KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

İNSANSIZ KARA ARAÇLARINDA 2D LİDAR KULLANARAK YOL SINIRLARI TESPİTİ

ÖZKAN YALÇIN

KOCAELİ 2014

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

İNSANSIZ KARA ARAÇLARINDA 2D LİDAR KULLANARAK YOL SINIRLARI TESPİTİ

ÖZKAN YALÇIN

Yrd. Doç. Dr. Ahmet SAYAR
Danışman, Kocaeli Üniv. .....

Yrd. Doç. Dr. Suhap ŞAHİN
Jüri Üyesi, Kocaeli Üniv. .....

Doç. Dr. Semih SEZER
Jüri Üyesi, YTÜ .....

Tezin Savunulduğu Tarih: 19.02.2014

ÖNSÖZ VE TEŞEKKÜR

Yüksek lisans eğitimim süresince değerli birikimlerini benimle paylaşan, tezimin her aşamasında sorunlarımı dinleyerek, çalışmalarıma yön veren ve yoğun akademik yaşamında değerli zamanını her türlü problemimi çözmeye ayıran tez danışmanım saygıdeğer hocam Yrd. Doç. Dr. Ahmet SAYAR'a teşekkürlerimi sunarım.

Bugünlere gelmemi sağlayan anneme, babama ve tez çalışmam dolayısıyla bazen ilgilenemediğim bana her konuda katlanarak çalışmalarımda manevi desteğini eksik etmeyen değerli nişanlım Büşra'ya saygı, sevgi ve sonsuz teşekkürler.

Ayrıca tez projemi destekleyerek bana tezim üzerinde çalışma olanağı sağlayan ve değerli vakitlerini tezimde karşılaştığım sorunlar için ayıran TÜBİTAK BİLGEM BTE Kara Yolu Ulaşım Bölümü yönetici ve çalışma arkadaşlarıma teşşekkürlerimi sunarım.

Şubat – 2014 Özkan YALÇIN

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER DİZİNİ VE KISALTMALAR

Kısaltmalar

1D	: 1 Dimension (1 Boyutlu)
2D	: 2 Dimension (2 Boyutlu)
3D	: 3 Dimension (3 Boyutlu)
ALVINN	: Autonomous Land Vehicle in a Neural Network (Otonom Kara Araçlarında Sinir Ağları)
ARCADE	: Automated Road Curvature and Direction Estimation (Kıvrımlı Yollarda Doğrultu Kestirimi)
AVC	: Autonomous Vehicle Competetion (Otonom Araç Yarışları)
DARPA	: Defense Advanced Research Projects Agency (İleri Savunma Projeleri)
GPS	: Global Positioning System (Küresel Konum Sistemi)
GOLD	: Generic Obstacle and Lane Detection System (Genel Engel ve Şerit Tanıma Sistemi)
IMU	: Inertial Measurement Unit (Ataletsel Ölçüm Birimi)
IPDAF	: Integrated Probabilistic Data Association Filter (Entegre Olasılıksal Veri İlişkilendirme Filtresi)
İKA	: İnsansız Kara Aracı
LIDAR	: Light Detection And Ranging (Işıma ile Uzaklık Tespiti)
LMS	: Laser Measurement System (Lazer Ölçüm Sistemi)
LOIS	: Likelihood of Image Shape (Resim Formlarının Olasılığı)
RALPH	: Rapidly Adapting Lateral Position Handler (Hızlı Adaptif Yanal Pozisyon Systemi)
ROS	: Robot Operating System (Robot İşletim Sistemi)
UGV	: Unmanned Ground Vehicle (İnsansız Kara Aracı)

İNSANSIZ KARA ARAÇLARINDA 2D LİDAR KULLANARAK YOL SINIRLARI TESPİTİ

ÖZET

Robotik alanının son yıllarda en popüler konularından bir tanesi de insansız otonom hareket eden taşıtlardır. Aynı zamanda bir çok alanda otonom araçları görmekteyiz. Özellikle savunma sanayiinde artık insansız araçlar vazgeçilmez olmuşlardır ve önemli bir çaydırıcı güç konumundadırlar. Bir çok devlet insansız kara, hava ve deniz aracı geliştirmek için büyük yatırımlar yapmaktadırlar. İnsansız araçların otonom hareketini sağlamak için çözülmesi gereken önemli problemler vardır. Bunlardan bazıları; konumlandırma, güzergah planlama, engel tanıma, engellerden kaçınma ve yol çıkarımı konularıdır. Bu çalışmada insansız kara taşıtlarının otonom hareketlerinde önemli bir problem olan yol sınırlarının tespiti için yeni bir yöntem önerilmektedir. Önerilen yöntemde 2 boyutlu bir lidar (light detection and ranging) sensör, belli bir açı kadar eğilerek yol taraması yapılmıştır. Elde edilen uzaklık bilgileri çeşitli filtrelerden geçirilerek taranan bölgenin yol olup olmadığı tespit edilmiştir. Bu çalışmada önerilen sistem simülasyon ortamında ve gerçek bir test aracı olan AKAY01 üzerinde test edilmiş ve çalışmanın son bölümünde elde edilen sonuçlar paylaşılmıştır.

Anahtar Kelimeler: IMU, İnsansız Kara Aracı, Lidar, Yol Sınırları Tespiti.

ROAD BOUNDARY DETECTION USING 2D LIDAR AT UNMANNED GROUND VEHICLE

ABSTRACT

This paper proposes a novel approach to detecting road boundaries for robusturban navigation of unmanned ground vehicles (UGVs). In this method, a 2-D Light Detection and Ranging (LIDAR) sensor mounted at a given pitch angle is used to extract the road surface. The information obtained about the road is evaluated by filters to detect whether the area scanned is a road or an obstacle. The proposed method is implemented through simulation and a real-life test vehicle called AKAY01.

Keywords: IMU, Unmanned Ground Vehicle, Lidar, Road Boundry Detection.

GİRİŞ

17. yüzyılın başından itibaren kullanılmaya başlanan araçlar ile ulaşım daha hızlı ve konforlu bir şekilde yapılabilmektedir. Yüzyıl boyunca araç konforunda ve teknolojisinde yenilikler olmuş ve insan yaşamı araçlardan bağımsız düşünülemez olmuştur. Doğal olarak bu durum araç sayısının artışına sebep olmuştur. Araçların yaygınlaşması yeni problemleri de beraberinde getirmiştir. Bu problemler kısaca; trafik sıkışıklığı, egzos salınımından kaynaklı hava kirliliği, trafik kazalarına bağlı yaralanmalar ve ölümler. Bu problemlerin sebebi araç yoğunluğu olmakla birlikte insanların kurallara uymamasının etkisi de azımsanamaz. Özellikle son 10 yılda otomotiv şirketleri tarafından sürücünün araç kullanımına yardımcı olan yeni özellikler eklenmiştir. Araçların kendi kendine park etmesi, şerit değişiminde araç var ise uyarı vermesi, öndeki araç ile olan mesafenin sabit tutulması gibi özellikler kullanılmaya başlanmıştır. Sürücünün kullanmasını gerektirmeyen, tamamen kendi kendine hareket eden araçların gelişimi paralel de hem akademik dünyada hem de otomotiv dünyasında devam etmektedir. Bu araçlar otonom araç veya sürücüsüz araç olarak isimlendirilmektedir. Konu ile ilgili değişik ülkelerde prototipler geliştirilmiştir. Geliştirilen bazı prototip araçlar U.S. deki Defense Advanced Research Projects Agency (DARPA) tarafından 2003 yılından itibaren düzenlenen Urban Challenge yarışmasına katılmışlardır. Sürücüsüz araçlara örnek olarak 2007 DARPA Urban Challenge yarışmasını kazanan Carnegie Mellon University üniversitesinin Boss isimli aracı verilebilir [1, 26, 29].

Kazaların çoğunun insan kaynaklı olduğunu düşünürsek, araçların daha akıllı olmaları kaza oranlarını düşürecek, yakıt tüketimini optimum seviyede tutarak enerji tüketimini en uygun seviyelere indirecektir. Ancak otonom araçlarda aşılması gereken bir çok problem vardır. Bu problemlerden bazıları: güzergah planlama, hassas konumlandırma, trafik ışıklarının tespiti, trafik işaretlerini algıma, engel tespiti, yol sınırlarının tespiti vb. Ancak bunlardan en önemlisi trafikte bulunan yayaların, hayvanların ve dolayısıyla sürücülerin zarar görmemesi için engelleri tanıyarak aracın hareket edebileceği yol sınırlarının tespitidir. Otonom araçlarda yol

sınırlar tespiti için kullanılan bir çok yöntem ve sensör vardır. Bunlar: kameralar ve görüntü işleme teknikleri, hassas radarlar, ultrasonik sensörler, Light Detection And Ranging (LIDAR) vb. Bu yöntemler arasında en çok tercih edilen ve bir çok akademik araştırmaya da konu olan kamera ve görüntü işleme teknikleridir [2]. Çünkü kameralar diğer sensörlerle karşılaştırıldığında tedarik edilmesi daha kolay ve ucuz cihazlardır. Kameralar yüksek çözünürlükte bilgi içerdikleri için ve eski bir tekonoloji olduğundan bir çok başarılı çalışma vardır [3]. Bunlardan bazıları: likelihood of image shape (LOIS) [4], generic obstacle and lane detection system (GOLD) [5], autonomous land vehicle in a neural network (ALVINN) [6], automated road curvature and direction estimation (ARCADE) [7], rapidly adapting lateral position handler (RALPH) [8], Görüntü işleme yaklaşımları; yağmur, sis, gölge, ışık miktarı gibi temel hava koşullarından olumsuz yönde etkilenmektedir. Yukarıda bahsedilen etkenlerden dolayı yol sınırı tespiti ile ilgili işlemleri görüntü işleme teknikleriyle yapmak oldukça zordur, ancak bu yöntemler belli bir oranda başarıya ulaşmışlardır [3]. Kameralar düşük enerjide çalışırlar, hafiftirler, maliyetleri düşüktür. Bu kadar avantajı olmasına rağmen hava koşulları gibi günlük hayatın olağan olaylarından etkilendikleri için yüksek güvenlik gerektiren otonom araçlarda yol sınırı tespiti için genelde yardımcı sistem olarak kullanılmaktadırlar [9].

Yol sınırları tespitinde kullanılan diğer bir teknoloji ise radar sistemleridir. Radarlar uzak mesafedeki nesneleri tespit edebilir; kötü hava koşullarından ve ortamın ışık durumundan çok fazla etkilenmezler [10]. Kara araçlarında 2 radar tipi kullanılır: 24-GHz dalga boyuyla çalışan kısa menzilli radarlar ve 76-77-GHz dalga boyunda çalışan uzun menzilli radarlar [11, 12]. Radarlar son teknoloji araçlarda bir çok problemi aşmak için kullanılmaktadır. Kör nokta kontrolü, araç takip mesafesinin korunması, park etme radarların kullanıldığı başlıca alanlardır. Yol sınırları tespitinde radar kullanımıyla ilgili bir çok akademik çalışma vardır [13]. Ancak radarlar mekanizmalarından dolayı tarama hızları düşüktür ve gerçek zamanlı yol sınırı tespiti için yeterli değildirler [3].

Yol sınırları tespitinde özellikle son yıllarda en çok kullanılan teknoloji lazer sensörlerdir. Lazer sensörler kameraların etkilendiği doğa olaylarından(sis, ışık geliş açısı, gölge vs) etkilemezler. Bu konu ile ilgili son yıllarda bir çok akademik çalışma

yapılmaktadır. Bu çalışmada da 2D lidar sensörler kullanılarak aracın içerisinde bulunduğu ortamda yol sınırları tespiti işlemi yapılacaktır.

1. GENEL BİLGİLER

1.1. Tez Çalışmasının Amacı ve Başlatılma Sebepleri

İnsansız kara araçları haraket halindeyken karşılarına çıkan bir engele çarptığında kendisine veya çarptığı nesneye zarar verecekse ya bu engelden kaçınmalı yada durmalıdır. Bu engeller; trafikteki araçlar, yayalar, yolda bulanan çukurlar, hız yavaşlatıcı tümsekler, yol kenarlarındaki kaldırımlar, bariyerler ve yolun üzerindeki farklı cisimler (lastik, trafik levhası vb.) olabilir. Şekil 1.1'i inceleyecek olursak problem aracın hareket edebileceği yol sınırlarını belirlemektir. Şekil 1.1'e göre dubanın sağ tarafından asfalt yolun sağ sınırına kadarki bölgeye A ve dubanın sol tarafından asfalt yolun sol sınırına kadarki alana B diyelim. Bir insansız kara aracının Şekil 1.1'deki gibi bir durum ile karşılaştığında engele çarpmadan A ve B alanlarından birini tercih ederek yoluna devam etmesi gerekmektedir. İnsansız bir kara aracının böyle bir durum karşısında yapması gereken, yol sınırını tespit ederek hareketine devam edebileceği alanları yani yol sınırlarını belirlemesidir.

Şekil 1.1. Ortasında bir duba bulunan bulunan bir otoyol

Yol sınırları tespit etmedeki problemleri sıralayacak olursak:

1. Yol insansız kara aracının geçebileceği genişlikte, hareket edebileceği düzgünlükte olmalıdır.

2. Değişken ortamlar için uygulanabilir olmalıdır. İç ve dış ortam, şehir içi yollar, asfalt yollar gibi ortamlarda çalışmalıdır.
3. Gerçek zamanlı sistemlerde gecikmelere sebep olmamalıdır.
4. Önerilen sistem uygulanabilir olmalıdır. İşlemci gücü, disk kapasitesi gibi hareketli sistemlerde sınırlı olduğu için bu gibi parametreler dikkate alınmalıdır.

1.2. Tez Çalışmasının Katkıları

Bu çalışma ile birlikte insansız kara araçlarında çözülmesi gereken önemli bir problem olan yol sınırları tespiti konusuna önemli bir katkıda bulunulacağı düşünülmektedir. Bu sebepden dolayı bu çalışma başlatılmış ve bu noktaya getirilmiştir.

Yukarıda bölümde bahsedilen problemlere karşı aşağıdaki çözümler önerilmiştir;
1. Bu çalışmada lidar sensör kullanılarak pozitif(dubalar, kaldırımlar, çimler, kasisler vb.), hatta negatif(çukurlar, hendekler, vb.) engeller başarılı bir şekilde tespit edilecektir.
2. Aracın hareket edeceği yolun sınırları geliştirilen yol sınırları tespit etme algoritması sayesinde tespit edilecektir.
3. Geliştirilen yöntem simülasyon ortamında test edilerek performans ayarları yapılmış ve eşik değerleri belirlenmiştir.
4. Önerilen yöntem Tübitak'ın otonom araç prototipi AKAY01 de test edilmiştir.

1.3. Tez Düzeni

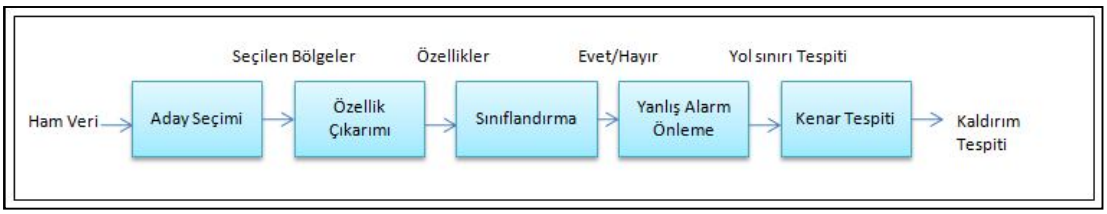
Bu tez çalışmasında Bölüm 1'de insansız kara araçlarında yol sınırları tespitiyle ilgili bazı problemlerden bahsedilmiş ve konuya giriş yapılmıştır. Bölüm 2'de yol sınırları tespiti ile ilgili geçmişte yapılmış çalışmalardan bahsedilmiştir. Bölüm 3'de Konuyla ilgili temel kavramlar hakkında bir ön bilgi verilmiştir. Bölüm 4'de yol sınırları tespitiyle ilgili oluşturulan sistemin mimarisinden bahsedilmiştir. Bölüm 5'de önerilen algoritmayı kullanmak için geliştirilen uygulamadan bahsedilmiştir. Bölüm 6'da önerilen sistemin simülasyon ve gerçek ortam testlerinin sonuçları paylaşılmıştır. Bölüm 7'de bu tez çalışması esnasında karşılaşılan zorluklar, yapılan akedemik katkılar ve bu konuyla ilgili gelecekte neler yapılabileceğiyle ilgili düşüncelerden bahsedilmiştir.

2. İLGİLİ ÇALIŞMALAR

Son yıllarda, insansız kara araçları için yol sınırları tespiti üzerine önemli oranda bilimsel çalışma yapılmıştır. Bu bölümde yol sınırları tespiti probleminin çözümü ile ilgili günümüze kadar ne tür çözümler önerildiğinden bahsedilecektir. Bahsedilen yöntemlerde kullanmış teknolojilerden, kullanılan yöntemlerin avantajları ve dezavantajlarıyla ilgili bilgiler verilecektir.

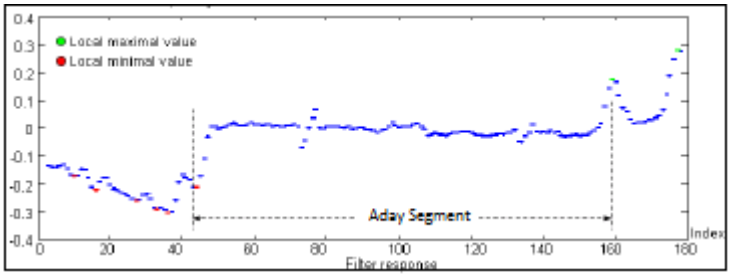
2.1. Boss Yol ve Yol Sınır Tespit Etme Algoritması

Zhang ve arkadaşlarının önerdiği yol ve yol sınırı tespiti algoritmasına göre yol kenarlarında bulunan kaldırımlardan daha alçakta ve düzgün olan yüzeydir. Bu düzgün yüzeyin lidar taraması sonucunda elde edilen datada kenar noktalarında kaldırımlardan kaynaklı ani değişimler olacaktır [14]. Yani yolun kenarlarında bulunan kaldırımlar sayesinde yolun başlangıç ve bitiş noktası tespiti yapılabilir.



Şekil 2.1. Zhang önermiş olduğu yol sınırı tespit algoritması

Önerilen algoritma 5 adımdan oluşmaktadır. Algoritmanın adımları Şekil 2.1'de gösterildiği gibidir. Birinci adım olan aday yol segment seçimi aşamasında ham olarak alınan lidar verisindeki bölgesel minimum ve maksimum değerler tespit edilerek ham veri aday yol segmentlerine ayrılır. Bölgesel minumum ve maksimumlar sabit eşik değerleriyle karşılaştırılarak tespit edilir. Şekil 2.2'de bir lidar taramasındaki aday yol segmentin nasıl belirlendiği gösterilmiştir.



Şekil 2.2. Aday segment seçimi

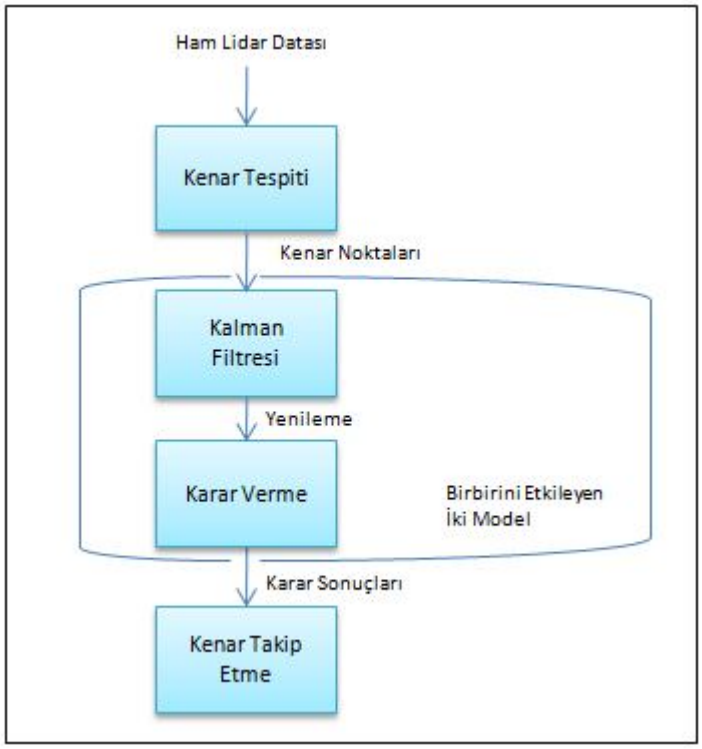
Önerilen algoritmanın ikinci aşamasında aday yol segmentlerinin özellik çıkarımı yapılır. Özellik olarak ilk paragrafta bahsettiğimiz yol yüzeyinin düzgün değiştiği kabul edildiği için yüzeyin yükseklik varyansı bir özellik olarak kabul edilir. Bu aşamada aday noktanın yükseklik varyansı çıkarılır.

Algoritmanın sınıflandırma aşamasında ise elde edilen yükselik varyansı değeri sabit bir eşik değeriyle karşılaştırılarak yol veya engel segmenti olarak sınıflandırılır. Yanlış alarmı önleme olarak isimlendirilen dördüncü aşamada ise elde edilen yol segmentlerinin genişliği önceden belirlenen genişlik eşik değeri ile karşılaştırılarak yanlış seçilimlerin önüne geçilmeye çalışılmıştır. Önerilen yöntemin son aşamasında ise kaldırım kenarı tespiti işlemi yapılmaktadır. Kaldırım kenarı tespiti ise elde edilen yol segmentlerinin maksimum ve minumum noktları yani iki zıt uçları olarak belirlenmiştir.

Önerilen sistem şehir içindeki yollar için ideal bir yöntemdir. Sistem 2007 yılında yapılan DARPA Urban Challange yarışmasında birinci olmuştur. Ancak algoritmanın yapısı düşünüldüğünde yolun her iki tarafında da kaldırım olması gerekmektedir. Şehirler arası veya toprak yollar düşünüldüğünde önerilen sistem yetersiz kalmaktadır.

2.2. Çoklu Kalman Filtresi Kullarak Yol Sınırları Tespiti

Diğer bir yol sınırları belirme yöntemi ise Kang önermiş olduğu çoklu kalman filtresi tabanlı bir sistemdir. Önerilen sistemde 2 boyutlu lazer sensör kullanılmıştır [15]. Lazer sönsöre belli bir pitch açısı verilmiştir. Bu yöntemde yol sınırlarını belirlemek için lidar ışınlarının geometrik dağılımı ve birbirleri arasındaki uzaklık değişimine bakılmıştır. Elde edilen tahmini yol sınırları arka arkaya iki kere kalman filtresinden geçirilerek yol sınırı hatasız bir şekilde tespit edilmeye çalışılmıştır.



Şekil 2.3. Kullanılan yöntemin akış şeması

Bu çalışmada ilk aşamada yol sınırları belirlenirken Hough doğru transformasyonu kullanılmıştır. Algoritmanın ikinci aşamasında arka arkaya gelen tarama noktaları Denklem (2.1) ve Denklem (2.2) kullanılarak tanjant ve uzaklık testinden geçirilerek muhtemel bir köşe noktası olup olmadığına karar verilmektedir. Uzaklık ve tanjant eşik değerleri araç tipi ve yol çeşidine göre önceden belirlenen sabit değerlerdir. Aşağıdaki eşitliklerde tanjant ve uzaklık eşik değerleri sırasıyla t_{th} ve p_{th} dır. Eğer eşitlikliklerden çıkan değerler bu eşik değerinin üzerinde ise bu nokta muhtemel yol sınırıdır.

$$\sqrt{\left(x_i - x_{i+1}\right)^2 + \left(y_i - y_{i+1}\right)^2} \leq p_{th} \tag{2.1}$$

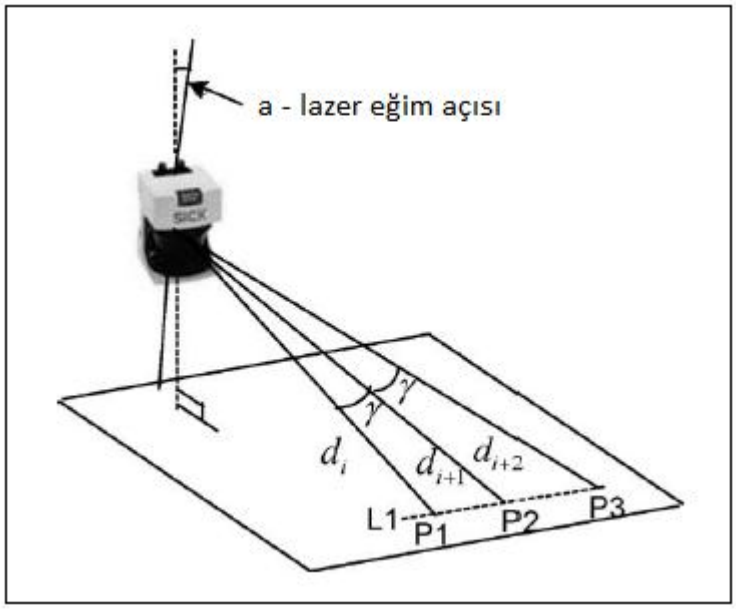
$$\left| \tan^{-1}\left(x_i - x_{i+1} / y_i - y_{i+1}\right) \right| \leq t_{th} \tag{2.2}$$

Algoritmanın son aşamasında ise tahmini olarak belirlenen yol sınırı noktaları iki kere kalman filtresi uygulanarak hatalı noktalar filtrelenmiş ve gerçek yol sınırı noktaları tespiti yapılmıştır.

İnsansız kara araçları için yol sınırları tespiti gerçek zamanlı çalışan sistemlerdir. Bu sebeple kurulan algoritmaların performansı çok önemlidir. Önerilen yöntem başarılı sonuçlar üretmektedir. Ancak kalman filtresi gibi ağır matematiksel işlemler içeren bir yöntemi iki kere üst üste uygulamak performans kaybına sebep olabilir.

2.3. Genişletilmiş Kalman Filtresi Kullarak Yol Sınırları Tespiti

Wijesoma genişletilmiş kalman filtresi kullanan farklı bir yol sınırı tespit etme algoritması önermiştir [3]. Bu yöntemde bir lidar taramasında arka arkaya gelen ışınların ölçümlerini kalman filtresi durum modeline uyarlamıştır.



Şekil 2.4. Bir lidar taraması

Kullanılan lidara a derece pitch açısıyla eğilmiştir. Şekil 2.4'de gösterilen bir lidar taramasında temel trigonometrik fonksiyonları kullanarak Denklem (2.3) elde edilmiştir.

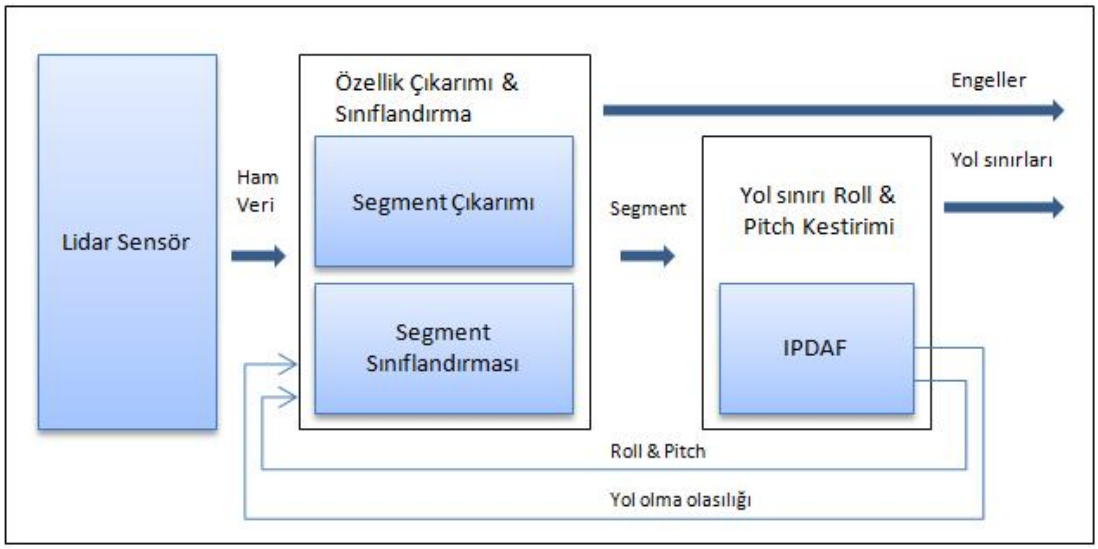
$$d_{i+2} = d_i d_{i+1} \,/\, (2d_i \cos\gamma - d_{i+1}) \qquad (2.3)$$

Elde edilen bu denklem $x_1(k+1)= d_{i+2}$, $x_2(k+1)= d_{i+1}$. $x_1(k)$ ve $x_2(k)$ şeklinde yazılarak zaman durum formuna dönüştürülmüştür. Yazar sistemi zaman durum formuna dönüştürdükten sonra tespit edilen noktalara genişletilmiş kalman filtresi uygulanarak bir yol sınırı noktası olup olmadığına karar verilmiştir.

Önerilen bu sistemde yazar tüm yollarda kaldırım bulunduğunu varsayarak yapmıştır. Ancak gerçek hayatta tüm yollarda kaldırım olması mümkün gözükmemektedir.

2.4. İyileştirilmiş Yol Sınırı Tespit Algoritması

Han ve arkadaşlarının önermiş olduğu iyileştirilmiş yol sınırları tespit algoritması yine lidar tabanlı bir algoritmadır [9]. Önerilen yöntem Şekil 2.5'deki gibi temelde 3 bölüme ayrılmıştır. İlk bölümde ham lidar verileri kutupsal kordinat sistemine dönüştürülmektedir. İkinci adımda kutupsal koordinat düzlemindeki lidar datası aday yol segmentlerine ayrılmaktadır. Bu aşamada aynı zamanda yol segmentlerinin özellik çıkarımı ve sınıflandırmanın birinci aşaması gerçekleştirimektedir. Bu aşamadan geçen aday yol segmentleri Integrated Probabilistic Data Association Filter (IPDAF) isimli kestirim tabanlı algoritmadan geçirilmektedir. Bu algoritmanın sonucunda kestirilen özelliklere göre 2. adımın segment sınıflandırma algoritması tekrar çalıştırılarak daha kesin sonuçlar üretilmeye çalışılmaktadır.



Şekil 2.5. Sistem mimarisi

Geliştirilen iyileştirilmiş yol sınırı tespit algoritması Kore'de düzenlenen Autonomous Vehicle Competetion (AVC) otonom araba yarışlarının 2010 ayağında A1 isimli aracıyla birinci olmuştur. Ancak algoritmanın IPDAF kısmında çok fazla matematiksel işlem bulunuduğu için iyileştirilmeye ihityaç vardır.

2.5. Hızlı Özellik Çıkarımıyla Olasılıksal Yol Sınırı Tespiti

Lidar kullanarak yol sınırları tespiti yapan başka bir yöntem ise Peterson önerdiği DARPA Urban Challange otonom araba yarışlarında kullandıkları araç için geliştirilen hızlı özellik çıkarımı yapan olasılıksak yol sınırı tepiti algoritmasıdır [16]. Önerilen sistemde 2D ve 3D lidarlar kullanılmıştır. Sistem 3 adımdan oluşmaktadır. Bu adımlar;

1.Ön işleme

2.Wavelet tabanlı özellik çıkarımı

3.Son işleme

Ön işleme adımında ikinci adım için ham lidar datası ayıklanır. Ayıklama işleminde komşu noktalar arasında kümeleme yöntemleri kullanılarak lidar ışınlarının yoğun olarak ulaştığı alanlar tespit edilir. Bu aşamada yol çıkarımı için gereksiz olan yol kenarındaki ağaçlar, trafik dubları, telefon direkleri gibi yol sınırıyla alakasız outlier data ayıklanır. İkinci adım olan ayrık wavelet transformu kullanılarak eldeki temizlenmiş verilerden özellik çıkarımı yapılır [17]. Algoritmanın son aşamasında ilk aşamada gözden kaçırılarak elenen muhtemel alanlar tespit edilmeye çalışılmaktadır. İlk aşamada seyrek datadan dolayı elenmiş ancak ikinci aşamada her iki tarafıda yol sınırı olarak tespit edilmiş alanlar sezgisel olarak yol sınırıdır şeklinde etiketlenir.

2.6. Hough Transformasyonu ve Dinamik Programlama Tabanlı Yöntem

Lin'in önerdiği rasgele hough transformasyonu ve dinamik programla tabanlı yol sınırı tespiti algoritmasında görüntü işleme teknikleri ve kamera kullanılmıştır [18]. Bu yöntemde yol sınırlarını belirlemek için yoldan çekilen resimlerde yoğunluk değişim bölgelerine odaklanılmıştır. Yol sınırlarında bulunan yol çizgileri dikkate alınmıştır. Araç hareket halindeyken sürekli hareket ettiği yolun resmini çekmektedir. Aracın yolun sınırları içerisinde olduğu varsayılmaktadır. Çekilen bu resimlerdeki renk yoğunlunun aniden değiştiği noktalar aday yol sınırı noktalarıdır.

Şekil 2.6. Çekilen bir yol resmi ve filtreden geçirilmiş hali

Geliştirilen yöntem ışık yoğunluğu hava şartları uygun olduğu sürece başarılı bir yöntemdir. Ancak görüntü işleme teknikleri çok fazla işlem yükü getirmektedir. Ayrıca bu yöntemde aracın her zaman yolun orta kısmında olduğu ve yol çizgilerinin düzgün olması kısıtı vardır. Bu kısıtlar düşünüldüğünde bu yöntemin emniyet kritik bir sistem olan otonom araçlarda kullanılması oldukça zor gözükmektedir. Ancak yardımcı sistem olarak kullanılabilir.

2.7. Local Binary Pattern Tabanlı Yol Sınırı Tespiti

Başarılı olmuş bir başka yol sınırı tespit etme yöntemi Sibert'in önerdiği local binary pattern tabanlı algoritmadır [19]. Bu yöntemde de kamera kullanılmılmıştır. Kameraların farklı hava koşullarından etkilenmesinden dolayı geliştirilen yöntem farklı hava koşullarında test edilmiştir. Bu yöntem doku tabanlı bir filtredir. Geliştirilen yöntem farklı hava koşullarında test edilmiştir. Yol sınırı tespitinde başarılı bir algoritma olmasına rağmen çok yavaş çalıştığı için gerçek zamanlı uygulamalarda kullanılması mümkün değildir.

2.8. Yol Sınır Tespitinde Kullanılan Diğer Yöntemler

Yukarıda bahsedilen yöntemler dışında yol sınırı ve engel tanımada farklı methodlarda kullanılmaktadır. Blanc önerdiği yöntemde infrared ve radar sensörler birlikte kullanılmaktadır [20]. Bu tip yöntemlerin yol sınırı tespit etmedeki en önemli zorlukları farklı sensör verilerinin bileştirilmesi işlemidir. Radarlar uzun menzilli mesafa ölçen sistemler olmalarına rağmen çözünürlükleri düşük olduğu için yol

çıkarımı yöntemlerinde yetersiz kalmaktadır. Radar sistemleri genellikle araçlarda takip mesafesinin korunması, cruise kontrol gibi alanlarda kullanılmaktadır.

3. TEMEL ANLATIMLAR

3.1. İnsansız Kara Araçları ve Sensörler

Otonom robotlar son yıllarda insan hayatında önemli bir yer edinmeye başladılar. Otonom robotlar kara, hava, denizaltı ve uzay ortamlarında insanların hizmetine sunulmuş ve son yıllarda askeri ve araştırma amaçlı çalışmalarda kullanılmışlardır. Denizaltında kullanılan otonom robotlar denizaltında sismik hareketleri kontrol etmek ve denizaltı yaşamını araştırmak; otonom araçları deprem, heyelan gibi karasal hareketleri veya kasırga, tsunami gibi afetleri havadan tespit edip içerisinden bilgi toplamak; otonom kara araçları da askeri amaçlı bomba ve mayın imha etmek, kütüphaneler ve hastanelerde insanlara kılavuzluk yapmak ve bilgi vermek, sanayide ise malzeme taşımak ve üretime katkıda bulunmak gibi çok geniş yelpaze de kullanılmaktadır. İnsansız kara araçları deprem, yangın ve kanalizasyon tıkanıklığı gibi durumlarda insanların ulaşamayacağı veya insanlar için tehlike arz eden yerlerde de kullanılmaktadır.

İki tür insansız kara aracı vardır.

3.1.1. Uzaktan kontrollü insansız kara aracı

Uzaktan kontrollü insansız kara araçları bir operatör tarafından kontrol edilirler. Çeşitli haberleşme altyapısı ile kontrol edilebilirler. En çok tercih edilen kontrol tipi uzaktan kumanda ile kablosuz kontroldür. Bu tip kontrol günümüz oyuncak arabalarda çok sık kullanılmaktadır. Askeri amaçla kullanılan insansız kara araçları en çok bu tiptedir.

3.1.2. Otonom insansız kara aracı

Son yıllarda ortaya çıkan otonom insansız kara araçları, aracın kendi kendisini kontrol ettiği insandan kontrolünden bağımsız çalışan insansız kara araçlarıdır. Bu tip araçlar avantajlı olmakla beraber uygulaması oldukça pahalı ve zordur. Kendisine verilen belirli bir görevi (yükü bir yerden bir yere taşıma, görme engelli bir hastayı

evden işe götürme gibi) yolunu etraftaki engellerden sakınarak belirleyen ve etraftaki yapıları, duvarları, araçları tespit ederek uygulamaya çalışmaktadır. Bu tip araçların kullanılmasındaki en önemli sebep insan kontrolünün olmadığı bölgeler veya zamanlarda robotun verilen işlevi yerine getirebilmesidir. Otonom insansız kara araçlarının verilen görevi başarıyla tamamlaması için çeşitli problemleri aşması gerekmektedir. Bu problemlerin başında gideceği güzergahtaki engellerin tespit edilmesi ve takip edebileceği yolun sınırları belirlemesi problemidir.

3.2. Sensörler

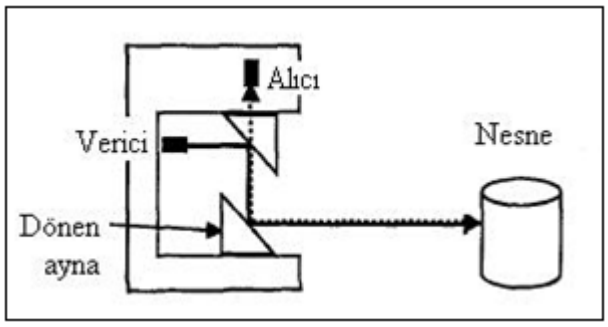
İnsansız kara araçlarında aracın etrafındaki engellerin algılanması haritalanması ve bu engellerden sakınarak hareket etmesi için çeşitli sensörlere ihtiyacı vardır. Sensörleri çalışma şekillerine göre aktif ve pasif sensörler olarak iki gruba ayırabiliriz.

3.2.1. Aktif sensörler

Aktif sensörler etraflarındaki cisimlere gönderdikleri radyo, ışın, ses gibi dalgaları gönderip cisimlerden geri yansımalarına göre uzaklık, renk, şekil gibi özellikleri hakkında bilgi almamızı sağlayan sensörlerdir. En sık kullanılan sensörler ultrasonik, lazer ve radar sensörlerdir.

3.2.1.1. Lidar sensörler

Light Detection And Ranging (LIDAR) sensörler lazer tabanlı bir teknolojidir. Lidar sensörlerin çalışma presipleri radar ve ultrasonic sensörlere benzer, ancak mesafe ölçmek için lazer kullanılır. Lazer kaynağından çıkan ışın bir nesneye çarptıktan sonra geri yansır, lazer kaynağına döner ve bu gidiş geliş süresinin ışık hızıyla çarpımı ikiye bölünür, sonuç olarak lazer ışınının yansıdığı noktanın lazer kaynağına ne kadar mesafede olduğu hesaplanmış olur. Lidar sensörler mesafe ölçmek için lazer teknolojisi kullandıkları için çözünürlükleri yüksektir. Lidar sensörler sis, yağmur, gölge gibi görüntü işlemenin yetersiz kaldığı durumlarda yüksek doğruluk oranlarına sahiptir.



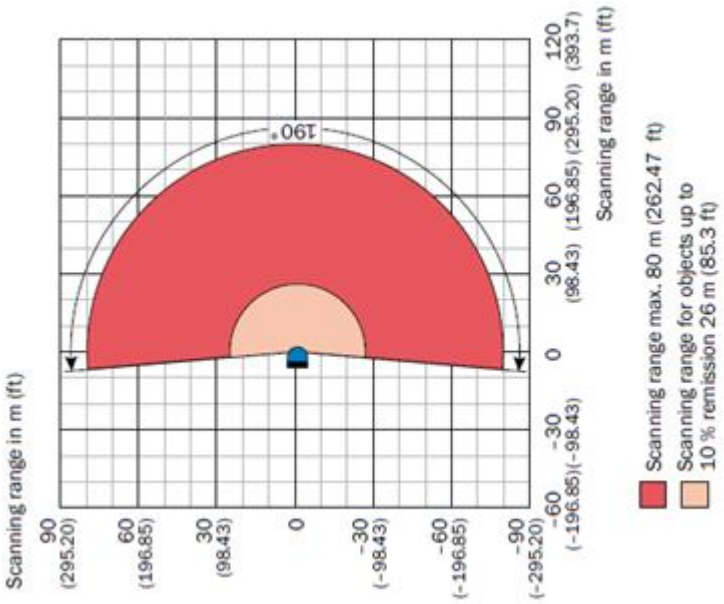
Şekil 3.1. Lidar sensörlerin iç yapısı

LIDAR sensörler endüstride çok farklı amaçlar için kullanılmaktadır. LIDAR sensörler, havadaki partiküllerin yoğunluğunu (yağmur, duman, sis, azot oksijen oranı, rüzgar hızı vb.) ölçmede, sanat eserlerinin üç boyutlu modelini çıkarmada, üç boyutlu topografik haritaların çıkarılmasında, fay hatları gibi sismik verilere erişmede, mars yüzeyi hakkında bilgi edinmede, okyanus taban analizi ve otonom araçlarda engel tanıma, yol kenarı sınırları çıkarma gibi alanlarda kullanılamaktadır. Bu çalışmada LIDAR sensörlerin otonom araçlarda kullanımlarıyla ilgili araştırmalara değinilecektir.

LIDAR sensörler engel tanımada kullanılan, engel hakkında detaylı bilgi üreten bir lazer tabablı sensördür. Lidar sensörlerin çözünürlükleri yüksektir. Örneğin Laser Mesurement System (LMS) 511 LIDAR modelinin çözünürlüğü 0.167 dereceler civarındadır. Aynı modelin 80 metrede hata oranı 30-40 mm civarındadır. LIDAR sensörleri kötü hava koşullarında, yağmur, sis vb. koşullarda dahi yüksek doğrulukta çalışır. Genel lazer sensörlerin dezavantajlarından biri olan siyah rengin ışığı emerek geri yansıtmaması LIDAR içinde geçerlidir, ancak bazı LIDAR sensörleri bu problemi belli oranda aşmıştır. Örneğin LMS 511 modelinin normalde yansıma menzili 80m iken siyah renkli bir engelden yansıma mesafesi maksimum 26 m' dir. Şekil 3.2'de 190 derece tarama açısı olan bir LIDAR sensörün normal ve siyah renktedeki menzili görülmektedir.

Tek boyutlu (1D), iki boyutlu (2D) ve üç boyutlu (3D) olmak üzere üç çeşit lidar sensör vardır. Tek boyutlu LIDAR sensörler lazer metre gibi çalışır; yani bir noktanın LIDAR sensöre göre uzaklık bilgisini verir. 2D LIDAR sensörler tarama açılarına göre bir düzlem taraması yapar ve lazer ışınlarının geri yansıdığı noktanın x ve y kordinatı hesaplanabilir. 3D LIDAR sensörler basitce birden fazla 2D LIDAR

sensörün üst üste konularak tarama yapılmasıyla taranan noktaların z eksenindeki yerden yükseklik değerlerinin elde edilmesi olarak düşünülebilir.



Şekil 3.2. 2D bir LIDAR sensörün tarama şekli

LIDAR sensörler 100Hz hızında veri üretebilen yüksek hızlı sensörlerdir. Bir otonom araçta ortamın risk haritasının çıkarılmasında kullanılacak 3-4 adet LIDAR sensörün saniyede ürettiği veri sayısı milyon nokta seviyelerine çıkabilmektedir. Bu yoğunluktaki bir verinin gerçek zamanlı olarak işlenip aracın karar destek sistemi için bilgiye dönüştürülmesi gerekmektedir. Verinin işlenmesindeki anlık gecikmeler veya veri işlemedeki hatalar aracın hareketi ile ilgili kritik sorunlara yol açabilir.

3.2.1.2. Ultrasonik sensörler

Robotik alanında son yıllarda kullanılan popüler sensörlerden bir taneside ultrasonik sensörlerdir [28]. Ultrasonik sensörler ses dalgasının bir nesneye gönderilip geri gelme süresine bağlı olarak mesafeyi algılayan sensörlerdir. Bu sensörlerin çalışma şekli şu şekildedir: verici tarafından bir ses dalgası gönderilir, bu ses dalgasının menzilinde bir engel varsa engele çarpar ve geri yansır, ses dalgasının gönderimi ile alımı arasında geçen bu süre ses hızıyla çarpılır ve engelin ne kadar uzaklıkta olduğu tespit edilir. Bu yöntemde ses yayılırken huni gibi genişler, sonra geri daralır. Bu sebeple bir dalganın yayılma genişliğinden daha küçük değişimler algınamaz. Tespit edilen nesnenin şekli ve boyutu hakkında detaylı bir bilgi elde edemeyiz. Diğer

taraftan sesi emerek geri yansıtmayan nesneler vardır. Ultrasonic sensörler ile bu özellikteki engeller tespit edilemeyebilmektedir. Bu sebeplerden dolayı ultrasonic sensörler yol kenarı tespiti gibi karmaşık ve kritik bir işlem için yetersiz ve güvensizdir. Günümüzde ultrasonic sensörler, sürücünün park etmesine yardımcı olan bir uyarıcı sistem olarak kullanılmaktadır. Denklem (3.1) ve Denklem (3.2)'de ultrasonic sensörle mesafe ölçümü formülleri verilmiştir.

TOF: Ses sinyalinin gidiş geliş süresi
T: Ortam sıcaklığı
d: Sensörün nesneye olan uzaklığı (m)
c: havadaki ses dalgasının hızı

$$c = 331.4\sqrt{\frac{T+273}{273}}\ \text{m/s} \tag{3.1}$$

$$d = c \times TOF / 2 \tag{3.2}$$

3.2.2. Pasif sensörler

Pasif sensörler nesnelerin yaydığı termal enerjiyi ve elektromanyetik dalgaboyunu algılayan sensörlerdir. Kameralar ve pyroelektik sensörleri birer pasif sensörlerdir. İnsansız kara araçlarında kameralar çok sık kullanılmaktadırlar. Genellikle otonom araçlarda kameralar ve lazer sensörler birlikte kullanılmaktadır [29, 30]. Belirli bir referans eksenine göre aracın konumunu ölçen sensörlere proprioseptif sensörler denir. GPS (Global Positioning System), Şaft enkoderleri, eğim ölçer, pusula, INS (Inertial Navigation System) gibi sensörler bu tip sensörlerdir.

Sensör seçimi yapılacak uygulamaya ve uygulamanın gerçekleşeceği çevreye göre belirlenir. Her sensörün birbirine göre avantaj ve dezavantajları bulunmaktadır. Sonar sensörleri ucuz olmakla beraber hassas alanlar için çok uygun değildirler. Hassas ve kapalı alanlarda lazer en uygun çözümdür. Pusula gibi sensörler manyetik etkileşimlerden dolayı çevrede çok fazla manyetik alan yayan cihazların bulunduğu bölgelerde iyi sonuçlar vermezler.

3.3. Simülasyon Programları

Bir işi yapmak pahalı veya tehlikeli ise geliştirilen algoritmaları test edip eksik yönlerini gidermek için simülasyon ortamı kullanılır. Simülasyon ortamının avantajı kritik bir sensörün çalışması test ediliyorsa kullanılan diğer sensörler idealleştirilerek kritik bileşenin davranışları gözlemlenebilir. Simülayonun diğer bir avantajı ise algoritmalardaki parametreler hızlıca değiştirilerek aynı senaryo tekrar oynatılabilir. Gerçek zamanlı bir problemin çözülmesi için ilk adım simülasyon ortamıdır. Robotik alanında kullanılan 2 ve 3 boyutlu bir çok simülasyon programı bulunmaktadır. En çok tercih edilen simülasyon uygulamaları Gazebo ve V-rep dir.

3.3.1. Gazebo simülasyon programı

Robotik alanda en sık kullanılan simülasyon programıdır. Robotik İşletim Sistemi (ROS) ile doğrudan bağlantı seçeneği sunmaktadır. Gazebo 3 boyutlu bir simülasyon programıdır. Gazebo simülasyon programında endüstride kullanılan bir çok sensör ve 3 boyutlu modeller bulunmaktadır. Gazebo simülasyon ortamında geliştirilen bir sistem eğer ROS üzerinde çalışıyorsa, geliştirilen algoritma kodlarında herhangi bir değişiklik yapmadan doğrudan gerçek robottaki donanımlarla uyumlu bir şekilde çalışmaktadır. Bu özelliğinden dolayı gazebo simülasyon programı robotik alanda çok fazla tercih edilmektedir. Gazebo simülasyon ortamında gerçek ortam birebir modellenebilir. Modellenen simülasyon ortamında robotun davranışları ve sonuçlarıda görülebilir [21].

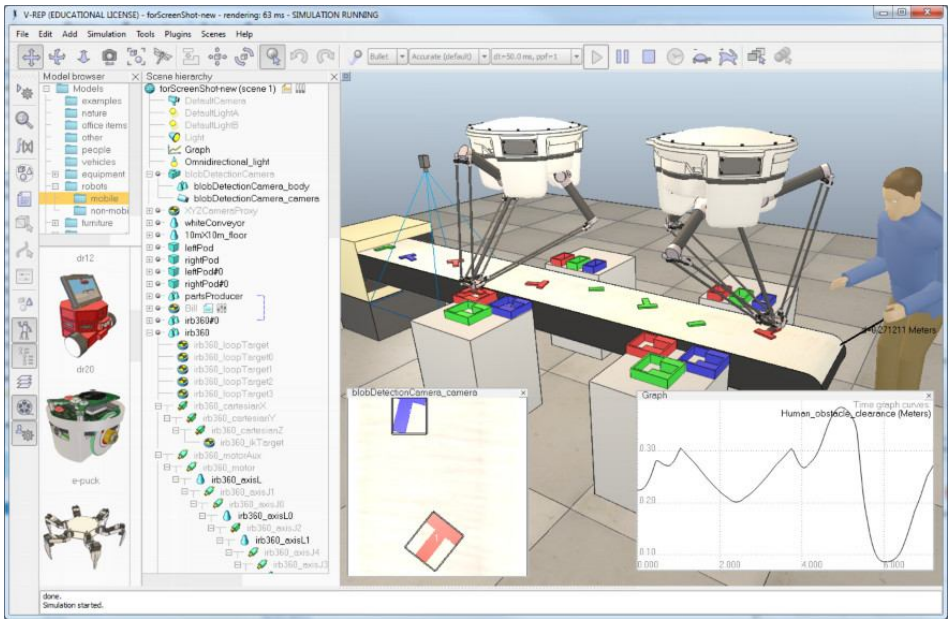
Şekil 3.3. Gazebo simülasyon ortamında modellenmiş bir ofis ve robot

Özellikleri:

1. Hızlı robot prototipi çıkarabilir.
2. Gerçek hareketler için içerisinde ODE fizik motoru bulunmaktadır.
3. Gerçek dünyada oluşabilecek sonuçlar gözlemlenebilir.(Gölge, robotun devrilmesi, bakış açısı vb.)
4. Dışarıdan yazılan kodlarla robotların davranışları yönetilebilir. Desteklediği programlama dilleri C, C++, Perl, Python, Java, URBI, MATLAB dir.

3.3.2. V-rep simülasyon ortamı

Son yıllarda popüler olan diğer bir 3 boyutlu simülasyon programıda coppellia robotics firmasının çıkarmış olduğu v-rep simülasyon uygulamasıdır. Vrep simülasyon programının kullanımı kolaydır. Arayüzleri diğer simülasyon uygulamalarına göre daha gelişmiştir. Bir çok sensör ROS eklentisi sayesinde ROS ile doğrudan haberleşebilmektedir [22].



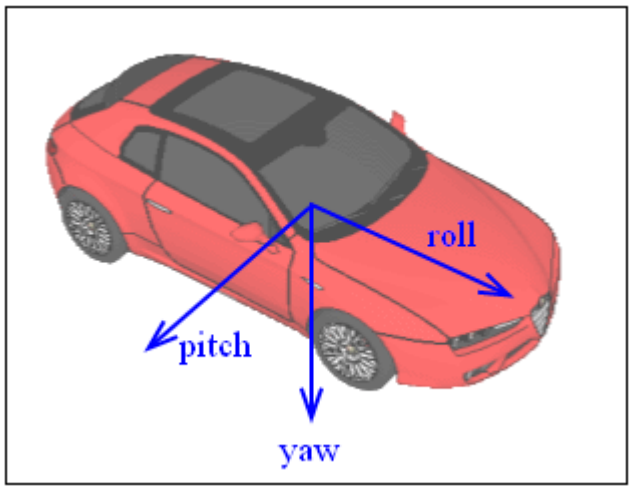
Şekil 3.4. V-rep simülasyon programında geliştirilmiş bir fabrika modeli

Özellikleri:

1. Farklı programlama dilleriyle simülasyon modelleri yönetilebilmektedir.
2. Simülasyon ortamındaki farklı bileşenler farklı bilgisayarlarda yazılmış uygulamalar tarafından yönetilebilmektedir.
3. Bullet ve ODE fizik motorlarına sahiptir. Geliştirilen simülasyon tek hareketle iki farklı simülasyon motorunda çalıştırılabilir.
4. Ters ve düz kinematik modellerde kullanılabilir.

3.4. Euler Açıları

Bir O noktası etrafında serbestçe dönebilen bir katı cismin konumunun belirtilmesinde kullanılan açılara euler açıları (yaw, pitch roll) denir. Katı cismin uzaydaki duruşunu x, y, z karteziyen koordinat düzleminde bir konum ve euler açılarıyla tamamamen tanımlayabiliriz.



Şekil 3.5. Euler açıları roll, pitch yaw

Robotik alanında aracın gerçek ortamda ve simülasyon ortamında konumunu belirlenirken veya araç üzerinde bir sensörün konumu belirlenirken kartezyen kordinatların yanında euler açıları kullanılır [23].

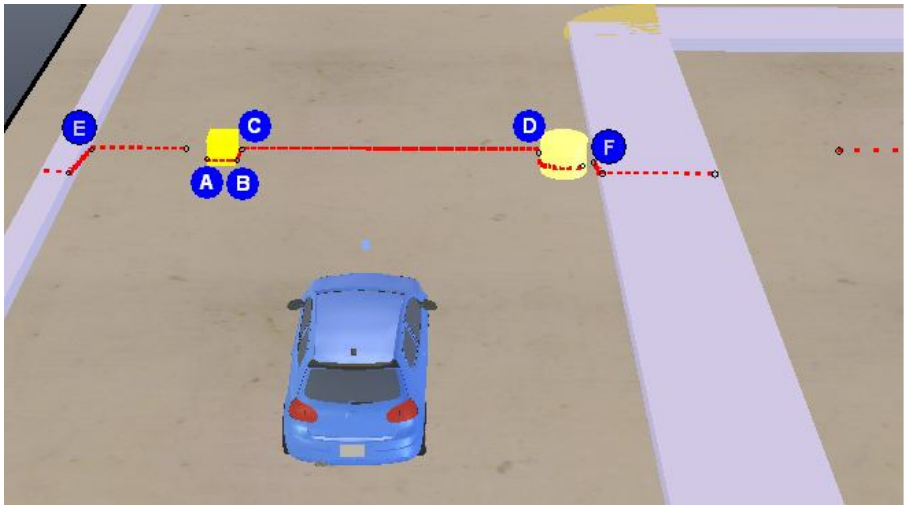
- Yaw açısı katı bir cismin kafa açısıdır. Cismin merkezine göre yaw ekseninin sağa sola dönüşünde oluşan açıdır.
- Pitch açısı Şekil 3.5'deki pitch ekseninin aracın merkezine göre sağa sola dönmesiyle elde edilir.
- Roll açısı ise Şekil 3.5'deki roll ekseninin aracın merkezine göre sağa sola dönme hareketiyle oluşan açıdır.

4. YOL SINIRLARI TESPİTİ İÇİN ÖNERİLEN SİSTEM

Bu bölümde insansız kara araçları için önerilen lidar sensör kullanılarak yol sınırları tespit etme algoritmasından bahsedilecektir. Bölüm 4.1'de yol sınırı ve engel tanımı Bölüm 4.2'de lidar sensörün araç üzerinde konumlandırılması, Bölüm 4.3'de önerilen sistemin algoritma akış diyagramı Bölüm 4.4'de kırılım noktaları tespiti algoritması Bölüm 4.5'de aday yol segmentlerinin çıkarılması ve sınıflandırılması ve Bölüm 4.6'da ise elde edilen yol segmentlerinin haritaya aktarılmasından bahsedilecektir.

4.1. Yol Sınırı

Bu çalışmada trafik akışının gerçekleştiği, yaya geçitlerinin, hız düşürücü engellerin olduğu yolun fiziksel olarak başlayıp bittiği sınırlara fiziksel yol sınırları; yol üzerindeki engeller de hesaba katılarak araçların geçebileceği genişlikteki düzlemin sınırlarına ise mantıksal yol sınırları olarak isimlendirilecektir. Şekil 4.1'de E-F noktaları fiziksel yol sınırlarını temsil ederken, C-D noktaları ise mantıksal yol sınırlarını temsil etmektedir. E-A noktaları mantıksal yol sınırları olarak kabul edilmektedir. Çünkü, aracın eni E-A arasındaki mesafeden daha geniştir, dolayısıyla araç bu aralıktan geçemeyecektir.



Şekil 4.1. Mantıksal ve fiziksel yol sınırı

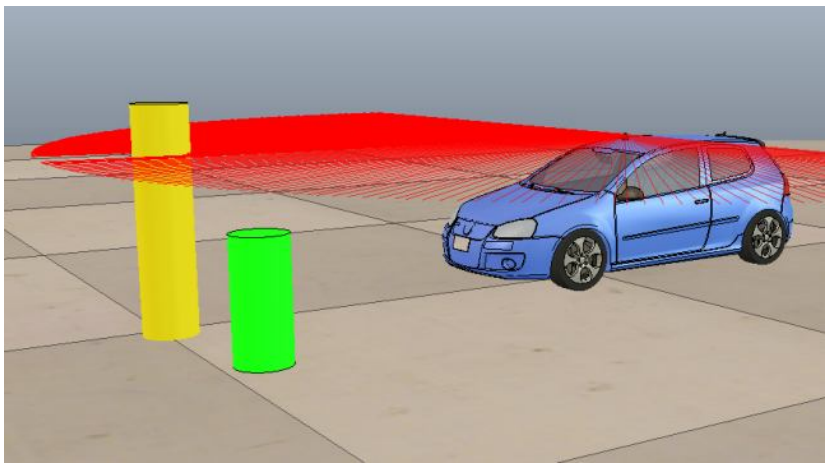
Şekil 4.2'i inceleyecek olursak, birinci resimdeki yol sınırları yolun her iki tarafında bulunan çalılardır. Bu resimdeki mantıksal yol sınırı ile fiziksel yol sınırı aynıdır. Ancak ikinci resimdeki yolu incelediğimizde yolun sol fiziksel ve mantıksal sınırı duvarken, sağ fiziksel sınırı asfaltın bittiği yer, sağ mantıksal yol sınırı ise trafik levhalarıdır.

Şekil 4.2. Yol sınırları

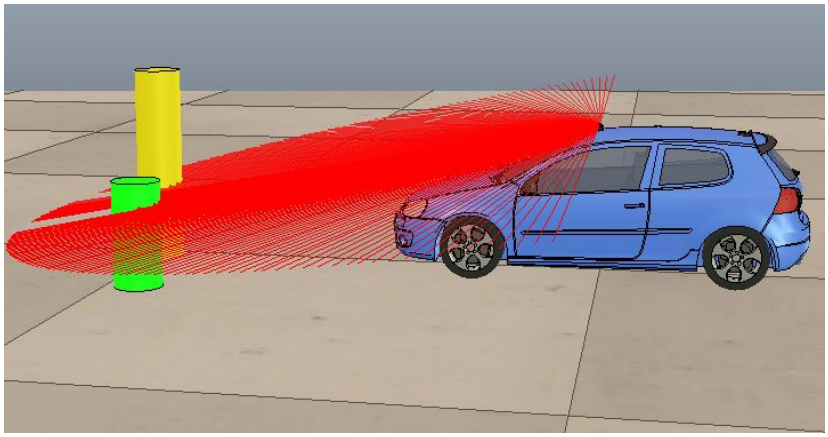
4.2. Lidarın Araç Üzerinde Konumlandırılması

İnsansız kara araçlarında yol sınırlarının tespitinde kullanılan lidar sensörlerinin araç üzerindeki bulundukları konum çok önemlidir. Şekil 4.3'de 180 derece tarama açısı olan 2D lidar sensör aracın tavanına yere paralel olacak şekilde yerleştirilmiştir. Böyle bir posizyonda yerleştirilmiş bir lidar sensörün pitch açısı sıfır derece, yerden yükseliği h cm olarak olarak kabul edelim. Bu şekilde yerleştirilmiş bir sensör ancak h yüksekliği veya h'tan daha yüksek engeller tespit edilebilir. Şekil 4.3'deki örnekte yeşil silindir h yüksekliğinden kısa; sarı silindir ise h yüksekliğinden daha uzundur. h değeri yeşil silindirin boyundan daha büyük olduğudan lidar sensörü ışınları araca zarar verebilecek yeşil silindiri kesmediği için bu engelin araca göre konumu tespit edilememektedir. Diğer taraftan; sarı silindirin yüksekliği h yüksekliğinden daha uzun olduğu için lidar sensörün ışınları bu silindire çarparak geri yansır ve engel olarak algılanmış olur. Bu problemi çözmek için lidar sensörü aracın tavanından tampon seviyesine indirilebilir. Tampon seviyesindeki sensör sarı engelin yanında yeşil engelide tespit eder. Ancak bu durumda bile; hız kesici tümsekler, zeminden daha alçakta araca zarar verecek derinlikte olan çukurların tespiti yapılamaz. Bu sebeplerden dolayı lidar sensörünün bir a pitch açısı kadar eğilmesi bu problemlerin

çözülmesini sağlamaktadır. Şekil 4.4'de lidar sensörünün belli bir açıyla eğildiği durum gösterilmiştir.

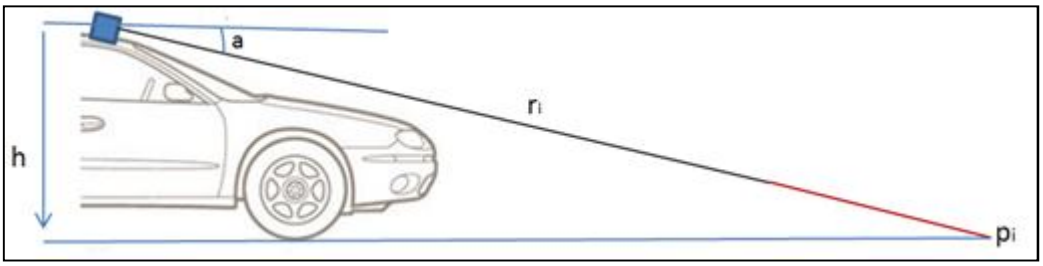


Şekil 4.3. Lidarın yere paralel yerleştirilmesi



Şekil 4.4. Lidarın belli bir pitch açısıyla yerleştirilmesi

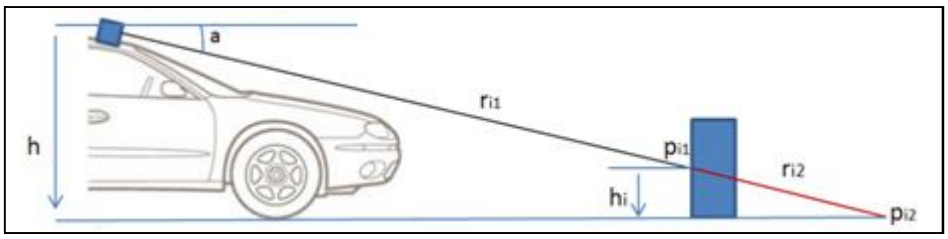
Bu çalışmada engelleri tespit etmek için aracın tavanına bir LMS111 model lidar sensör Şekil 4.5'deki gibi yerleştirilmiştir. Şekil 4.5'de lidarın pitch açısı a derece olarak belirlenmiştir. Lidara verilen bu pitch açısı sayesinde lidar ışınları aracın hareket ettiği zeminden yansıyacaktır. Bu sayede lidarın ürettiği veri işlenerek yol hakkında çeşitli çıkarımlar yapılabilecektir. Lidar sensörünün yerden yüksekliği ise h cm olarak belirlenmiştir. Lidarın pitch açısı ve yerden yüksekliği sensörün tarama menzilini belirler. Sensörün pitch açısını azaltmak ve yerden yüksekliğini artırmak tarama menzilini artırır.



Şekil 4.5. Lidarın araç üzerindeki konumu

Şekil 4.6'de belli bir pitch açısıyla yerleştirilen lidar sensörün taramasının yol üzerinde hangi noktalardan yansıdığı görülmektedir. Lazer ışınları eğer bir pozitif veya negatif engelden yansımıyorsa ölçülmesi gereken uzaklık değeri trigonmetrik fonsiyonlarla rahatlıkla hesaplanabilir.

Şekil 4.6'de aracın üzerine a pitch açısıyla yerleştirilmiş lidardan çıkan bir lazer ışını engel üzerindeki p_{i1} noktasından yansıdığı gözükmektedir. Eğer aracın önünde bu engel olmasaydı lazer ışını p_{i2} noktasından yansıyacaktı.



Şekil 4.6. Bir lidar ışının bir engelden geri yansıması

Lidarın pitch, roll açısı sırasıyla a, b olsun, lidarın yerden yüksekliği h, lidarın çözünürlüğü w ise bir lidar taramasındaki i inci lazer ışının boyu r_i olmalıdır. Trigonometrik hesabı aşağıdadır [25].

$$r_i = r_{i1} + r_{i2} \tag{4.1}$$

$$r_i = h / (\sin w_i \sin a + \cos w_i \cos a \sin b) \tag{4.2}$$

Yukarıdaki Şekilde 4.6'daki gibi bir durumda i inci lidar ışının bir engelden yansıması durumunda, i inci lazer ışınından gelen ölçüm r_{i1} dir. Denklem (4.2)'yi Denklem (4.3) ve Denklem (4.4) gibi yeniden yazarsak:

$$\alpha = \sin w_i \text{ ve } \beta = \cos w_i \cos a \tag{4.3}$$

$$r_{i1} = h'/(\alpha \sin a + \beta \sin b) \quad (4.4)$$

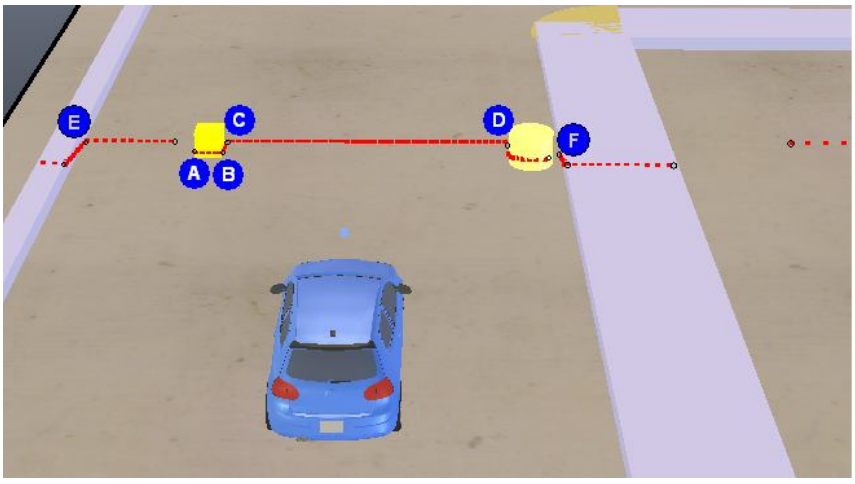
Denklem (4.4)'de h' ve h çekilirse Denklem (4.5) elde edilir.

$$h_i = \Delta h = h - h' = (r_i - r_{i1})(\alpha \sin a + \beta \sin b) \quad (4.5)$$

Şekil 4.6'daki gibi bir durumda i inci lidar ışının bir engelden yansıması durumunda, i inci lidar ışının yansıdığı noktanın yerden yüksekliği Denklem (4.5)'e göre h_i dir. Buradan çıkan sonuç;

- Eğer $h_i = 0$ ise lidardan çıkan bu lazer ışını herhangi bir engelden yansımıyor demektir. Bu durumda lidar ışının yansığı nokta araç ile aynı yükseklikteki bir noktadır. Bu noktanın yol olma ihtimali yüksektir.
- Eğer $h_i > 0$ ise lidardan çıkan lazer ışını aracın bulunduğu seviyeden daha yüksek bir cisimden yansımıştır. Lidar ışının yansıdığı bu nokta araç için bir engeldir. Bu tür engellere pozitif engel denir. Pozitif engellere örnek verecek olursak; yol kenarındaki bariyerler, hız kesiciler, dubalar, vb.
- Eğer $h_i < 0$ ise lidardan çıkan lazer ışını aracın bulunduğu seviyeden daha alçakta bir noktada yansımıştır. Lidar ışının yansıdığı bu nokta araç için bir engeldir. Bu tür engellere negatif engel denir. Negatif engellere örnek verecek olursak; yolda bulunan çukurlar, yol kenarlarının yol seviyesinden aşağıda olması vb.

Şekil 4.7'deki lidar taramasında A ve B noktaları arasındaki ışınların için $r_i > r_{i1}$ ve ışınlar pozitif bir engelden yansımıştır. Ancak C ve D noktaları arasındaki ışınlar için $r_i = r_{i1}$ dir ve bu noktaların yol olması muhtemeldir.



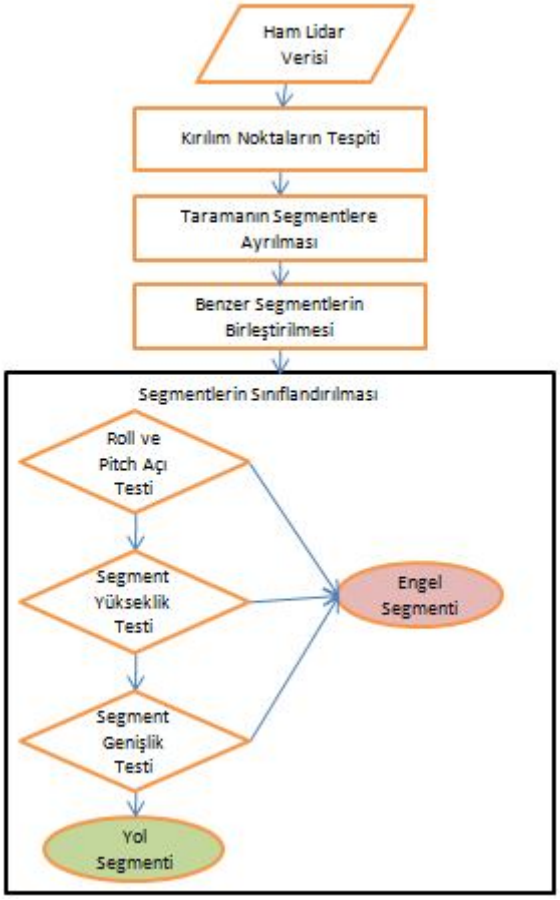
Şekil 4.7. Engellerin bulunduğu bir yolda lidar ışınlarının dağılımı

4.3. Yol Sınırı Tespiti Algoritması Akışı

Lidar sensör kullanarak mantıksal yol sınırlarının tespitinin ana prensibi; daima yol etrafındaki yapılardan daha düzenli, devam eden ve aracın geçebileceği genişlikteki düzlemin bulunmasıdır. Burada bahsi geçen "daha düzenli" kelimesinin anlamı lidar sensöründen arka arkaya çıkan lazer ışınlarının yansıdığı noktaların yükseklik, eğim, uzaklık gibi özeliklerinin liner değişmesidir.

Şekil 4.8'de yol sınırları tespiti için önerilen sistemin akış şeması görülmektedir. Sistemin dışarıdan girdisi ham lidar verisidir. Bu aşamadan sonra ilk adım kırılım noktalarının belirlenmesidir. Kırılım noktası, lidar taramasının ölçülen değerlerinin liner olmayan bir şekilde ani değişimler yaşadığı noktalardır. Şekil 4.6'daki A, B, C, D, E, F gibi noktalar birer kırılım noktasıdır. Kırılım noktası tespiti algoritması Bölüm 4.4'de detaylıca anlatılacaktır. İki kırılım noktası arasında kalan ardışıl ölçümler ise segment olarak isimlendirilmiştir. Segmente örnek olarak A-B, B-C, C-D aralıkları verilebilir.

Kırılım noktalarından elde edilen segmentler çeşitli özelliklerine göre sınıflandırılarak bir yol segmenti mi yoksa engel segmenti mi olduğunu karar verilir. Bölüm 4.5'de sınıflandırma işleminin aşamaları anlatılacaktır.



Şekil 4.8. Sistemin genel mimarisi

4.4. Kırılım Noktası Tespiti

Şekil 4.9'da gösterilen p_n noktaları bir lidar taramasının yansıdığı yüzeydeki noktaları temsil etmektedir. Lidar taraması eğer düzgün bir yüzeyden yansıyorsa bu noktaların lineer bir şekilde değişmesi beklenir. Bu nokta kümesinde noktaların dağılımlarının aniden yüksek oranda değişti noktalara kırılım noktası denir. Doğru çıkarma işlemlerinde kırılım noktası çıkarma yöntemi çok önemlidir. Arka arkaya gelen nokta arasındaki değişimin sebebinin kesin bir farklılaşma olduğunun anlaşılması gerekir. Örneğin bir silindir üzerinden yansıyan noktaların oluşturduğu yapı lineerdir. Ancak noktalar yatay eksende değişirken bir anda dikey eksende değişmeye başladığı nokta bir kırılım noktasıdır.

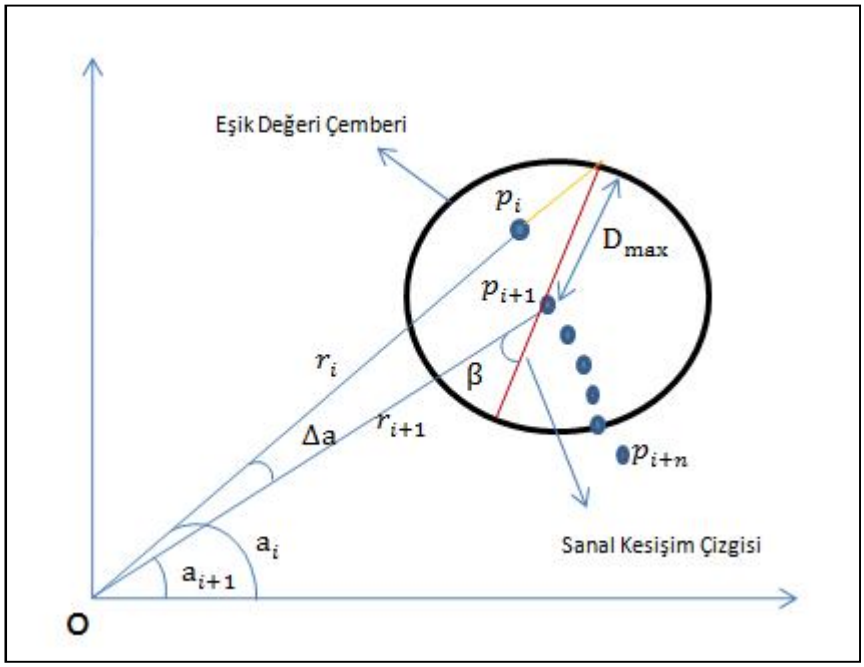
Bu çalışmada adaptif kırılım noktası tespit etme algoritması kullanılmıştır. Bu algoritmaya göre arka arkaya gelen iki tarama noktası arasındaki mesafe belli bir uzaklıktan daha büyükse bu bir kırılım noktasıdır. Şekil 4.10'da gösterilen p_i ve

p_{i+1} numaralı iki ardışıl tarama noktasının kırılım noktası olup olmadığı tespit edilirken adaptif olarak değişen D_{max} değerinden büyüklük küçüklük durumuna bakılır. Kırılım noktası tespit etme algoritmasının sözde kodu aşağıdaki gibidir [24].

```
01: FOR butun tarama noktaları
02: D_max = r_i+1 (sin(β) / sin(β - Δa) -1) + α
03: IF |p_i - p_i+1| > D_max THEN
04: i ve i – 1 inci tarama noktaları kırılım noktası olur.
05: ENDIF
06: ENDFOR
```

Şekil 4.9. Kırılım noktası tespiti pusedo kodu

Bu algoritmaya göre D_{max} değerini etkileyen eşik değeri β açısıdır. β açısının belirlenmesi insansız kara aracının boyutlarına ve hareket kısıtlarına bağımlıdır. α Değeri hata eşik değeridir. Testler sonucunda 0.09 olarak kabul edilmiştir.



Şekil 4.10. Adaptif kırılım noktası tespit etme algoritması

4.5. Segmentlerin Çıkarılması ve Sınıflandırılması

Elde edilen tüm kırılım noktaları arasında kalan tarama noktalarını aday segment olarak isimlendireceğiz. Başlangıç ve bitiş kırılım noktası arasında kalan ölçüm değerleri bağlantılı noktalara ise segment denir. Aday segmentlerin çeşitli özelliklerine bakılarak bir segment olup olmadığına karar verilir.

Aday segmentlerin gerçek segment olması için şu özelliklere sahip olması beklenir: 1) Aday segmentlerin kaç noktadan oluştuğu, 2) Bir aday segmentte bulunan tüm noktaların yüksekliğinin başlangıç ve bitiş noktası yükseklikleri arasında olup olmadığıdır. Eğer aday segment içerisindeki bir noktanın yüksekliği aday segmentin başlangıç ve bitiş noktalarının yüksekliğinden belirlenmiş bir eşik değerinden fazlaysa yüksekliği fazla olan nokta yeni bir kırılım noktası kabul edilir ve aday segment ikiye bölünür. Lidar tarama noktlarının yüksekliğinin nasıl hesaplandığı Bölüm 4.2'de anlatılmıştır. Segment çıkarım sözde kodu aşağıdadır:

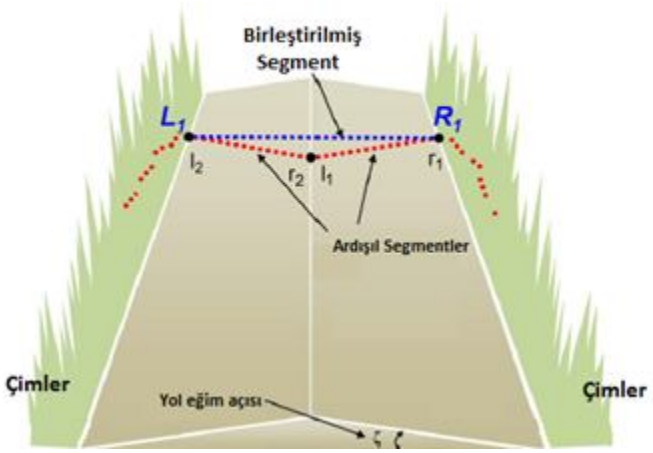
Eğer bir aday segment bu iki özelliği sağlıyorsa artık segment olarak kabuledilir. Segmentler çıkarıldıktan sonra çeşitli özellikleri aynı olan segmentler bileştirilir. Segment birleştirme işlemi bir sonraki bölümde anlatılacaktır. Elde edilen segmentler için bu aşamadan sonra sınıflandırma algoritmasına geçirilir.

01: SET n_s to 1
02: WHILE $n_s \leq$ TaramaNokstasıSayısı
03: SET n_s to n_e
04: INCREMENT n_e
05: WHILE n_e Tarama noktası kırılım noktası değilse
06: INCREMENT n_e
07: IF n_e == TaramaNokstasıSayısı THEN
08: BREAK
09: ENDIF
10: ENDWHILE
11: WHILE $n_e - n_s + 1 >$ *Nmin
12: CALCULATE α ve β denklem (3) göre hesapla
13: FOR Bütün yükseklik değerleri başlangıç ve bitiş noktaları arasında olmalı
14: COMPUTE h_i denklem (5) göre hesapla
15: ENDFOR
16: COMPUTE d_{max}
17: IF $h_{max} > h_{th}$ THEN
18: SET n_e değerine d_{max} tarama noktasının indeksini ata
19: ELSEIF
20: n_s ve n_e gerçek bir segmentin başlangıç ve bitiş noktası
21: BREAK
22: ENDIF
23: ENDWHILE
24: ENDWHILE
*Nmin: Bir segmente olması gereken nokta sayısı

Şekil 4.11. Segment çıkarma işlemi pusedo kodu

4.5.1. Benzer segmentlerin birleştirilmesi

Pratikte yol yüzeyi her zaman dümdüz değildir. Yağmur kar sularını yol yüzeyinde tutmaması için belli bir eğimi vardır. Yada yol yüzeyi zamanla yıpranır ve dezenformasyonlar oluşur, yol bakım çalışmalarında çeşitli yerlere yamalar yapılmış olabilir. Bu gibi sebeplerden dolayı lidar taraması sonucunda elde edilen taramalar tek bir segment olmak yerine bir kaç segmente dönüşmüş olabilir. Algoritmanın daha hızlı çalışması için ardışıl segmentlerin roll ve pitch açıkları arasındaki fark belirlenen eşik değerlerinden daha düşükse iki segment birleştirilip tek bir segment haline getirilir [9].



Şekil 4.12. Ardışıl iki segmentin birleştirilmesi

Şekil 4.12'deki kırmızı noktalar lidar ışınlarının yansıdığı noktalardır. Başlangıç bitiş noktası l_1 ve l_2 olan segment ile başlangıç bitişi r_2 ve r_1 olan segmentler ardışıl segmentlerdir. Bu iki segmentte yol segmentidir. Fakat yol eğiminden dolayı farklı iki segmente dönüşmüşlerdir. Bu iki segmentin roll ve pitch açıları farkı eşik değerinden daha düşük olduğu için birleştirilirler ve sonuç olaran başlangıç bitiş noktası sırasıyla L_1 ve R_1 olan yeni bir segment elde edilmiş olur. Bu aşamadan sonra elde edilen segmentlerin çeşitli özellik çıkarımı yapılarak yol mu yoksa engel mi olduğuna bakılır.

4.5.2. Segment roll ve pitch kontrolü

Elde edilen segmentlerin yol segmenti mi yoksa bir engel segmenti mi olduğunu anlamak için segmentin özelliklerinden bakılan ilk kriter roll ve pitch açı özellikleridir. İdeal bir yolun roll ve pitch açısındaki değişim çok fazla değildir [9]. Bu sebepten dolayı elde edilen segmentlerin roll ve pitch açıları hesaplanır. Önceden belirlenen eşik değerleriyle kıyaslanır. Bu kıyaslama sonucunda;

- Eğer karşılaştırılan segmentin roll ve/veya pitch açısı belirlenen eşik değerinden fazlaysa bu bir engel segmentidir.
- Eğer roll ve picth açı değeri belirlenen eşik değerinin altında ise bu bir yol segmenti olabilir bu sebeple bir sonraki adım olan segment yükseklik testine geçilir.

Bu çalışmada roll (Φ_{max}) ve pitch (θ_{max}) açı eşik değerleri sırasıyla 4 ve 2 derece olarak belirlenmiştir. Eşik değeri belirlenirken aracın tırmanma ve kayma gibi parametreleri dikkate alınır.

4.5.3. Segment yükseklik kontrolü

Roll ve pitch açı kontrolünden geçen segmentler için yükseklik kontrolü yapılır. Bu aşamada kalan segmentlerin yükseklik değerleri aracın aşabileceği H_{max} değerinden küçük mü diye bakılır. H_{max} yükseklik değeri aracın teker yarı çapı ve yere maksimum ne kadar yakın olması değerine göre belirlenir. Gerçek ortam testlerinde kullanılan AKAY01 test aracının için bu değer 0.05 m olarak belirlenmiştir. Elde edilen bir segmentin maksimum yüksekliği H_{max} ile karşılaştırılır.

- Eğer segment değeri H_{max} değerinden büyükse bu segment engel olarak sınıflandırılır.
- Eğer segment değeri H_{max} değerinden küçük ve eşitse bu durumda bir sonraki aşama olan segment genişlik kontrolüne geçilir.

4.5.4. Segment genişlik kontrolü

Bir segmentin yol veya engel segmenti olduğuna karar verilen son kontrol segment genişlik kontrolüdür. Yükseklik testinden geçen segmentlere genişlik kontrolü yapılır. Bu aşamada segmentlerin genişlikleri yol genişlik eşik değeri ile karşılaştırılır. Yol genişlik eşik değeri (L_{min}) aracın genişliğine göre belirlenir. Testlerde kullanılan AKAY01 için bu değer 1.5 m dir. Bu aşamada segment genişlikleri hesaplanır ve L_{min} değeri ile karşılaştırılır.

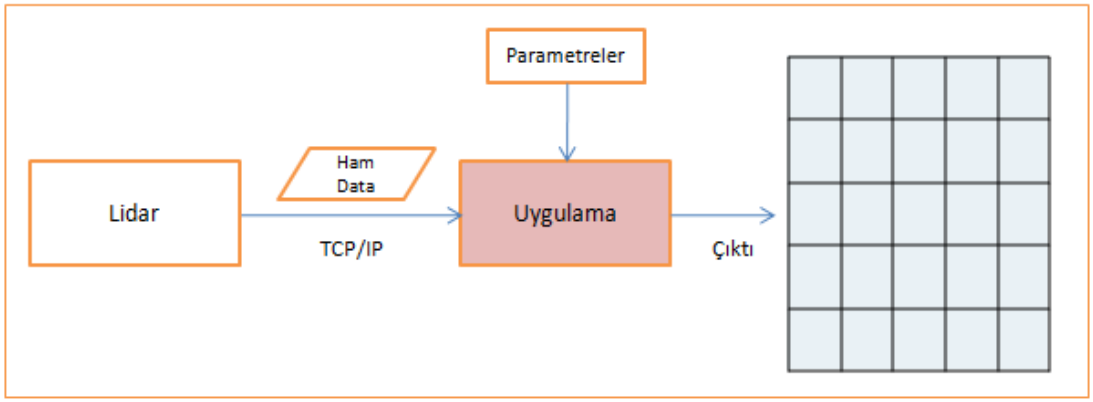
- Eğer segment genişliği L_{min} değerinden büyük ve eşitse bu bir yol segmentidir.
- Eğer segment genişliği L_{min} değerinden küçükse bu bir engel segmentidir.

5. UYGULAMA

Önerilen yol çıkarımı algoritmasını kullanmak ve sonuçlarını göstermek için bir uygulama geliştirilmiştir.

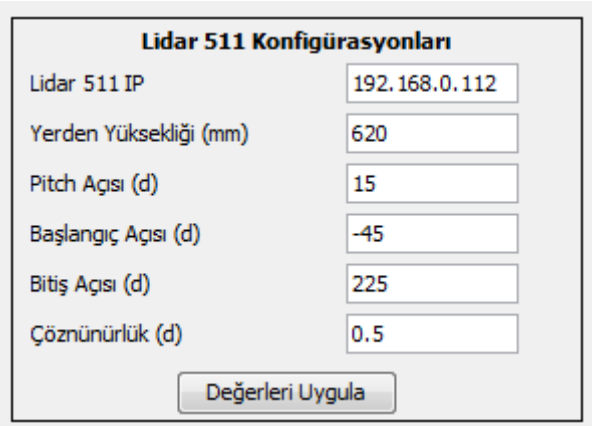
5.1. Geliştirilen Uygulama

Ham lidar verisini işlemek ve önerilen algoritmaları test etmek için java programlama dilinde bir uygulama geliştirilmiştir. Ham lidar verisinin yapısıdan ekler kısmındaki Ek-A bölümünde bahsedilmiştir. Geliştirilen uygulama TCP/IP protokolü üzerinden bağlanan ve parametreleri ayarlanan bir lidar sensörle doğrudan çalışabilir. Uygulama girdi olarak ham lidar datası alır çıktı olarak ise yine TCP/IP protokolü üzerinden bir çıktı matrisi (Occupacy Grid Map) sağlamaktadır. Uygulama bu özelliği sayesinde farklı platformlarda TCP/IP protokolü üzerinden doğrudan çalıştırılabilir.



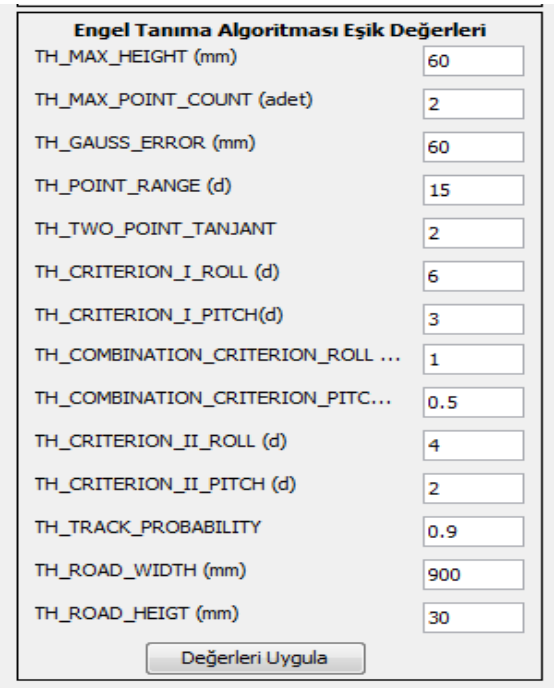
Şekil 5.1. Uygulama mimarisi

Şekil 5.1'de geliştirilen uygulamamanın mimarisi gösterilmektedir. Yukardaki mimariye göre uygulama, ham lidar datasını TCP/IP protokolü üzerinden almaktadır. Uygulamanın hangi IP üzerinden veri alacağı ve lidar sensörün özellikleri Şekil 5.2'deki lidar sensör özellik belirleme ekranından girilmektedir.



Şekil 5.2. Lidarın özelliklerinin girildiği ekran

Geliştirilen uygulama sensör özellik belirleme arayüzü sayesinde lidar sensör kullanılan başka bir araç üzerinde doğrudan çalıştırılabilmektedir. Şekil 5.2'deki lidar bilgileri sağlandıktan sonra uygulama çalıştırıldığında bir çıktı matrisi üretilmektedir. Üretilen çıktı matrisi ise istenildiği taktirde TCP/IP üzerinden yayınlanabilir veya uygulama ekranından gerçek zamanlı olarak görüntülenebilmektedir. Uygulamada algoritmanın parametreleri kolaylıkla değiştirilerek sonuçları çıktı ekranında görselleştirilmektedir.

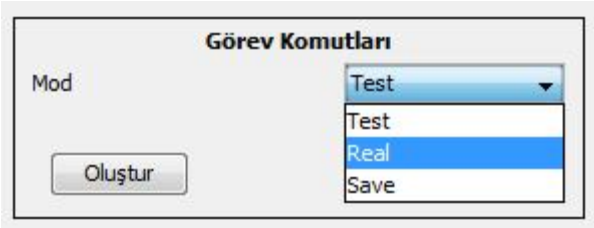


Şekil 5.3. Parametreleri değiştirme ekranı

Bu ekran vasıtasıyla algoritma parametreleri algoritma çalışırken değiştirilerek etkisi doğrudan görüntülenebilmektedir.

Uygulamadaki diğer bir önemli özellik ise uygulama çalışma modu seçeneğidir. Uygulamanın 3 tip çalışma modu vardır.

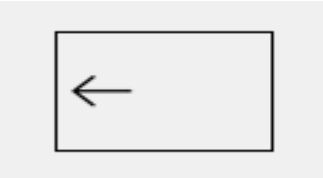
a) Save mode: uygulama bu modda çalışırken bağlı olduğu lidar sensörden aldığı ham lidar datasını bir txt formatında günün tarih saatine göre kaydeder.
b) Test mode: uygulama test modda çalışırken daha önceden save modda kaydedilmiş bir ham lidar datası üzerinden tekrar tekrar çalıştırılabilir. Bu sayede algoritma parametrelerini güncellemek için lidar ve diğer donanımlar tekrar tekrar çalıştırılmamış olur.
c) Real mode: uygulama real modda çalışırken sistem gerçek zamanlı yol sınırlarını tespit eder ve elde edilen çıktı matrisini otonom hareket için ihtiyacı olan diğer algoritmalara sağlar.



Şekil 5.4. Çalışma modu seçme ekranı

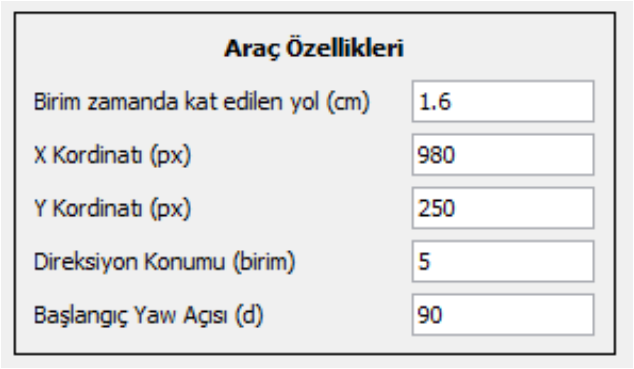
Şekil 5.4'de görüldüğü uygulama mod seçimi basit bir seçme kutucuğundan yapılmaktadır. Bu seçim işlemi yapıldıktan sonra sistem ne yapacağına karar vermektedir.

Önerilen sistem bir kara aracında test edilecektir. Uygulamada bu kara aracını temsilen uygulamaya göre ölçeklendirilmiş bir dikdörtgen kullanılmıştır. Şekil 5.5'deki dikdörtgen aracın uygulamadaki temsilidir.



Şekil 5.5. Aracın temsili

Uygulamada bulunan bir diğer özellik ise aracın hız ve konum bilgileri doğrudan uygulama araç konfigürasyonu düzenleme ekranından yapılabilmektedir.



Şekil 5.6. Araç özelliklerini güncelleme ekranı

5.1.1. Uygulama çıktı matrisi

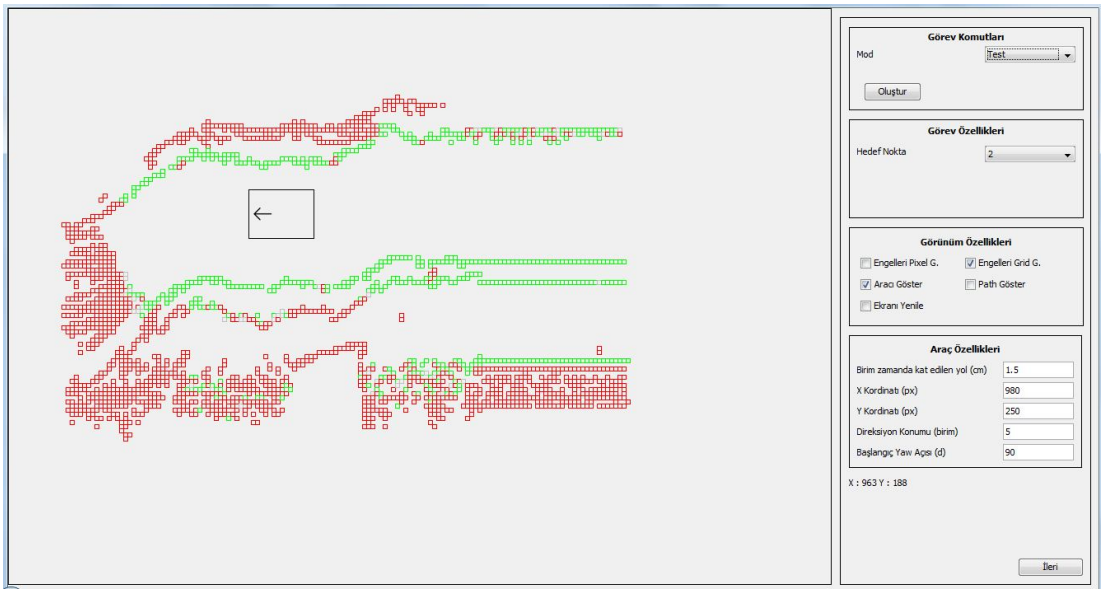
Uygulamada ham lidar verisi alındıktan sonra Bölüm 4'deki algoritmalardan geçirilerek işlenmiş lidar verisi elde edilir. İşlenmiş lidar verisini kullanmak için bir data formuna dönüştürmek gerekmektedir. İşlenmiş lidar verisini tutmak için genellikle iki tür yöntem benimsenmiştir occupancy grid map [31] veya özellik haritası [32] şeklinde. Bu çalışmada işlenmiş lidar verisini tutmak için occupancy grid map yöntemi kullanılmıştır. Bir çok çalışmada haritalama işleminde occupacy grid map kullanılmaktadır [31]. Occupancy grid map her bir gözü gerçek ortamdaki belli bir alana karşılık gelmektedir. Şekil 5.7'deki ekran çıktısında herbir matris gözü yaklaşık olarak 25 cm^2 lik alanı temsil etmektedir. İşlenmiş lidar verisindeki herbir elemanın konumu aracın konumu orjin kabul edilerek (x, y) şeklinde hesaplanır. Elde edilen bu (x, y) konumdaki bölgenin engel mi yoksa yol sınırımı olduğu işlenmiş lidar verisinde meta data olarak tutulur ve haritanın o bölgesinin değeri bu meta dataya göre belirlenir.Bu meta data değerine göre (x, y) konumundaki matris gözü;

a) Engel olarak tespit edilmişse matrisin kutucuğunun değeri -1

b) Yol sınırı olarak tespit edilmişse 1

c) Eğer matrisin o bölgesi hakkında herhangi bir tespit yapılmamış ise değeri 0 olarak belirlenir.

Şekil 5.7'de görüldüğü gibi oluşturulan çıktı matrisinin (occupancy grid map) değerlerine göre uygulmanın ekranında matris gözleri renklendirilir.



Şekil 5.7. Uygulama çıktı matrisnin görselleştirilmiş hali

Uygulamdaki ekranda matrisin renk kodları şu şekildedir;

i. Eğer matrisin kutucuğunun değeri -1 ise o kutucuk kırmızı.

ii. Eğer matrisin kutucuğunun değeri 1 ise o kutucuk yeşil.

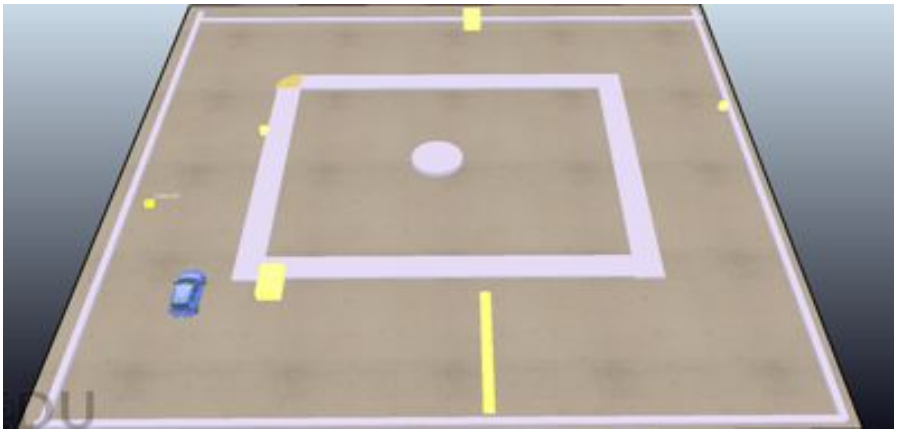
iii. Eğer matrisin kutucuğunun değeri 0 ise o kutucuk renksizdir.

6. DENEY VE SONUÇLAR

Bu bölümde önerilen algoritmanın simülasyon ve gerçek test ortamında yapılan deneylerinin parametre ve sonuçları anlatılacaktır. Önerilen algoritmaların eşik değerlerini ayarlamak, zaman ve maliyetten kazanmak için önce simülasyon ortamında testler yapılmıştır. Algoritmaların eksikleri simülasyon ortamında giderildikten sonra gerçek ortamda da test edilmiştir.

6.1. Simülasyon Ortamı Testleri

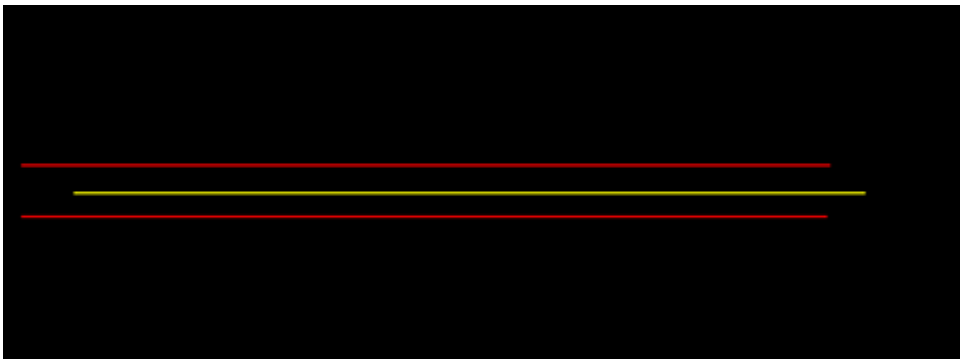
Bu çalışmada geliştirilen algoritma gerçek araçta test edilmeden önce V-rep simülasyon ortamında test edilmiştir. Gerçeklenen algoritmaları koşturmak için üç boyutlu bir simülasyon ortamı kurulmuştur. Şekil 6.1'deki mor cisimler kaldırımları sarı cisimler ise engelleri temsil etmektedir. Oluşturulan üç boyutlu modeldeki engellerin, kaldırımların boyutları gerçek ortamla birebir aynıdır. Simülasyon ortamında kullanılan araç wolksvogen markasının golf modeli bir otomobilin hareket kabiliyeti ve boyut olarak yaklaşık birebir üç boyutlu modelidir. Model üzerine 2 boyutlu tarama yapan bir lazer sensör 10 derecelik pitch açısıyla 1 metre yüksekliğe yerleştirilmiştir.



Şekil 6.1. V-rep simülasyon ortamında oluşturulan 3 boyutlu ortam

Simülasyon ortamında aracın hareketi bilgisayar klavyesiyle sağlanmaktadır. V-rep simülasyon ortamıyla java programla dili arasında v-rep remote api aracılığıyla

haberleşme sağlanmıştır. Bu haberleşme alt yapısı kullanılarak v-rep ortamında yönetilen araç üzerindeki lazer sensörden alınan veri işlenerek java ortamında görselleştirilmiştir.



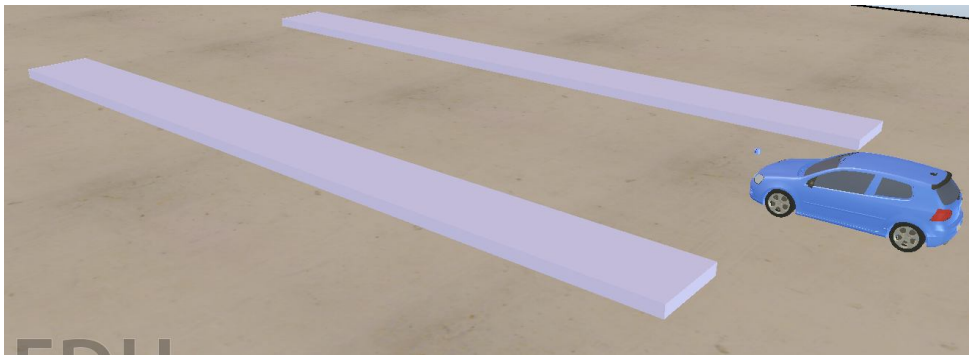
Şekil 6.2. Vrep test ortamında algoritma çıktısı

Şekil 6.2'de v-rep simülasyon ortamında çalıştırılan algoritmanın bir ekran çıktısı görülmektedir. Kırmızı pikseller yol sınırlarını temsil etmektedir. Sarı çizgi ise aracın takip etmiş olduğu güzergahı göstermektedir.

Tablo 6.1. Algoritmanın v-rep ortamındaki eşik değerleri

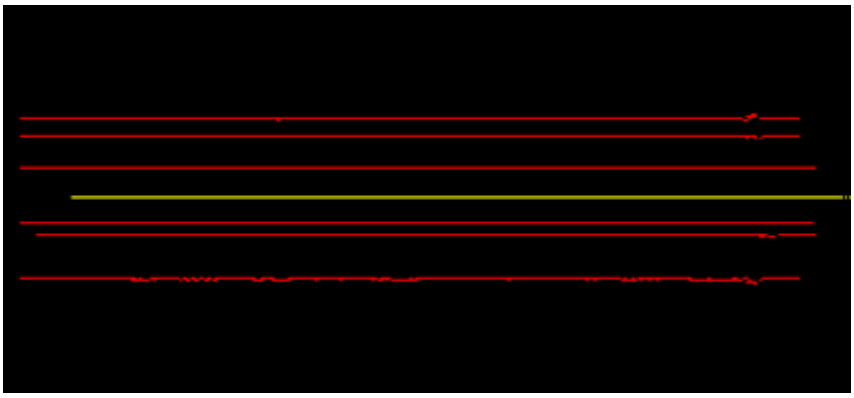
Eşik Değeri	Değeri	Açıklama
h	1.6 metre	Lidarın yerden yüksekliği
a	10 derece	Lidarın pitch açısı
c	1 derece	Lidarın çözünürlüğü
θ	3 derece	Yol segmenti maksimum yaw açısı değeri
β	6 derece	Yol segmenti maksimum roll açısı değeri
width	2 metre	Yol segmenti minumum yol genişliği değeri
height	0.1 metre	Yol segmenti maksimum yerden yüksekliği
β_b	15 derece	Kırılım noktası maksimum uzaklık açısı
θ_c	0.5 derece	Ardışıl iki segment birleştirme maks yaw açısı
β_c	1 derece	Ardışıl iki segment birleştirme maks roll açısı

Tablo 6.1'de simülasyon ortamında kullanılan eşik değerleri listesi ve değerleri sıralanmıştır. Bu değerler belirlenirken aracın ve yolun özellikleri dikkate alınarak belirlenmiştir. Gerçek ortamda kullanılan araç ve test ortamı farklı olduğu için bu eşik değerleri farklı olacaktır.



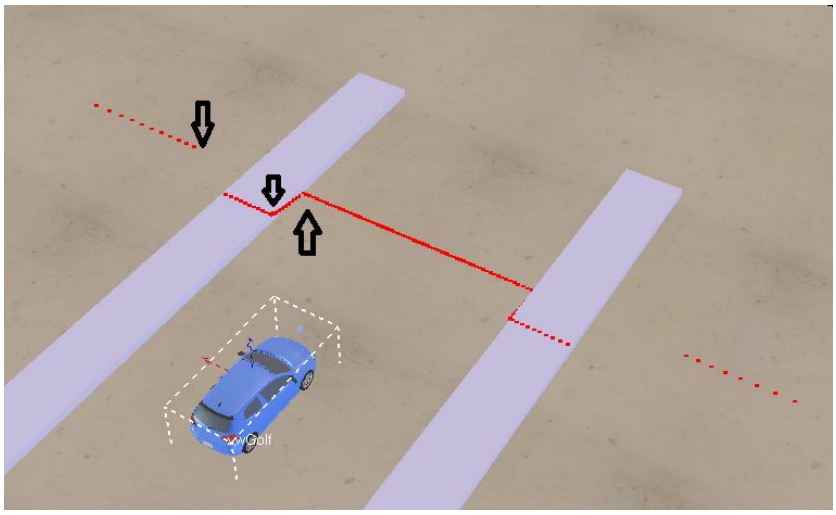
Şekil 6.3. Simülasyon ortamında aracın iki tarafında kaldırım bulunan yol

Şekil 6.3'de oluşturulan 3 boyutlu ortamda iki tarafında kaldırım bulunan bir yol simüle edilmiştir. Test seneryosuna göre araç bu yol boyunca bir v hızıyla hareket edecektir. Bu hareketi esnasında yol sınırları tespit edilecektir. Böyle bir seneryoda algoritmanın çıktısı Şekil 6.4'deki gibi olmuştur.



Şekil 6.4. İki taraflı kaldırımlı bir yolda algoritmanın çıktısı

Algoritmanın iki tarafıda kaldırımlı bir yoldaki çıktıları oldukça düzgündür. Yol sınırları ortadaki sarı çizginin iki tarafındaki kırmızı çizgilerdir.

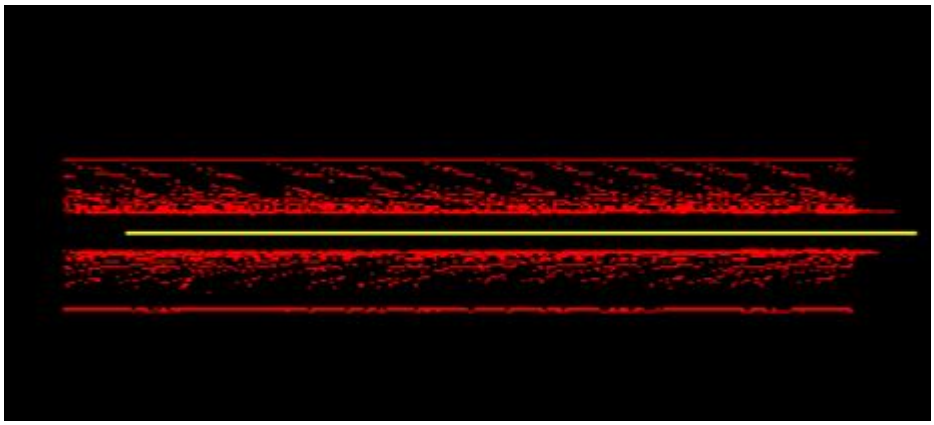


Şekil 6.5. İki tarafı kaldırımlı yolda kırılım noktaları

Şekil 6.5 ise iki tarafı kaldırımlardan oluşan yoldaki kırılım noktlarına sebep olan bölgeler gösterilmektedir. Yolun sol tarafında gösterilen siyah oklarla gösterilen noktalar lazer ışınlarındaki ani kırılımların yaşandığı bölgelerdir. Algoritmanın çıktısında görülen kırmızı pikseller siyah oklarla gösterilen noktalardır. Bunlardan en uçta bulunan kırılım noktası ise yolun sol sınırı olarak başarılı bir şekilde sınıflandırılmıştır.

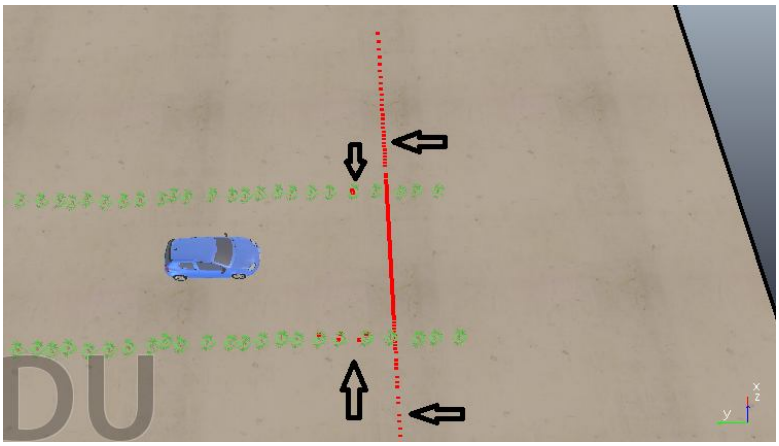
Şekil 6.6. Etrafında çimler bulunan bir yol

Yol sınırı tespit edilirken önemli problemlerden bir taneside kaldırım olmayan yerlerdeki yol sınırlarının tespitdir. Çünkü yol kenarlarında bulunan çimler kaldırımlar kadar düzgün seyretmez. Simülasyon ortamında kaldırımlar kaldırılarak yolun her iki tarafınada bitki örtüsü Şekil 6.6'daki gibi yerleştirilmiştir.



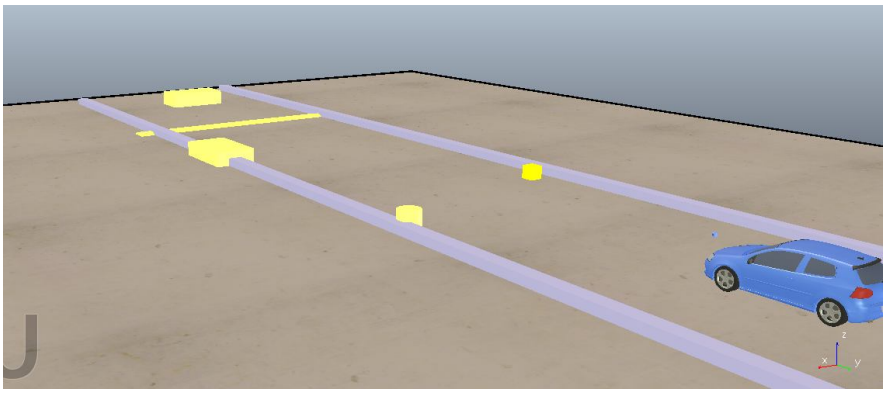
Şekil 6.7. İki tarafında çimler olan yolda algoritma çıktısı

Şekil 6.7'de iki tarafında da bitki örtüsü bulunan bir yolda algoritma çalıştırıldığında oluşan ekran görüntüsü görülmektedir. Algoritmada bu ortamda da oldukça başarılı sonuçlar alınmıştır. Şekil 6.7'de yol sınırları oldukça net bir şekilde görülmektedir.



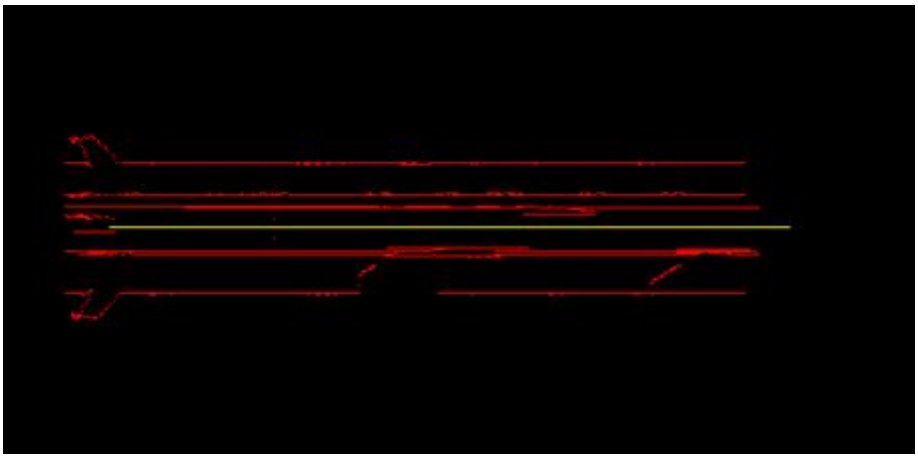
Şekil 6.8. Tespit edilen kırılım noktaları

Şekil 6.8 siyah oklarla gösterilen noktalarda lidar ışınlarının dağılımları bozukmaktadır. Lidar ışınları bu noktalarda pozitif bir engelden yansımaktadırlar. Algoritmada kırımızı piksellerin dağınık şekilde gözükmesinin nedenide çimlerin rasgele dağılmış olmalarından kaynaklanmaktadır.



Şekil 6.9. Düz ve engelli bir yol modeli

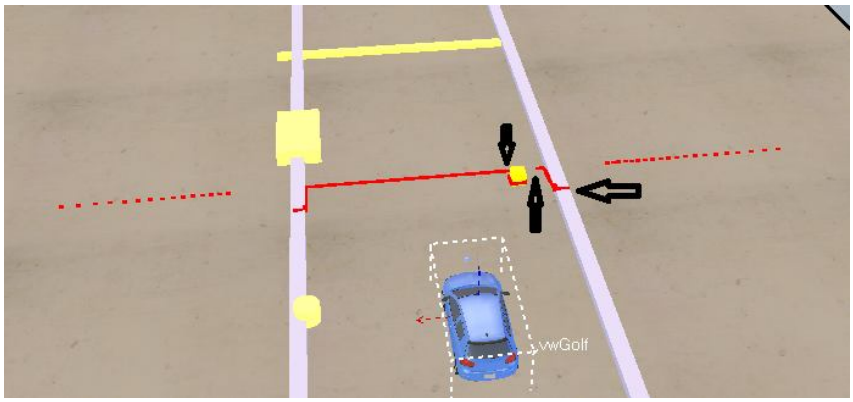
Şekil 6.9'da ise engellerin bulunduğu bir yol modeli oluşturulmuştur. Bu ortamda yolun her iki tarafında kaldırım, yol içersinde sarı renkli engeller yerleştirilmiştir. Yolu soldan sağa kesen sarı engel ise bir hız kesicidir. Yerden 5 cm yüksekliğinde kavisli bir engeldir.



Şekil 6.10. Engelli bir yolda algoritma çıktısı

Şekil 6.10'da algoritmanın engelli bir yolda çalıştırılmasının sonuçları gösterilmektedir. Yolun sağında ve solunda bulunan engeller bir miktar iç kısmında olduğu gözükmektedir.Algoritma başarılı bir şekilde engelleri tespit etmiştir. Yolun sonunda ise yolu tamamen kapatan bir engel gözükmektedir. Yolun son kısmına doğru ise hız kesici için 2 adet kırmızı piksel gözükmektedir. Algoritmada bu hız kesiciyi tespit etme kısmında eksiklikler bulunmaktadır.

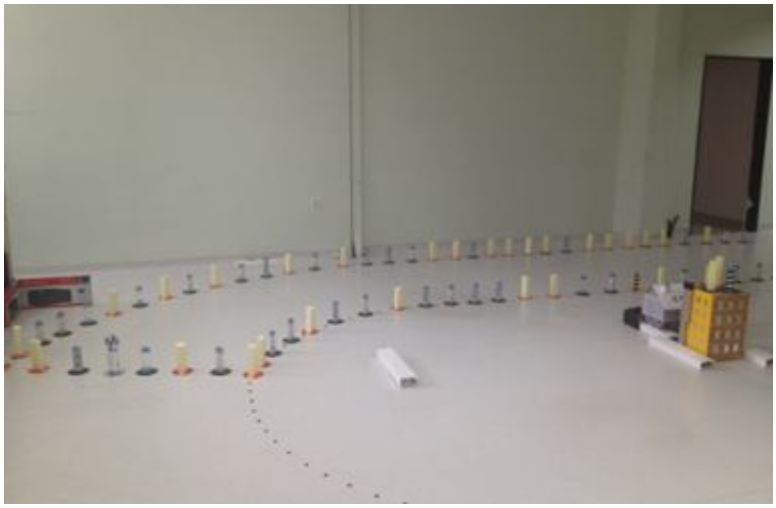
Şekil 6.11 ise engelli yolda bulunan bir engeldeki lazer ışınlarının yansıdığı noktalar gözükmedir. Taramadaki bu keskin değişimlerin yaşandığı noktalar Şekil 6.10'da bazı piksellerin yolu daraltığı güzükmektedir.



Şekil 6.11. Engelli yolda kırılımlara sebep olan noktalar

6.2. Gerçek Ortam Testleri

Simülasyon ortamında doğrulanan algoritmalar otonom bir araç olan AKAY01 aracında kontrollü bir ortamda test edilmiştir. Test ortamına dubalar, levhalar ve hız kesiciler gibi gerçek yollarda bulunabilecek engeller yerleştirilmiştir.



Şekil 6.12. İç test ortamı

Gerçek ortam testlerinde kullanılmak üzere bilgisayarla kontrol edilebilen Şekil 6.13'deki akülü AKAY01 otonom aracı kullanılmıştır. Bu araç TÜBİTAK tarafından geliştirilmiştir.

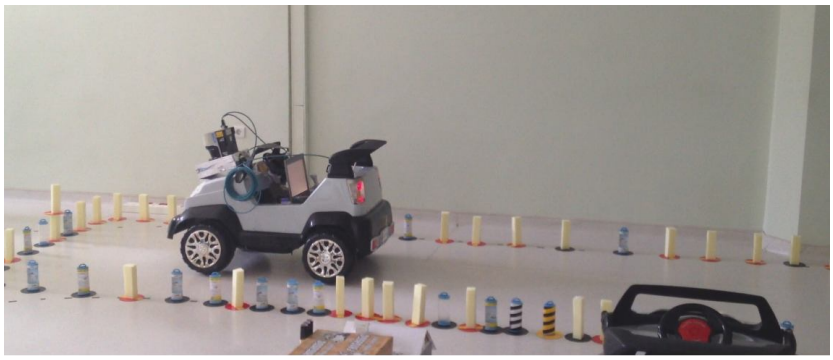
Şekil 6.13. Gerçek ortam testlerinde kullanılan araç

Gerçek ortamda kullanılan araçtaki algoritmanın eşik değerleri aracın boyutları, tipi ve test ortamı farklı olduğu için küçük araca uygun olacak şekilde değiştirilmiştir.

Tablo 6.2. Gerçek ortam eşik değerleri

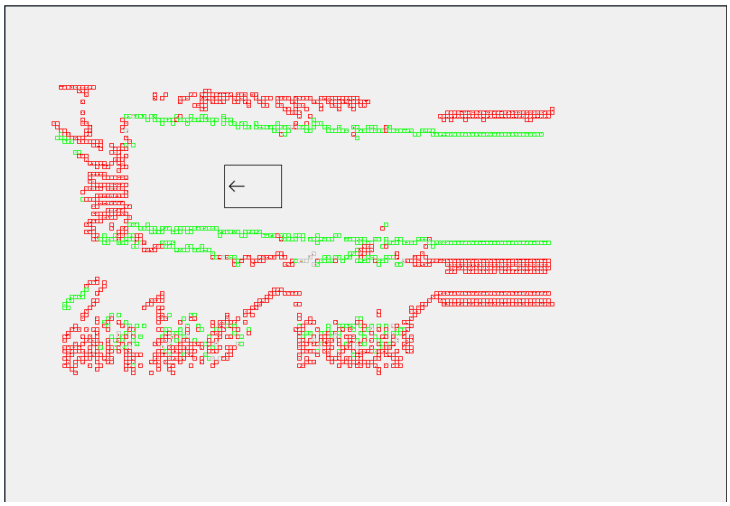
Eşik Değeri	Değeri	Açıklama
h	0.62 metre	Lidarın yerden yüksekliği
a	14.8 derece	Lidarın pitch açısı
c	0.5 derece	Lidarın çözünürlüğü
θ	3 derece	Yol segmenti maksimum yaw açısı değeri
β	6 derece	Yol segmenti maksimum roll açısı değeri
width	1 metre	Yol segmenti minumum yol genişliği değeri
height	0.5 metre	Yol segmenti maksimum yerden yüksekliği
$\boldsymbol{\beta_b}$	15 derece	Kırılım noktası maksimum uzaklık açısı
$\boldsymbol{\theta_c}$	0.5 derece	Ardışıl iki segment birleştirme maks yaw açısı
$\boldsymbol{\beta_c}$	1 derece	Ardışıl iki segment birleştirme maks roll açısı

Gerçek ortamda gerçekleştirilen ilk test yolun iki tarafında trafik dubaları bulunan düz bir yolda yol sınırlarının tespit edilmesidir. Bu test işlemi için Şekil 6.14'deki gibi bir ortamı hazırlanmıştır. Oluşturulan bu test ortamında araç otonom bir şekilde hareket ederken yol sınırları tespit edilmiştir.



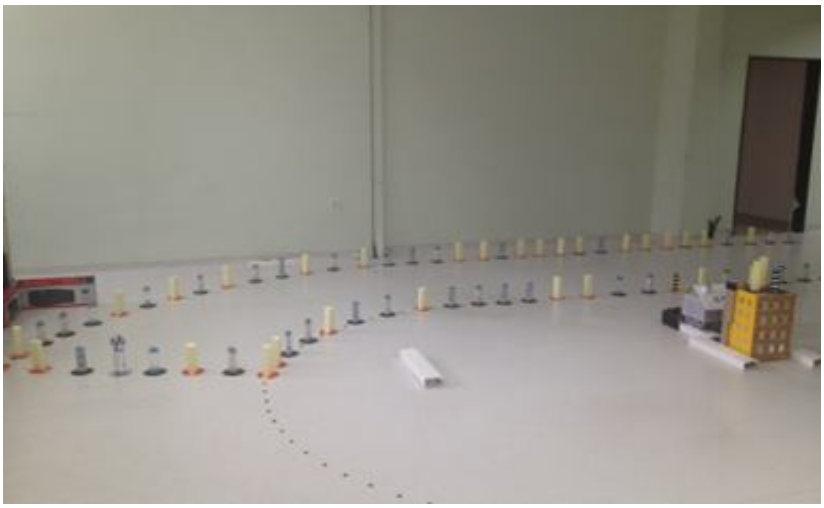
Şekil 6.14. İç ortam yol sınırları tespit edilen düz dubalı yol

Şekil 6.14'de görülen ortamda test edilen yol sınırları tespit etme algoritmasının sonuçları Şekil 6.15'deki gibidir. Şekil 6.15'de görülen kırmızı bölgeler engellerin bulunduğu alanları temsil etmektedir. Yeşik bölgeler ise tespit edilen yol sınırlarıdır. Yolun sonu dubalar ile kapalı olduğu için o bölgede kırımızı (engel var) olarak tespit edilmiştir.



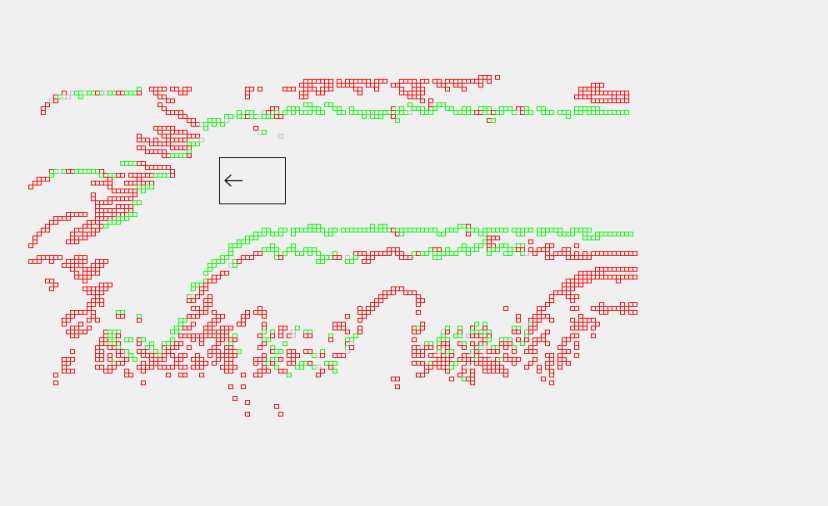
Şekil 6.15. İç ortam dubalı yol uygulama ekran çıktısı

Şekil 6.16'da ise test ortamında yol tasarlanırken yola önceden belirlenen bir oranda viraj verilmiştir. Böyle bir ortamda araç viraja girmeden önce yolun dönmeye başladığını tepit etmesi gerekmektedir. Önerilen algoritmanın bu ortamdaki başarısı test edilmiştir.



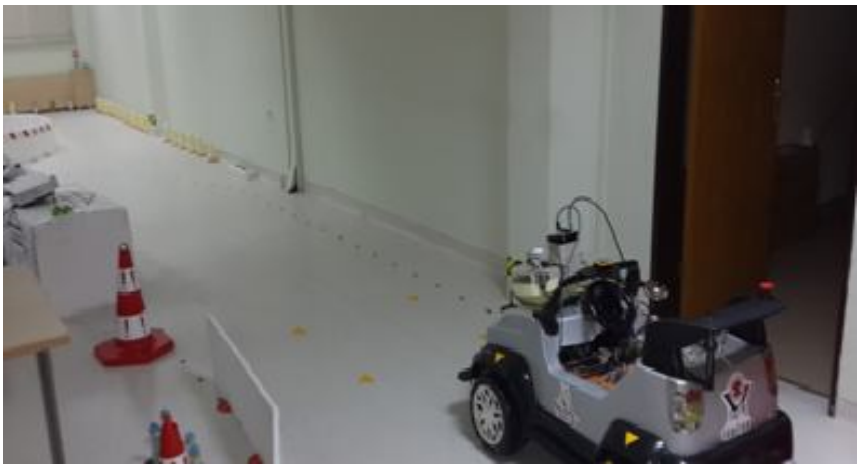
Şekil 6.16. İç ortam yol sınırları tespit edilen virajlı dubalı yol

Şekilde 6.17'de ise önerilen algoritmanın çıktısı görülmektedir. Başarılı bir şekilde yolun virajlı kısmıda tespit edilmiştir. Virajın sonunda yol kapalı olduğu için o bölge kırmızı (yol kapalı) olarak tespit edilmiştir.



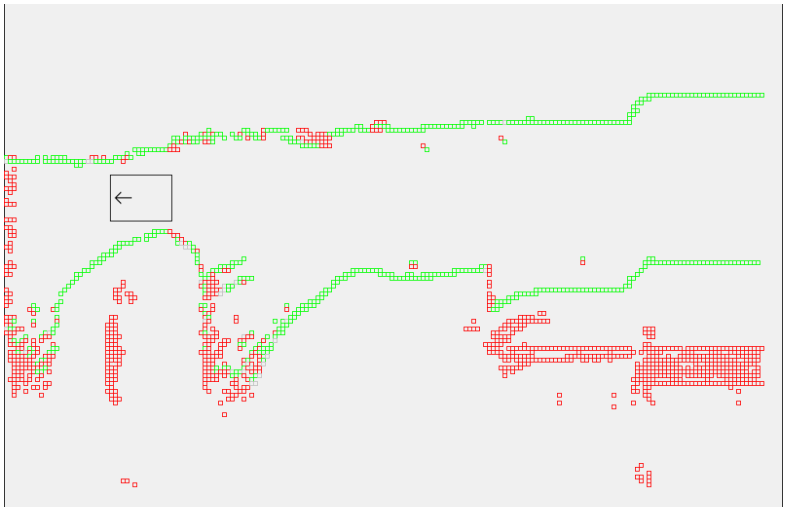
Şekil 6.17. İç ortam virajlı dubalı yol uygulama ekran çıktısı

Önerilen algoritma son olarak iç test ortamında oluşturulan bir tarafında duvar diğer tarafında ise virajlar, dubalar ve yapay duvarlardan oluşan bir parkurda test edilmiştir. Bu ortamın özelliği farklı yapıların bir arada bulunmasıdır. Bazı yerlerde yol daralmakta bazı yerlerde dubalarla yol kapatılmıştır.



Şekil 6.18. İç ortam kenarları duvar ve köpük olan uzun parkur

Şekil 6.18'de oluşturulan test ortamının bir resmi görsterilmektedir. Bu ortamda yol ileride daralmaktadır. 3 metre sonra yolda sola doğru 90 derecelik bir bir viraj varıdr. Aracın hareketinden 1 metre sonra yolun sol kısmında bir trafik dubası ve son olarak yol bir duvarla sonlanmaktadır.



Şekil 6.19. İç ortam uzun parkur uygulama ekran çıktısı

Şekil 6.18'deki ortamda çalıştırılan algoritmanın çıktısı Şekil 6.19'daki gibi olmuştur. Çıktı incelendiğinde oluşturulan test ortamıyla birebir olduğu görülecektir.

Geliştirilen algoritmalar farklı ortamlarında test edilmiş ve başarılı sonuçlar elde edilmiştir.

7. SONUÇLAR VE ÖNERİLER

Bu çalışmada insansız kara araçları için yol sınırları tespiti ile ilgili yeni bir yöntem önerilmiştir. Önerilen yöntemde lidar sensörü belli bir açı ile yere eğimli olarak araca yerleştirilmiş ve taramalar elde edilmiştir. Elde edilen taramalar önerilen algoritmalardan geçirilerek yol sınırı tespiti yapılmıştır. Geliştirilen algoritmalar önce simülasyon ortamında test edilmiş ve başarılı sonuçlar elde edilmiştir. Simülasyon ortamında elde edilen başarılı sonuçlardan sonra sistem AKAY01 isimli gerçek bir otonom araç üzerinde test edilmiş ve sonuçları paylaşılmıştır. Elde edilen başarılı sonuçlara göre önerilen algoritmalar insansız kara araçlarında yol sınırları tespiti adımında kullanılabilecektir.

Geliştirilen algoritmanın yol sınırları tespiti kısmı farklı sistemlerde de kullanılabilmesi için parametrik bir yapıda yazılmıştır. Bu yapı sayesinde geliştirilen yol sınırı tespit etme algoritması farklı projelerde kullanılması oldukça kolaydır. Sistem TCP/IP üzerinden ham lidar verisini almakta ve algoritmanın çıktısı olan yol sınırlarını içeren çıktı matrisini aynı şekilde TCP/IP üzerinden sunmaktadır.

Gelecekte yapılacak çalışmalardan bir tanesi önerilen algoritmanın Robot Operating System (ROS) ile uyumlu bir paket haline getirilmesidir. Bu uyumluluk sayesinde sistem herhangi bir robotik uygulamada girdileri sağlandığı taktirde doğrudan kullanılabilecektir. ROS uyumlu hale getilen algoritma açık kaynak kod olarak diğer robot geliştiricilerin kullanımına sunulacaktır.

Bu çalışmanın eksik yönlerinden bir tanesi hata filtresi bulunmamasıdır. Gelecek çalışmalarda bu algoritma üzerine bir hata filtresi eklenmesi planlanmaktadır. Gelecekte yapılcak bir diğer uygulama farklı eğim açılarında birden fazla lidar sensör kullanılarak yol sınırları tespiti yapılmasıdır

KAYNAKLAR

[1] Urmson C., Anhalt J., Bagnell D., Autonomous driving in urban environments: Boss and the urban challenge, *Journal of Field Robotics*, 2008, **17**, 426-468.

[2] Choi J., Lee J., Kim D., Environment detection and mapping algorithm for autonomous driving in rural or off-road environment, *IEEE Transactions on Intelligent Transportation Systems*, 2012, **13**, 974-983.

[3] Wijesoma W.S., Kodagoda K. R. S., Balasuriya, A. P., Road-boundary detection and tracking using ladar sensing, *IEEE Transactions on Robotics and Automation,* 2004, **20**, 456-455.

[4] Kluge K., Lakshmanan S., A deformable-template approach to lane detection, *Intelligent Vehicles '95 Symposium*, Detroit, MI, USA, 25 September 1995.

[5] Bertozzi M., Broggi A., A parallel real-time stereo vision system for generic obstacle and lane detection, *IEEE Transaction on Image Processing,* 1995, **7**, 62-81.

[6] Pomerleau D. A., Progress in neural network-based vision for autonomous robot driving, *Intelligent Vehicles '92 Symposium*, Detroit, MI, USA, 01 July 1992.

[7] Kluge K., Extracting road curvature and orientation from image edge points without perceptual grouping into features, *Intelligent Vehicles '94 Symposium*, Paris, France, 24-26 October 1994.

[8] Pomerleau D., Jochem T., Rapidly adapting machine vision for autonomated vehicle steering, *IEEE Expert,* 1996, **11**, 19–27.

[9] Han J., Kim D., Lee M., Enhanced road boundary and obstacle detection using a downward-looking lidar sensor, *IEEE Transactions on Vehicular Technology*, 2012, **61**, 971-986.

[10] Wenger J., Automotive radar-status and perspectives, *IEEE Compound Semiconductor Integrated Circuit Symposium,* Palm Springs, CL, USA, 1-2 November 2005.

[11] Wenger J., Short range radar-being on the market, *37th Europ. Microw. Conf.*, Munich, Germany, 10-12 October 2007.

[12] Freundt D., Lucas B., Long range radar sensor for high-volume driver assistance systems market, *SAE Technical Paper,* DOI:10.4271/2008-01-0921.

[13] Lakshmanan S., Grimmer D., A deformable template approach to detecting straight edges in radar images, *IEEE Trans. Pattern Anal. Mach. Intell.,* 1996, **18**, 438–443.

[14] Wende Z., Lidar-based road and road-edge detection, *IEEE Intelligent Vehicles Symposium*, San Diego, CA, USA, 29 June 2010

[15] Kang Y., Roh C., Suh S., A lidar-based decision-making method for road boundary detection using multiple kalman filters, *IEEE Transactions on Industrial Electronics,* 2012, **59**, 4360-4369.

[16] Peterson K., Ziglar J., Rybski P. E., Fast feature detection and stochastic parameter estimation of road shape using multiple lidar*, IEEE/RSJ International Conference,* Nice, France, 22-26 September 2008.

[17] Chui C., *An introduction to wavelets*, 1th ed., Academic Press*,* San Diego, 1992.

[18] Haiping L., Hyongsuk K., Chun-shin L., Leon O. C., Road boundary detection based on the dynamic programming and the randomized hough tansform*, International Symposium on Information Technology Convergence,* Joenju, South Korea, 23-24 November 2007.

[19] Topi M., The local binary pattern approach to texture analysis extensions and applications, Academic Dissertation, University of Oulu, Infotech Oulu and Department of Electrical and Information Engineering, Oulu, 2003.

[20] Blank C., Trassoudaine L., Le Guilloux Y., Track to track fusion method applied to road obstacle detection, *IEEE International Conference on Information Fusion*, Stockholm, Sweden, 1 july 2004.

[21] http://gazebosim.org/wiki/Main_Page (Ziyaret tarihi: 20 Ekim 2013).

[22] http://www.coppeliarobotics.com/index.html (Ziyaret tarihi: 20 Ekim 2013).

[23] Grewal M., Weill L., Andrews A., *Global positioning systems, inertial navigation, and integration*, 2nd ed., John Wiley & Sons*,* Inc., New York, 2007.

[24] Borges G. A., Line extraction in 2D range images for mobile robotics, *Journal of Intelligent and Robotic Systems,* 2004, **40**, 267–297.

[25] Cremean L., Murray R., Model-based estimation of off-highway road geometry using single-axis ladar and ınertial sensing, *IEEE Conf. Robot*. Orlando, FL USA, 15-19 May 2006.

[26] Petrovskaya A., Thrun S., Model based vehicle tracking for autonomous driving in rural environments, *Autonomous Robots Journal*, 2009, **26**, 123-139.

[27] Urmson C., Anhalt J., Autonomous driving in rural environments: Boss and the rural challenge, *J. Field Robot.,* 2008, **25**, 425-466.

[28] Lee Y., Lim C., Cho J. H., Chung S. W. K., Map construction for autonomous mobile robots using a data association filter, *Advanced Robotics*, 2009, **23**, 185-201.

[29] Matthies L., Xiong Y., Hogg R., Zhu D., Rankin A., Kennedy B., Hebert M., Maclachlan R., Won C., Frost T., Sukhatme G., McHenry M., Goldberg S., A portable, autonomous, urban reconnaissance robot, *Robotics and Autonomous Systems*, 2002, **40**, 163-172.

[30] Moghadam P., Wijesoma W., Feng, D., Improving Path Planning and Mapping based on stereo vision and lidar, *10th Intl. Conf. on Control, Automation, Robotics and Vision*, Hanoi, Vietnam, 17-20 December 2008.

[31] Elfes A., Sonar-based real-world mapping and navigation, *IEEE Journal of Robotics and Automation*, 1987, **3**, 249-265.

[32] Chatila R., Laumond J. P., Position referencing and consistent world modeling for mobile robots, *IEEE International Conference on Robotics and Automation*, St. Louis, MO, USA , 15 March 1985.

EKLER

EK-A

Geliştirilen Algoritmanın Bir Girdisi Olan Ham Lidar Verisinin Yapısı

Uygulamanın temel girdisi olan ham lidar verisi aşağıdaki gibidir. Bu veride ki değerlerin anlamları aşağıdaki gibidir. Ham lidar verisi hexadecimal olarak gelir. Bu veri lidarın konfigürasyonuna göre 10 milisaniyede bir lidar sensör TCP/IP, CANBUS, Seri Port arayüzlerinden herhangi biriyle gönderebilir. Bu çalışmada TCP/IP üzerinden haberleşme sağlanmıştır. Ham lidar verisinden iki anlamlı veri arasına boşluk bırakılmıştır. Aşağıdaki tabloda bu verinin ne anlama geldiği gösterilmektedir.

Header	STX	4 byte
Komut turu	sSN	3 byte
Komut	LMDscandata	11 byte
Versiyon	0000	2 byte
Cihaz No	0001	2 byte
Seri No	00C8D186	4 byte
Status	00 00	1 byte 1 byte (00 00 ok) (00 01 error) (00 02 kirlilik uyarısı) (00 04 kirlilik hatası)
Paket Sayısı	3089	2 byte
Tarama Sayısı	3095	2 byte
Cihazın Çalıştığı süre	82804790	4 byte (milisaniye)
Taramanın Gönderim Zamanı	82809174	4 byte (milisaniye)
Dijital Inputların Durumu	00 00	1 byte 1 byte (00 00 all low) (00 03 all high)
Rezerve	003F	2 byte
Dijital Çıkışların Durumu	00 00	1 byte 1 byte (00 00 all low)

		(00 07 all high)
Tarama Frekansı	00001388	4 byte (50Hz)
Ölçüm Frekansı	00000168	4 byte (360)
Kodlayıcı	0000	2 byte (Eğer sıfırsa Kodayıcı pozisyonu ve Kodlayıcı Hızı Değerleri yoktur)
Kodlayıcı Pozisyonu	-	2 byte
Kodlayıcı Hızı	-	2 byte
16 Bit Kanal Miktarı	01	2 byte
İçerik	DIST1	5 byte (İlk çarpışın ışınsal değeri)
Ölçüm Faktörü	40000000	4 byte
Ölçüm Faktörü Offset	00000000	4 byte
Başlangıç Açısı	FFFF3CB0	4 byte (1/10.000 ölçek -50.000 +1850.000)
Çözünürlük	1388	2 byte (0.5Derece) (1/10.000 ölçek 1667 10.000)
Data Miktarı	17D	2 byte (381)
Data	Data Miktarı Kadar	Her bir ölçüm 2 byte
Konum	0000	2 byte (no postion data)
İsim Durumu	0000	2 byte (Eğer 0 sa isim yok.)
İsim boy	-	1 byte
İsim	-	2 byte
Yorum Durumu	-	2 byte
Zaman Durumu	-	2 byte
Yil	-	2 byte
Ay	-	1 byte
Gün	-	1 byte
Saat	-	1 byte
Dakika	-	1 byte

Saniye	-	1 byte
Milisaniye	-	4 byte
Olay bilgisi	-	2 byte (0 ise olay yok)

Ham lidar verisi:

sRA LMDscandata 1 1 C9B8D1 0 0 EEDC EEDF 98F2509F 98F2AA10 0 0 7 0 0 1388 168 0 1 DIST1 3F800000 00000000 FFF92230 1388 21D 74 69 67 63 5C 62 62 60 5C 5E 60 61 64 62 61 57 59 63 66 64 6C 6D 60 6A 71 70 63 74 69 77 75 7F 81 7B 7F 79 81 86 84 88 8D 8C 93 96 96 92 9A 9E 9D AA A6 AD B2 B8 BB BB C0 C0 DC 1096 1105 1180 1222 12E9 13A6 14B4 1579 1641 165E 1652 1646 1632 1634 162F 1632 1625 1628 1610 1618 1613 160C 1605 15F4 1600 15F3 15F4 1602 15F0 15F9 15FC 15FD 15F2 1602 15F9 15F6 15F9 1601 160A 160A 1611 161A 1617 1614 1626 162E 1636 163A 1610 1622 16B5 1676 167A 16CA 168B 16B2 16EC 16AD 16E3 170B 16D2 171C 1731 1705 1757 1741 1732 1770 170C 1718 1754 1762 176A 176E 1762 1702 169B 163F 15E8 15A7 1554 14E5 14D2 13B8 1332 12D2 128A 1237 11EE 3 3 10DB 10CE 3 3 0 0 3 3 8B5 8B1 8B4 8B3 8B4 8BE 8B9 8AD 8A9 8B9 8BA 8C2 8C1 8BC 8BD 8C2 8D9 8CE 8E1 8D4 8E6 8E2 8EE 8EB 3 958 3 B4C B87 B76 B5A B35 B2C B0F B0A AF5 ACC ABC ABB AE9 AFA AE6 ADB 72A 6DD AA0 AAB A8F A84 A70 A6E A63 A51 A45 A3F A2F A1E A1B 9F3 95A 9FA 9F4 9E1 9D7 992 826 816 870 9AC 9AA 982 8D8 8CF 940 97F 97B 97B 96C 966 956 95C 95E 95C 94C 946 94A 93F 93F 932 93A 933 930 92C 927 92B 920 92A 91E 915 91E 915 913 91E 916 90F 90F 918 911 905 905 90D 90B 919 917 911 910 917 91E 91C 911 913 918 923 927 91E 92E 92C 92F 925 92D 931 936 93C 943 94C 951 956 959 958 964 965 96B 97A 989 987 990 99F 99F 9B8 971 967 973 9D6 97F 8BB 811 7E9 7FD 89C 9CC 9B0 9AB 9AE 9AC 993 988 98F 9A5 995 6CD 64D 905 941 92A 8FF 8E5 8C1 8A5 895 872 86D 876 882 895 89C 8A9 8B6 933 9EF 8FC 8DA 8DD 8F7 90D 92A 93F 958 968 984 99B 9C3 9E4 9FE A26 A40 A6B A87 AB0 AD9 AF4 B20 B4D B6B BA3 C60 F6A FA1 FCF 1007 1038 1060 109B 10D3 3 5B4 5A4 5A7 59E 5A6 597 57C 574 578 563 55F 566 561 541 55D 550 541 53A 52C 539 525 525 528 51D 514 51A 505 4FF 4F8 4FC 4F3 4F2 4F3 4F1 4EB 4F6 4E9 4E4 4D2 4DC 4DB 4D3 4D0 4C2 4E0 4D6 4C7 4CA 4CC 4BE 4C3 4BF 4BD 4C3 4B9 4BF 4B3 4B1 4AA 4C8 4BD 4C8 4AE 4B5 4B7 4AD 4BA 4B7 4A9 4C1 4B8 4B2 4AF 4B0 4BA 4BE 4C2 4C6 4C1 4B3 4B4 4BF 4CA 4C7 4CB 4C8 4D7 4D1 4D4 4CA 4C3 4D4 4C4 4B7 49C 416 34B 2B6 20C 194 141 C2 C1 C8 BE B6 B4 AF 9D A0 95 99 91 8B 95 7C 87 80 82 6E 6A 7D 60 6F 75 68 59 64 66 5E 55 56 59 5C 62 57 5F 52 45 57 50 52 44 48 4B 4D 50 4D 48 50 43 49 38 3F 41 39 48 51 4D 0 0 0 0 0 0

EK-B

Yol Sınırları Tespiti İçin Geliştirilen Sınıflara Ait Program Kodları

Bu ekte, yol sınırlarının tespit edilmesine ait Java dilinde geliştirilen ugulamada kullanılan önemli sınıfların kodları sunulmuştur.

Eşik değeri sınıfı:

```
package com.etp.algorithm;

public class Threshold {
// line segment extract algoritim
public static double TH_MAX_HEIGHT = 60;// mm bir tarama noktasının yerden
olabilecegi maximum yukseklik mm cinsinden
public static double TH_MAX_POINT_COUNT = 2;// adet bir line segment en az kac
tarama noktasindan olusur

// breakpoint detection algoritim threaholds
public static double TH_GAUSS_ERROR = 60; // milimetre 0.09 metre
public static double TH_POINT_RANGE = 15; // 16 - 8 derece cinsinden iki tarama
noktası arasındaki max uzaklık.
public static double TH_TWO_POINT_TANJANT = 2; // iki tarama noktasi arasinda
olmasi gereken tanjant max degeri

// line segment siniflandirma algoritmasi
public static double TH_CRITERION_I_ROLL = Math.tan(Math.toRadians(6));//derece
cinsinden
public static double TH_CRITERION_I_PITCH = Math.tan(Math.toRadians(3));//derece
// combination kriterleri
eklenecek.....................................................
public static double TH_COMBINATION_CRITERION_ROLL =
Math.tan(Math.toRadians(1));
public static double TH_COMBINATION_CRITERION_PITCH =
Math.tan(Math.toRadians(0.5));
public static double TH_CRITERION_II_ROLL = Math.tan(Math.toRadians(4));
public static double TH_CRITERION_II_PITCH = Math.tan(Math.toRadians(2));

public static double TH_TRACK_PROBABILITY = 0.9;
//.............................
public static double TH_ROAD_WIDTH = 900; // mm 1400 olmali cinsinden
public static double TH_ROAD_HEIGT = 30; // mm cinsinden 5 cm = 50 mm olmali

}
```

Yol sınırı tesipiti algoritmasının implemente sınıf:

```
package com.etp.algorithm;

import java.util.ArrayList;
import java.util.List;
```

```
import org.apache.log4j.Logger;
import com.etp.domain.lidar.LazerBeam;
import com.etp.domain.lidar.LineSegment;
import com.etp.domain.lidar.Scan;

public class ObstacleAlgorithm {

static Logger logger = Logger.getLogger(ObstacleAlgorithm.class);

public static boolean checkBreakPoint(LazerBeam t, LazerBeam t1){ // kutupsaldan
kartezyene cevirme ve krırılım noktası tespiti.

boolean isPoint = false;

double tan = Math.abs(1 / Math.tan((t1.x - t.x) / (t1.y - t.y))); // ardisil iki nokta arasi
tanjant degeri
if (tan < Threshold.TH_TWO_POINT_TANJANT ) { // eger tanjant degeri ve uzaklik
degeri threshold degerini asiyorsa kirilim noktasi
isPoint = true;
}

return isPoint;
}

/**
 * Bir birinden uzakliklari th degerinden buyuk olan noktalar kirilim noktasidir.
 * gelen t t1 noktalari bir kirim noktasi midir?
 * @param t
 * @param t1
 */
public static boolean checkAdaptiveBreakPoint(LazerBeam t, LazerBeam t1){ //
kutupsaldan kartezyene cevirme ve krırılım noktası tespiti.

boolean result = false;

double resulotion = t.scan.lidar.resolution; // conurluk 1;
double rangeTd = Math.toDegrees(Threshold.TH_POINT_RANGE); // max olmasi
gereken mesafa treasholdu;8 derece
double maxRange = (t1.range * (Math.sin(resulotion) / Math.sin(rangeTd - resulotion)))
+ Threshold.TH_GAUSS_ERROR;

if (Math.abs(t.range - t1.range) > maxRange) {// && checkBreakPoint(t,t1) && t.z >
TH_MAX_HEIGHT
t.breakpoint = true;
t1.breakpoint = true;
result = true;
}

return result;
```

```
}


/**
 * bir taramdaki linesegmentleri cikarir.
 * @param scan
 */
public static void extractLineSegment(Scan scan){

scan.lineSegment = new ArrayList<LineSegment>();
int numberOfScanPoint = scan.beams.size();

if (numberOfScanPoint > 0) {// tarama sayisi sifirdan buyuk mu

int endPointIndex = 0; //
int startPointIndex = 0;

while(endPointIndex < numberOfScanPoint){ // son lazer isini taranan sayisindan buyuk

startPointIndex = endPointIndex;

++endPointIndex;
while((endPointIndex < numberOfScanPoint - 1) &&
!scan.beams.get(endPointIndex).breakpoint){// bir break pointi bulunan kadar endpointi
artir.
++endPointIndex;
if(endPointIndex == numberOfScanPoint - 1){
break;
}
}

while(endPointIndex - startPointIndex > Threshold.TH_MAX_POINT_COUNT){ //
line segmentte bulunan point sayisi 2 den buyuk mu

double dMax = 0;
int dMaxIndex = 0;
for(int i = startPointIndex+1; i<endPointIndex; i++ ){// line segmenttedki maksimum
yukseklikteki pointi bulalim. dMax degerine atayalim
if(scan.beams.get(i).z > dMax){
dMax = scan.beams.get(i).z;
dMaxIndex = i;
}
}

if(dMax > Threshold.TH_MAX_HEIGHT){
endPointIndex = dMaxIndex;
}else{
LineSegment lineSegment = new LineSegment();
lineSegment.startPoint = scan.beams.get(startPointIndex);
```

```
lineSegment.endPoint = scan.beams.get(endPointIndex);
lineSegment.scan = scan;
lineSegment.type = LineSegmentType.UNKNOWN;
calculateLineSegmentRollAndPitch(lineSegment);
calculateLineSegmentWidth(lineSegment);
scan.lineSegment.add(lineSegment);
break;
}

}

}
}
}

/**
 * bir line segmentin genisligi ve yuksekligi hesaplanir.
 * @param lineSegment
 */
public static void calculateLineSegmentWidth(LineSegment lineSegment){
double width = Math.sqrt(Math.pow((lineSegment.startPoint.x -
lineSegment.endPoint.x), 2) + Math.pow((lineSegment.startPoint.y -
lineSegment.endPoint.y), 2));
lineSegment.width = width;

double maxH = 0;
for(int i=lineSegment.startPoint.id; i<lineSegment.endPoint.id; i++){

if(lineSegment.scan.beams.get(i).z > maxH){
maxH = lineSegment.scan.beams.get(i).z;
}
}

lineSegment.height = maxH;
}

/**
 * bir line segment roll ve pitch acilari hesaplanir.
 * @param lineSegment
 */
public static void calculateLineSegmentRollAndPitch(LineSegment lineSegment){
double h = (lineSegment.startPoint.z - lineSegment.endPoint.z);
lineSegment.rollTan = Math.abs(h / lineSegment.startPoint.x);
lineSegment.pitchTan = Math.abs(h / lineSegment.startPoint.y);
}

/**
 * Bir taramanın line segmentlerinin siniflandirilmasi yapilir. 5 adimda gerceklestirilir.
 * @param scan
```

```
 */
public static void lineSegmentClassification(Scan scan){

lineSegmentRollAndPitchCriterionI(scan);
lineSegmentCombination(scan);
lineSegmentWidthCriterion(scan);
lineSegmentHeightCriterion(scan);

}

/**
 * bir line segmenti roll ve pitch aci II kriterine gore siniflandirir.
 * @param scan
 */
private static void lineSegmentRollAndPitchCriterionII(Scan scan) {
for(int i = 0; i<scan.lineSegment.size(); i++){
LineSegment segment = scan.lineSegment.get(i);
if(segment.type == LineSegmentType.UNKNOWN && (segment.rollTan >
Threshold.TH_CRITERION_II_ROLL || segment.pitchTan >
Threshold.TH_CRITERION_II_PITCH)){
segment.type = LineSegmentType.OBSTACLE;
//System.out.println(" RPC II " + segment.rollTan +"  " + TH_CRITERION_I_ROLL +
"  " + segment.pitchTan + "  " +TH_CRITERION_I_PITCH );
}
}
}

/**
 * bir line segmenti roll ve pitch aci I kriterine gore siniflandirir.
 * @param scan
 */
private static void lineSegmentRollAndPitchCriterionI(Scan scan) {
for(int i = 0; i<scan.lineSegment.size(); i++){
LineSegment segment = scan.lineSegment.get(i);

if(segment.rollTan > Threshold.TH_CRITERION_I_ROLL || segment.pitchTan >
Threshold.TH_CRITERION_I_PITCH){
segment.type = LineSegmentType.OBSTACLE;
//System.out.println(" RPC I " + segment.rollTan +"  " + TH_CRITERION_I_ROLL + "
" + segment.pitchTan + "  " +TH_CRITERION_I_PITCH );
}
}
}

/**
 * ardisilil line segmentlerin ozellikleri birbirine yakinsa birlestirir.
 * @param scan
 */
private static void lineSegmentCombination(Scan scan) {
```

```
List<LineSegment> combineSegments = new ArrayList<LineSegment>();
int index = 0;
while(index < scan.lineSegment.size() -1){

LineSegment segment1 = scan.lineSegment.get(index);
LineSegment segment2 = scan.lineSegment.get(index+1);

while(index < scan.lineSegment.size()-2 && checkTHTwoLineSegmet(segment1,
segment2)){
++index;
segment1 = combineTwoLineSegmet(segment1,segment2);
segment2 = scan.lineSegment.get(index + 1);
}

combineSegments.add(segment1);
++index;
}

scan.lineSegment = combineSegments;
}

/**
 * Bir line segmentin genisligini yol segmenti genisligi TH ile karsilastirarak
siniflandirir.
 * @param scan
 */
private static void lineSegmentWidthCriterion(Scan scan) {
for(int i = 0; i<scan.lineSegment.size(); i++){
LineSegment segment = scan.lineSegment.get(i);

if(segment.type == LineSegmentType.UNKNOWN && segment.width >
Threshold.TH_ROAD_WIDTH){
segment.type = LineSegmentType.ROAD;
}
else{
segment.type = LineSegmentType.OBSTACLE;
}
}
}

/**
 * Line segmentleri yukseklik kriterine gore siniflandirir.
 * @param scan
 */
private static void lineSegmentHeightCriterion(Scan scan) {
for(int i = 0; i<scan.lineSegment.size(); i++){
LineSegment segment = scan.lineSegment.get(i);
```

```
if(segment.type == LineSegmentType.UNKNOWN){
LineSegment nearestSegment = findNearestRoadSegment(segment,i);

if(nearestSegment != null && Math.abs(nearestSegment.height - segment.height) <
Threshold.TH_ROAD_HEIGT){
segment.type = LineSegmentType.ROAD2;
}else{
segment.type = LineSegmentType.OBSTACLE;
}
}
}
}

/**
 * bir line segmente en yakin road line segmenti dondurur.
 * @param segment
 * @param index
 * @return
 */
private static LineSegment findNearestRoadSegment(LineSegment segment, int index)
{

LineSegment nearest = null;

if(index > 0 && segment.scan.lineSegment.get(index-1).type ==
LineSegmentType.ROAD){
nearest = segment.scan.lineSegment.get(index-1);
}else if (index < segment.scan.lineSegment.size()-1 &&
segment.scan.lineSegment.get(index+1).type == LineSegmentType.ROAD) {
nearest = segment.scan.lineSegment.get(index+1);
}

return nearest;
}

/**
 * iki line segmentin roll ve pitch acilari karsilastirilir. Eger yakinsa true dondurur.
 * @param segment1
 * @param segment2
 * @return
 */
private static boolean checkTHTwoLineSegmet(LineSegment segment1, LineSegment
segment2) {
boolean result = false;
if(segment1.type == LineSegmentType.UNKNOWN && segment2.type ==
LineSegmentType.UNKNOWN
&& (Math.abs(segment1.rollTan - segment2.rollTan) <
Threshold.TH_COMBINATION_CRITERION_ROLL )
&& (Math.abs(segment1.pitchTan - segment2.pitchTan) <
```

```
Threshold.TH_COMBINATION_CRITERION_PITCH )){
result = true;

}
return result;
}

/**
 * İki line segmenti birlestirir geriye birlesmis halini dondurur.
 * @param segment1
 * @param segment2
 * @return
 */
private static LineSegment combineTwoLineSegmet(LineSegment segment1,
LineSegment segment2) {
segment1.endPoint = segment2.endPoint;
calculateLineSegmentWidth(segment1);
calculateLineSegmentRollAndPitch(segment1);
return segment1;
}
}
```

KİŞİSEL YAYIN VE ESERLER

[1] **Yalçın Ö.,** Sayar A., Arar Ö. F., Akpinar S., Kosunalp S. Approaches of Road Boundary and Obstacle Detection Using LIDAR, 1st IFAC Workshop on Advances in Control and Automation Theory for Transportation Applications (ACCATTA) Istanbul, Turkey, 16-17 September 2013.

[2] Arar Ö. F., Öncü A., Akpinar S., **Yalçın Ö.**, Mert Ü., Sadak M. S., Kosunalp S. A Flexible E-exam Framework and Air Traffic Controller Selection System, Global Journal on Technology, **3**, 2013.

ÖZGEÇMİŞ

1987 yılında Kahramanmaraş'ın Elbistan ilçesinde doğdu. İlköğrenim ve ortaokulu Elbistan'da, lise öğrenimini Ordu 'da tamamladı. 2005 yılında girdiği Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği bölümünden 2010 yılında mezun oldu. 2009-2011 yılları arasında çeşitli özel sektör firmalarında yazılım mühendisi olarak çalıştıktan sonra 2011 yılından itibaren TÜBİTAK BİLGEM bünyesindeki Bilişim Teknolojileri Enstitüsü'nde Araştırmacı olarak çalışmaktadır. 2011 yılında başladığı Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'ndaki Yüksek Lisans eğitimine devam etmektedir.