kalitesi yükselecektir.

Ortaya çıkartılmış olan yöntemin adımlarının yürütülmesi için ihtiyaç duyulan kullanıcı müdahalelerinin mümkün olduğunca azaltılması ve kullanıcı arayüzü operatör

etkileşiminin azaltılması gelecekte izlenecek yolun belirleyici unsuru olacaktır. Bu bağlamda kullanılan yöntemlerin girdi ihtiyaçlarının azaltılması gerekmektedir. Bununla birlikte robotun çizgisel hareket dışında eğrisel hareket yapabilmesi için hareket planının oluşturulması hedeflenmektedir.

KAYNAKÇA

[1] Neto, P., Pires, J.N. and Moreira, A.P. (2010). High-level programming and control for industrial robotics: using a hand-held accelerometer-based input device for gesture and posture recognition. *Industrial Robot: An International Journal*, 37, 137-147.

[2] Pires, J.N., Godinho, T., Ferreira, P. and Loureiro, A. (2005). Industrial robotic system programmed from CAD files–an update. *Industrial Robot: An International Journal*, 32, 314-317.

[3] Pan, Z., Polden, J., Larkin, N., Duin, S.V. and Norrish, J. (2012). Recent progress on programming methods for industrial robots. *Robotics and Computer-Integrated Manufacturing*, 28, 87-94.

[4] Ang, M.H., Wei, L. and Yong, L. S. (2000). An industrial application of control of dynamic behavior of robots-a walk-through programmed welding robot. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on* (Vol. 3, pp. 2352-2357). IEEE.

[5] Sugita, S., Itaya, T. and Takeuchi, Y. (2004). Development of robot teaching support devices to automate deburring and finishing works in casting. *The International Journal of Advanced Manufacturing Technology*, 23(3-4), 183-189.

[6] Choi, M.H. and Lee, W.W. (2001). A force/moment sensor for intuitive robot teaching application. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on* (Vol. 4, pp. 4011-4016). IEEE.

[7] Yap, H. J., Taha, Z., Dawal, S. Z. M. and Chang, S. W. (2014). Virtual reality based support system for layout planning and programming of an industrial robotic work cell. *PloS one*, *9*(10), e109692.

[8] Pan, Z. and Zhang, H. (2007, November). Robotic programming for manufacturing industry. In *Proceedings of ICMEM, international conference on mechanical engineering and mechanics. Wuxi, China*.

[9] Ragaglia, M., Zanchettin, A.M., Bascetta, L. and Rocco, P. (2016). Accurate sensorless lead-through programming for lightweight robots in structured environments. *Robotics and Computer-Integrated Manufacturing*, *39*, 9-21.

[10] Schraft, R.D. and Meyer, C. (2006). The need for an intuitive teaching method for small and medium enterprises. *VDI BERICHTE*, *1956*, 95.

[11] Zhang, H., Chen, H., Xi, N., Zhang, G. and He, J. (2006, October). On-line path generation for robotic deburring of cast aluminum wheels. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on* (pp. 2400-2405). IEEE.

[12] Solvang, B., Sziebig, G. and Korondi, P. (2008). Robot programming in machining operations. In *Robot Manipulators*. InTech.

[13] Gonzalez-Galvan, E.J., Loredo-Flores, A., Laborico-Aviles, E.D., Pazos-Flores, F. and Cervantes-Sanchez, J. J. (2007, April). An algorithm for optimal closed-path generation over arbitrary surfaces using uncalibrated vision. In *Robotics and Automation, 2007 IEEE International Conference on* (pp. 2465-2470). IEEE.

[14] Hu, Z., Marshall, C., Bicker, R. and Taylor, P. (2007). Automatic surface roughing with 3D machine vision and cooperative robot control. *Robotics and Autonomous Systems*, *55*(7), 552-560.

[15] Reinhart, G., Vogl, W. and Kresse, I. (2007, June). A projection-based user interface for industrial robots. In *Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2007. VECIMS 2007. IEEE Symposium on* (pp. 67-71). IEEE.

[16] Bi, Z.M. and Lang, S.Y. (2007). A framework for CAD-and sensor-based robotic coating automation. *IEEE transactions on industrial informatics*, *3*(1), 84-91.

[17] Burdea, G. C. (1999). Invited review: the synergy between virtual reality and robotics. *IEEE Transactions on Robotics and Automation*, *15*(3), 400-410.

[18] Pettersen, T., Pretlove, J., Skourup, C., Engedal, T. and Lkstad, T. (2003, October). Augmented reality for programming industrial robots. In *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality* (p. 319). IEEE Computer Society.

[19] Chong, J.W.S., Ong, S.K., Nee, A.Y. and Youcef-Youmi, K. (2009). Robot programming using augmented reality: An interactive method for planning collision-free paths. *Robotics and Computer-Integrated Manufacturing*, *25*(3), 689-701.

[20] Fang, H.C., Ong, S.K. and Nee, A.Y.C. (2012). Interactive robot trajectory planning and simulation using Augmented Reality. *Robotics and Computer-Integrated Manufacturing*, *28*(2), 227-237.

[21] Rosario, J.M., Aviles, O.F., Kuteken, R. and Melo, L.F. (2014, June). Virtual Based Antropomorhic Gripper Application For Automation Grasping Tasks. In *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2014 International Symposium on* (pp. 521-526). IEEE..

[22] Rebelo, J., Sednaoui, T., den Exter, E.B., Krueger, T. and Schiele, A. (2014). Bilateral robot teleoperation: A wearable arm exoskeleton featuring an intuitive user interface. *IEEE Robotics & Automation Magazine*, *21*(4), 62-69..

[23] Tsarouchi, P., Makris, S. and Chryssolouris, G. (2016). Human–robot interaction review and challenges on task planning and programming. *International Journal of Computer Integrated Manufacturing*, *29*(8), 916-931.

[24] Uribe, A., Perez-Gutierrez, B. and Alves, S. (2012, October). Gesture-based teleoperation using a holonomic robot. In *Control, Automation and Systems (ICCAS), 2012 12th International Conference on* (pp. 208-213). IEEE..

[25] Pedersen, M.R., Herzog, D.L. and Krüger, V. (2014, September). Intuitive skill-level programming of industrial handling tasks on a mobile manipulator. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on* (pp. 4523-4530). IEEE.

[26] Lambrecht, J., Kleinsorge, M., Rosenstrauch, M. and Krüger, J. (2013). Spatial programming for industrial robots through task demonstration. *International Journal of Advanced Robotic Systems*, *10*(5), 254..

[27] Dániel, B., Korondi, P., Sziebig, G. and Thomessen, T. (2014). Evaluation of flexible graphical user interface for intuitive human robot interactions. *Acta Polytechnica Hungarica*, *11*(1), 135-151..

[28] Takano, Y., Yamashita, N. and Fujita, Y. (2007). Robot-Type Integrated User Interface Platform. *NEC technical journal*, *2*(2), 20-23..

[29] Mateo, C., Brunete, A., Gambao, E. and Hernando, M. (2014, September). Hammer: An Android based application for end-user industrial robot programming. In *Mechatronic and Embedded Systems and Applications (MESA), 2014 IEEE/ASME 10th International Conference on* (pp. 1-6). IEEE..

[30] Veiga, G., Malaca, P. and Cancela, R. (2013). Interactive Industrial Robot Programming for the Ceramic Industry. *International Journal of Advanced Robotic Systems*, *10*(10), 354..

[31] Balazs, D., Thomessen, T. and Korondi, P. (2013). Simplified Human-Robot Interaction: Modeling and Evaluation. *Modeling, Identification and Control*, *34*(4), 199..

[32] Norberto Pires, J., Veiga, G. and Araújo, R. (2009). Programming-by-demonstration in the coworker scenario for SMEs. *Industrial Robot: An International Journal*, *36*(1), 73-83.

[33] Müller, R., Matthias Vette, M. and Mailahn, O. (2016). Simplified path programming for industrial robots on the basis of path point projection by a laser source. *Applied Mechanics & Materials*, *840*.

[34] İğne Deliği Kamera Modeli. https://docs.opencv.org/2.4/_images/pinhole_camera_model.png (Erişim tarihi: 06.11.2017).

[35] Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, *22*(11), 1330-1334.

[36] Harris, C., & Stephens, M. (1988, August). A combined corner and edge detector. In *Alvey vision conference* (Vol. 15, No. 50, pp. 10-5244).

[37] Shi, J. (1994, June). Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on* (pp. 593-600). IEEE..

[38] Turgut, K., Dutagaci, H., Soyleyici, C., Secil, S., Ozkan, M., Parlaktuna, O. and Parlaktuna, M. (2017, April). A method for determining local coordinate frames attached to objects by a laser profile sensor in industrial robot workspaces. In *Autonomous Robot Systems and Competitions (ICARSC), 2017 IEEE International Conference on* (pp. 260-265). IEEE.

[39] Rusu, R.B. Sample Consensus Model Plane. http://docs.pointclouds.org/trunk /classpcl_1_1_sample_consensus_model_plane.html (Erişim tarihi: 06.11.2017).

[40] Rusu, R. B., Marton, Z. C., Blodow, N., Dolha, M. and Beetz, M. (2008). Towards 3D point cloud based object maps for household environments. *Robotics and Autonomous Systems*, *56*(11), 927-941.

[41] Rusu, R.B. Statistical Outlier Removal. http://docs.pointclouds.org/trunk/classpcl _1_1_statistical_outlier_removal.html (Erişim tarihi: 06.11.2017).

[42] Rusu, R. B. (2010). Semantic 3d object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, *24*(4), 345-348.

[43] Rusu, R.B. Euclidean Cluster Extraction. http://docs.pointclouds.org/trunk/ classpcl_1_1_euclidean_cluster_extraction.html (Erişim tarihi: 06.11.2017).

[44] Raoul Hoffmann, K.H., Rusu, R.B. (06.11.2017). *Sample Consensus Model Circle 3D*. http://docs.pointclouds.org/1.8.1/classpcl_1_1_sample_consensus_model_ circle3_d.html (Erişim tarihi: 06.11.2017).

[45] Rusu, R.B. *Sample Consensus Model Line*. http://docs.pointclouds.org/1.8.1/ classpcl_1_1_sample_consensus_model_line.html (Erişim tarihi: 06.11.2017).

[46] Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

[47] Uyanik, C. and Ozkan, M. (2017, October). A method for determining 3D surface points of objects by a single camera and rotary stage. In *Computer Science and Engineering (UBMK), 2017 International Conference on* (pp. 124-129). IEEE.

[48] Alcantarilla, P. F. and Solutions, T. (2011). Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Trans. Patt. Anal. Mach. Intell*, *34*(7), 1281-1298.

[49] OpenCv. AKAZE Key Point Detector. https://docs.opencv.org/3.2.0/d8/d30/ classcv_1_1AKAZE.html (Erişim tarihi: 06.11.2017).

[50] Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer vision and image understanding*, *110*(3), 346-359.

[51] OpenCv. SURF Key Point Detector. https://docs.opencv.org/3.2.0/d5/df7/ classcv_1_1xfeatures2d_1_1SURF.html (Erişim tarihi: 06.11.2017).

[52] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

[53] OpenCv. *SIFT Key Point Detector*. https://docs.opencv.org/3.2.0/d5/d3c/classcv_1_1xfeatures2d_1_1SIFT.html (Erişim tarihi: 06.11.2017).

[54] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011, November). ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE international conference on*(pp. 2564-2571). IEEE.

[55] OpenCv. ORB Key Point Detector. Available: https://docs.opencv.org/3.2.0/db/d95/classcv_1_1ORB.html (Erişim tarihi: 06.11.2017).

[56] OpenCv. Triangulate Points. https://docs.opencv.org/3.0-beta/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#triangulatepoints (Erişim tarihi: 06.11.2017)

[57] Seçil, S. (2015). Endüstriyel robot kolları için kolay programlama yöntemi geliştirme (Master's thesis, ESOGÜ, Fen Bilimleri Enstitüsü).

[58] Qt. Qt Creator. http://doc.qt.io/qtcreator/ (Erişim tarihi: 17.11.2017)

[59] Qt. (17.11.2017). *Qt QML*. http://doc.qt.io/qt-5/qtqml-index.html (Erişim tarihi: 17.11.2017)