BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ TEZLİ YÜKSEK LİSANS PROGRAMI

OTONOM ARAÇLAR İÇİN ŞERİT TAKİBİ TEKNİĞİ GELİŞTİRİLMESİ

HAZIRLAYAN

SİNAN GÜVEN

YÜKSEK LİSANS TEZİ

ANKARA-2024

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ TEZLİ YÜKSEK LİSANS PROGRAMI

OTONOM ARAÇLAR İÇİN ŞERİT TAKİBİ TEKNİĞİ GELİŞTİRİLMESİ

HAZIRLAYAN

SİNAN GÜVEN

YÜKSEK LİSANS TEZİ

TEZ DANIŞMANI

DR. DENİZ KARAÇOR

ANKARA-2024

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Elektrik Elektronik Mühendisliği Anabilim Dalı Tezli Yüksek Lisans Programı çerçevesinde Sinan GÜVEN tarafından hazırlanan bu çalışma, aşağıdaki jüri tarafından Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Savunma Tarihi: 27/08/2024

Tez Adı: Otonom Araçlar İçin Şerit Takibi Tekniği.

Tez Jüri Üyeleri (Unvanı, Adı- Soyadı, Kurumu)	**İmza**
Doç. Dr. Hakkı Alparslan Ilgın, Ankara Üniversitesi	..……..
Prof. Dr. Sedat Nazlıbilek, Başkent Üniversitesi	..……..
Dr. Deniz Karaçor, Başkent Üniversitesi	..……..

ONAY
Prof. Dr. Dilek Çökeliler Serdaroğlu
Fen Bilimleri Enstitüsü Müdürü
Tarih: … / … / ……..

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
YÜKSEK LİSANS TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih: 10/09/2024

Öğrencinin Adı, Soyadı: Sinan GÜVEN
Öğrencinin Numarası:22110324
Anabilim Dalı: Elektrik-Elektronik Mühendisliği Anabilim Dalı
Programı: Elektrik-Elektronik Mühendisliği Tezli Yüksek Lisans Programı
Danışmanın Unvanı/Adı, Soyadı: Dr. Deniz KARAÇOR
Tez Başlığı: Otonom Araçlar İçin Şerit Takibi Tekniği

Yukarıda başlığı belirtilen Yüksek Lisans/Doktora tez çalışmamın; Giriş, Ana Bölümler ve Sonuç Bölümünden oluşan, toplam 107 sayfalık kısmına ilişkin, 25/08/2024 tarihinde şahsım/tez danışmanım tarafından Turnitin adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı %3'dür. Uygulanan filtrelemeler:

1. Kaynakça hariç
2. Alıntılar hariç
3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç

"Başkent Üniversitesi Enstitüleri Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Usul ve Esaslarını" inceledim ve bu uygulama esaslarında belirtilen azami benzerlik oranlarına tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Öğrenci İmzası:

ONAY
Öğrenci Danışmanı
Dr. Deniz KARAÇOR
Tarih: 10/09/2024

Bu tezi kendisiyle harcayabileceğim vakitlerimi bu tezi tamamlamak için harcamak zorunda olduğum kızım İpek ve çok değerli eşim Şenay'a ithaf ediyorum. Sabrınız için sonsuz teşekkürler. Geçip giden zamanları telafi edeceğiz…

Sinan GÜVEN

Ankara-2024

TEŞEKKÜR

Tez çalışmamın tamamlanması sürecinde desteklerini esirgemeyen herkese en içten teşekkürlerimi sunarım.

Öncelikle, bu tezin hazırlanmasında rehberlik eden, bilgeliği ve yönlendirmesiyle beni destekleyen danışmanım Dr. Deniz KARAÇOR'a teşekkür ederim. Sizlerin cesaretlendirici sözleri ve değerli geri bildirimleri, bu çalışmanın kalitesini artırmama büyük katkı sağladı.

Aynı zamanda, bu süreçte yanımda olan aileme teşekkürlerimi iletmek isterim. Eşim Şenay GÜVEN ve kızım İpek GÜVEN, benim bu yoğun dönemde daima yanımda oldunuz ve bana destek verdiniz. Sizin sevginiz ve anlayışınız olmasaydı, bu tez çalışmasını tamamlamak çok daha zor olurdu. Sizlerin desteğiyle, bu süreci daha kolay atlatmayı başardım.

Ayrıca, tezimde yer alan veri toplama sürecinde yardımcı olan ve bilgi paylaşımında bulunan meslektaşlarıma teşekkür ederim. Katkılarınız, bu çalışmanın kalitesini artırdı ve sonuçların daha anlamlı olmasını sağladı.

ÖZET

Sinan GÜVEN

OTONOM ARAÇLAR İÇİN ŞERİT TAKİBİ TEKNİĞİ

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalı

2024

Bu tezde, otonom araçlar için yol çizgisi takibi algoritmalarının geliştirilmesi ve performanslarının değerlendirilmesi amaçlanmıştır. Yol çizgisi takibi, otonom sürüş sistemlerinin güvenilirliği ve etkinliği açısından kritik bir bileşendir. Yol çizgisi takibi algoritmaları makine öğrenmesi tabanlı yöntemler ve geleneksel görüntü işleme teknikleri kullanılarak ikiye ayrılmaktadır. Makine öğrenmesi tabanlı yöntemler, belirli veri setleri üzerinde eğitilen modellerle çalışmakta olup, veri setinin dışındaki senaryolarda performans düşüşü yaşayabilmektedir. Bu yöntemlerin başarısı, kullanılan veri setinin çeşitliliği ve kapsamı ile doğrudan ilişkilidir [1]. Geleneksel yöntemler, veri setine bağımlı olmaksızın çalışmakta ve model eğitimi gerektirmediğinden hızlı bir şekilde uygulanabilmektedir. Bu yöntemler, yol çizgisi bilgilerini çıkartmak için kenar tespiti ve görüntü segmentasyonu gibi klasik görüntü işleme tekniklerinden yararlanmaktadır. Bu araştırmada yol çizgisi takibi için geleneksel yöntemlerin etkinliği değerlendirilmiştir. Yatay filtre komşuluk değerinin (τ) görüntünün parlaklığına göre adaptif hale getirilmesi gibi yenilikçi yaklaşımlar kullanılmıştır. Adaptif yöntem, değişen ışık koşullarında yol çizgilerinin doğru tespit edilmesini kolaylaştırmaktadır. Deneysel sonuçlar, adaptif τ seçiminin yol çizgisi tespiti üzerindeki olumlu etkilerini göstermiştir. Bu bulgular, otonom sürüş sistemlerinin güvenilirliğini ve etkinliğini artırmaya yönelik önemli katkılar sunmaktadır.

ANAHTAR KELİMELER: Görüntü İşleme, Yol Çizgisi Takibi, Otonom Sürüş, Elektronik Sensör, Yol Çizgisi

Danışman: Dr. Deniz Karaçor, Başkent Üniversitesi, Elektrik-Elektronik Mühendisliği Bölümü

ABSTRACT

Sinan GÜVEN

LANE DETECTION TECNIQUE FOR AUTONOMOUS CARS

Başkent University Institute of Science and Engineering

Department of Electrical and Electronics Engineering

2024

In this thesis, the aim is to develop and evaluate lane tracking algorithms for autonomous cars. Lane tracking is a critical component for the reliability and efficiency of autonomous driving systems. Lane tracking algorithms are divided into two categories: machine learning-based methods and traditional image processing techniques. Machine learning-based methods work with models trained on specific datasets and may experience performance drops in scenarios outside of these datasets. The success rates of these methods are directly related to the diversity and scope of the dataset used [1]. Traditional methods, on the other hand, operate independently of the dataset and can be implemented quickly as they do not require model training. These methods utilize classic image processing techniques such as edge detection and image segmentation to extract lane information. In this research, the effectiveness of traditional methods for lane tracking has been evaluated. Innovative approaches, such as making the horizontal filter neighbourhood value (τ) adaptive to the brightness of the image, have been employed. The adaptive method facilitates the accurate detection of lanes under varying lighting conditions. Experimental results have shown the positive effects of adaptive τ selection on lane detection. These findings offer significant contributions to enhancing the reliability and efficiency of autonomous driving systems.

KEYWORDS: Image Processing, Lane Detection, Autonomous drive, Electronic Sensor, Lane

Advisor: Dr. Deniz Karaçor, Başkent University, Department of Electrical and Electronics Engineering

İÇİNDEKİLER

TABLOLAR LİSTESİ

Sayfa

ŞEKİLLER LİSTESİ

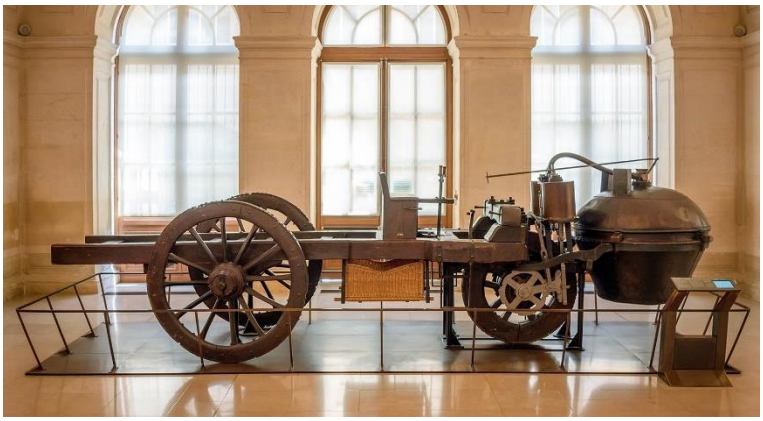
Sayfa

SİMGELER VE KISALTMALAR LİSTESİ

ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance Systems
AEB	Automatic Emergency Braking
b	Doğru dikey eksen kesişim noktası
BMP	Windows Bitmap
CUR	Cursor File
DMS	Driver Monitoring System
ESP	Electronic Stability Program
GIF	Graphics Interchange Format
GPS	Global Positioning System
HDF4	Hierarchical Data Format
ICO	Icon File
IMU	Inertial Measurement Unit
JPEG	Joint Photographic Experts Group
JPEG 2000	Joint Photographic Experts Group 2000
JSON	JavaScript Object Notation
L	Elde edilen bir doğrunun uzunluğu
LIDAR	Light Detection and Ranging
m	Doğru eğimi
MATLAB	Matrix Laboratory
PBM	Portable Bitmap
PCX	Windows Paintbrush
RGB	Red, Green, Blue
PGM	Portable Graymap
PNG	Portable Network Graphics
PPM	Portable Pixmap
RAS	Sun Raster
RGB	Red, Green, Blue
ROI	Region Of Interest
TIFF	Tagged Image File Format
X	Z skoru hesaplanacak veri
XWD	X Window Dump
Z	X noktası için Z-skoru
ρ	Doğrunun orijine olan en kısa mesafesidir (normal mesafesi).
θ	Normalin açısıdır (0 ile 180 derece arasında).
(x, y)	Görüntü uzayındaki bir noktanın koordinatlarıdır.
σ	Standart sapma
M(x,y)	Gauss filtre gradyan büyüklüğü
$\emptyset$(x,y)	Gauss filtre gradyan yönü
G_x	X yönündeki gradyan
G_y	Y yönündeki gradyan
$A(\rho,\theta)$	Hough dönüşümü akümülatör matrisi
μ	Verilerin ortalaması
τ	Yatay filtre komşuluk değeri

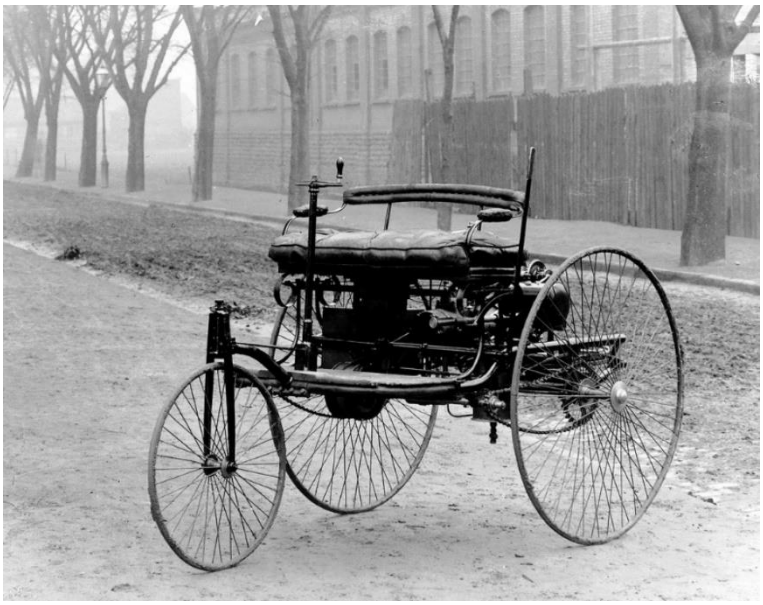
GİRİŞ

Tekerleğin icadı, medeniyetin ilerlemesinde önemli bir dönüm noktası olmuştur. Tekerlek sayesinde tarım, taşımacılık, ulaşım ve mimari alanlarda büyük gelişmeler kaydedilmiştir. Buharlı araçların Fransız mühendis Nicolas Joseph Cugnot tarafından 1769 yılında icat edilmesiyle ulaşım alanında yeni bir döneme girilmiştir [2].



Şekil 0.1 Nicolas Joseph Cugnot tarafından icat edilen buharlı araç [2]

Buharlı trenler, tonlarca yükü kolaylıkla taşıma kapasiteleri sayesinde ticaretin gelişmesine büyük katkı sağlamışlardır. Alman mühendis Karl Benz'in ilk benzinli aracı icat etmesiyle birlikte, araçlar daha hızlı ve daha güçlü hale gelmiştir [3].



Şekil 0.2 Karl Benz tarafından icat edilen ilk araç [4]

Buharlı araçların saatte 3,5 kilometrelik hızından günümüz araçlarının saatte yüzlerce kilometrelik hızına ulaşması, araçlarda güvenlik ve konforun önemini büyük ölçüde artırmıştır.

Yüksek hızlardaki sürüş güvenliğini sağlamak için gelişmiş fren sistemleri, çarpışma önleme teknolojileri ve sürücü destek sistemleri gibi yenilikçi çözümler geliştirilmiştir.

Araçların hızlanması ve yaygınlaşması ile trafik karmaşası artmıştır. Trafik kazalarının en önemli nedenlerinden biri, sürücü hatalarıdır. Yol çizgisi ihlalleri, hatalı yol çizgisi değiştirme ve yorgunluk gibi faktörler, trafik kazalarına sebep olmaktadır. Edward N. Hines, 1911 yılında yol çizgilerini ilk kez uygulayarak trafik güvenliğinde önemli bir adım atmıştır [5]. Bu çizgiler, trafik akışını bir nebze düzenlemiş olsa da kazaları tamamen engelleyememiştir.

Teknolojinin gelişmesiyle birlikte, araçlarda sürüş güvenliği amaçlı daha fazla elektronik sensör kullanılmaya başlanmıştır. Bu sensörler, araç arızalarını önceden tespit ederek sürüş güvenliğini artırmaktadır. Sürücünün farkındalığını artırmak ve sürüş güvenliğini sağlamak için kullanılan başlıca sensörler şunlardır:

- **RADAR:** Araçların etrafındaki nesneleri tespit etmek, hızlarını ve mesafelerini ölçmek için kullanılır. Özellikle adaptif hız sabitleyici (ACC) ve otomatik acil frenleme (AEB) gibi sistemlerde kullanılır.
- **LIDAR:** Radar gibi nesneleri tespit etmek ve mesafe ölçmek için kullanılır, ancak daha yüksek çözünürlükte ve daha detaylı 3D haritalar oluşturma yeteneğine sahiptir. Otonom araçlarda yaygın olarak kullanılır.
- **Kameralar:** Trafik işaretlerini tanıma, yol çizgisi takip sistemi, sürücü yorgunluk tespiti ve yayaların algılanması gibi birçok farklı amaç için kullanılır.
- **Ultrasonik Sensörler:** Aracın yakın çevresindeki nesneleri tespit etmek için kullanılır. Özellikle park sensörleri ve kör nokta uyarı sistemlerinde kullanılır.
- **Sürücü İzleme Sistemleri (DMS):** Sürücünün dikkatini ve yorgunluğunu izlemek için kameralar ve yapay zekâ kullanır. Sürücünün dikkatsiz veya uykulu olduğunu tespit ederse uyarı verir.
- **Atalet Ölçüm Birimleri (IMU):** Aracın hızlanma, dönüş ve eğim gibi hareketlerini ölçer. Elektronik Denge Programı (ESP) ve yokuş kalkış desteği gibi sistemlerde kullanılır.
- **Küresel Konumlandırma Sistemi (GPS):** Aracın konumunu belirlemek için kullanılır. Navigasyon sistemleri ve bazı gelişmiş sürücü destek sistemleri (ADAS) için gereklidir.

Araçlarda kullanılan sensörlerin sayısı artarken, paralelden görüntüleme sistemlerinde de gelişmeler yaşanmıştır. Görüntüleme sistemlerinde yaşanan gelişmeler, görüntü işleme algoritmalarının daha hızlı ve başarılı hale gelmesini sağlamıştır. Görüntü işleme teknikleri,

otonom araçlarda yol çizgisi takibi gibi kritik uygulamalarda kullanılmaktadır. Otonom araçlarda kullanılan diğer sensörler çevresel durum farkındalığı sağlarken, aracın yol çizgisi içinde kalması için görüntü işleme algoritmalarına ihtiyaç duyulmaktadır. Bu bağlamda kameralar, otonom araçlar için en önemli sensörlerden biri haline gelmiştir. Örneğin, Tesla araçlarında 360° görüş alanı sağlamak için 8 adet kamera bulunmaktadır [6].

Şekil 0.3 Tesla araçlarda kamera yerleşimi [6]

Geleneksel görüntü işleme yöntemleri, yol çizgisi takibi gibi uygulamalarda büyük önem taşımaktadır. Verimli ve hesaplaması nispeten basit algoritmalar kullanarak gerçek zamanlı uygulamalarda yüksek performans sağlarlar. Geleneksel yöntemler, özellikle veri setine bağımlı olmayan yapılarıyla makine öğrenmesi tabanlı yöntemlere göre avantajlıdır [7]. Makine öğrenmesi yöntemleri belirli veri setleri üzerinde eğitilirken, geleneksel yöntemler veri setlerine bağımlı olmaksızın çalışabilmektedir. Bu durum, geleneksel yöntemlerin esnekliğini ve farklı koşullarda kullanılabilirliğini artırır.

Nieto vd., [8] tarafından geliştirilen, yatay filtreleme temelli geleneksel bir görüntü işleme yöntemi, yol çizgisi tespitinde hızlı ve başarılı sonuçlar vermektedir. Ancak, özellikle değişken ışık koşullarında, yatay filtre parametresinin (τ) seçimi, yol çizgisi tespit performansı üzerinde kritik bir etkiye sahiptir. Bu çalışmada, farklı ışık senaryolarında yol çizgisi tespit başarımını artırmak amacıyla, adaptif yatay filtre parametresi (τ) seçimine dayanan bir yol çizgisi takip algoritması önerilmektedir. Önerilen algoritmanın performansı, sabit τ seçimi yöntemiyle TuSimple veri seti [9] kullanılarak karşılaştırılmıştır.

VERİ SETİ

1.1 Veri Seti Hakkında

Bu araştırmada yol çizgisi bilgisi elde etmek amacıyla, akademik çalışmalarda yaygın olarak kullanılan ve uygunluğu tespit edilen TuSimple veri seti tercih edilmiştir. 2020-2023 yılları arasında TuSimple veri seti kullanılarak toplam 60 adet makale yayınlanmıştır. Veri setinin yıllara göre kullanım dağılımı aşağıdaki şekildedir [9]:

- 2020: 10 makale
- 2021: 10 makale
- 2022: 17 makale
- 2023: 12 makale

TuSimple veri seti, ABD otoyollarında farklı koşullarda çekilmiş 6.408 yol görüntüsünden oluşmaktadır. Veri setindeki görüntüler JPEG formatında ve ".jpg" uzantılıdır. Her bir görüntü 720x1080 piksel çözünürlüğe sahip olup, 0-255 aralığında değerler alan Kırmızı (R), Yeşil (G) ve Mavi (B) renk bileşenlerinden oluşmaktadır.

Bu çalışmada kullanılan ve tezde belirtilen tüm yol görüntüleri TuSimple veri setinden elde edilmiştir [9].

1.2 Veri Setinin Yönetimi

TuSimple veri seti, Manideep Sridhara tarafından yönetilmektedir.

1.3 Veri Setinin Özellikleri

Veri setindeki görüntüler, çeşitli koşulları yansıtarak aşağıdaki temel özelliklere sahiptir:

- Hava Koşulları: İyi ve orta derece hava koşulları
- Gün ve Gece Koşulları: Gündüz ve gece zamanlarında çekilmiş görüntüler
- Yol çizgisi Sayısı: İki veya daha fazla şeride sahip otoyollar
- Trafik Koşulları: Farklı trafik yoğunlukları ve durumları

1.4 Veri Setinin Yapısı

TuSimple veri setindeki görüntüler aşağıdaki dağılıma sahiptir:

- Eğitim Görüntüleri: 3.626 adet
- Doğrulama Görüntüleri: 358 adet
- Test Görüntüleri: 2.782 adet

Veri seti, her biri ardışık olarak çekilmiş 20 görüntü içeren klasörlerden oluşmaktadır. Klasörlerdeki son görüntü, yol çizgisi bilgilerini içeren etiket dosyasına sahip olan görüntüdür.

1.5 Veri Setinden Örnek Görüntüler

Veri seti farklı ışık koşullarında çekilmiş yol görüntülerinden oluşmaktadır. Şekil 0.1'de düşük ışık koşullarında çekilmiş yol görüntüsü yer almaktadır. Şekil 0.2'de yüksek ışık koşullarında çekilmiş yol görüntüsü yer almaktadır.

Şekil 0.1 Düşük ışık koşullarında çekilmiş bir yol görüntüsü

Şekil 2.2 Yüksek ışık koşullarında çekilmiş bir yol görüntüsü

1.6 Veri Setindeki Yol Çizgisi Koordinat Bilgileri

TuSimple veri seti, hesaplanan yol çizgisi bilgileriyle karşılaştırma yapılabilmesi için "label_data_(tarih).json" formatında etiket dosyaları içerir. Bu JSON dosyaları, her bir görüntü için aşağıdaki yol çizgisi bilgilerini barındırır:

- Görüntü dosyasının dosya yolu
- Görüntüdeki yol çizgilerinin dikey koordinatları
- Görüntüdeki yol çizgilerinin yatay koordinatları

1.6.1 Dosya yolu

Etiket dosyalarında, yol çizgisi bilgileriyle etiketlenen görüntülerin dosya yolları *"raw_file"* etiketi altında yer alır. Bu dosya yolları, ilgili görüntülerin veri seti içerisindeki göreceli dosya yolunu belirtir. Örnek bir dosya yolu verisi aşağıdaki gibidir:

"raw_file": "clips/0601/1494453497604532231/20.jpg"

1.6.2 Görüntüdeki yol çizgilerinin dikey koordinat bilgileri

Etiket dosyalarında, *h_samples"* başlığı altında bir görüntüdeki yol çizgilerinin dikey konumları yer alır. Bu veriler, 10 piksel aralıklarla örneklenmiş yol çizgilerinin dikey eksen boyunca başlangıç ve bitiş indeksleri arasındaki koordinatları temsil eder.

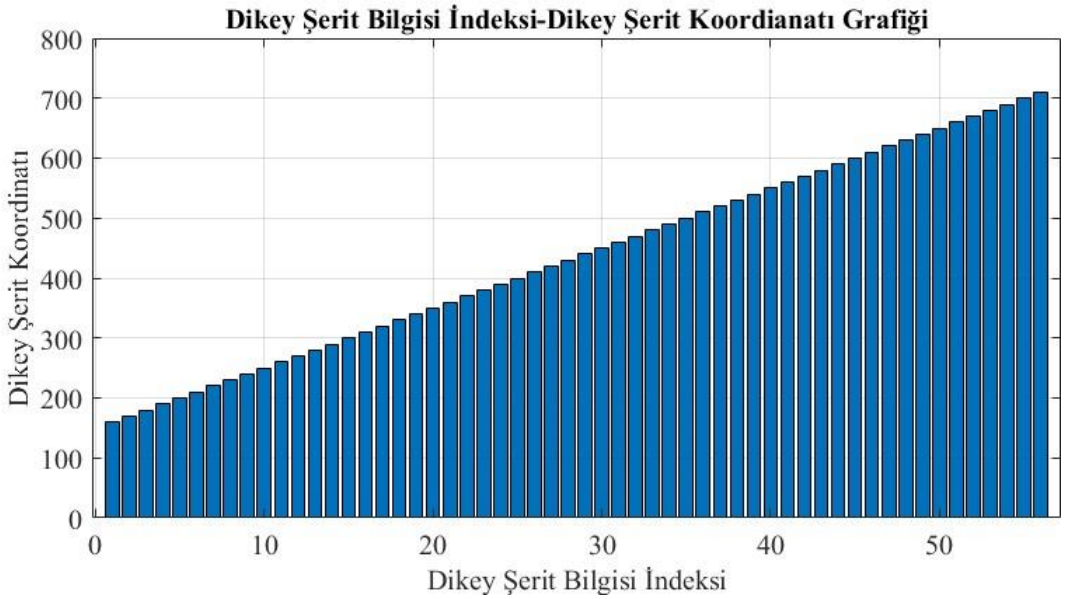
Dikey koordinat bilgileri, bir görüntüdeki yol çizgilerinin dikey eksende hangi yüksekliklerde (piksel cinsinden) bulunduğunu gösterir. Her bir "h_samples" değeri, o yükseklikteki yol çizgilerinin yatay konumlarının ("lanes" başlığı altında) eşleştirildiği referans noktalarını belirtir.

Örnek bir h_samples içeriği:

"h_samples": [160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330, 340, 350, 360, 370, 380, 390, 400, 410, 420, 430, 440, 450, 460, 470, 480, 490, 500, 510, 520, 530, 540, 550, 560, 570, 580, 590, 600, 610, 620, 630, 640, 650, 660, 670, 680, 690, 700, 710]

Bu örnekte, yol çizgilerinin dikey konumları 160 pikselden başlayarak 710 piksele kadar 10'ar piksel aralıklarla örneklenmiştir.

Verilen örnek için h_samples verilerinin MATLAB ortamında *plot* fonksiyonu ile çizdirildiğinde Şekil 0.2'deki gibi bir görselleştirme elde edilir. Bu görselleştirme, yol çizgilerinin görüntü boyunca dikey olarak nasıl konumlandığını gösterir.



Şekil 0.2 Yol çizgisi bilgisi dikey koordinat grafiği

1.6.3 Görüntüdeki yol çizgilerinin yatay koordinat bilgileri

Etiket dosyalarında, bir görüntüdeki yol çizgilerinin yatay konumları "lanes" başlığı altında listelenir. Her yol çizgisinin yatay konum bilgisi, "h_samples" koordinatlarına karşılık gelen bir dizi değer içerir.

Yatay koordinat bilgileri listesindeki her bir eleman, belirli bir yol çizgisinin farklı yüksekliklerdeki (h_samples) yatay konumlarını gösterir. Eğer belirli bir yükseklikte yol çizgisi tespit edilemezse, bu durum "-2" değeri ile belirtilir.

Örnek bir "lanes" içeriğinin 1 numaralı indeksi:

"lanes(1)": [-2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, 646, 635, 624, 613, 603, 592, 581, 570, 560, 549, 538, 527, 517, 506, 495, 484, 474, 463, 452, 441, 431, 420, 409, 398, 388, 377, 366, 355, 345, 334, 323, 312, 302, 291, 280, 269, 259, 248, 237, 227, 216, 205, 194, 184, 173]

Bu örnekte, her bir yol çizgisi için 56 adet yatay konum bilgisi bulunmaktadır (her biri bir "h_samples" değerine karşılık gelir). Verideki "-2" değerleri, o yükseklikte yol çizgisinin tespit edilemediğini gösterir. Diğer değerler ise yol çizgisinin o yükseklikteki yatay piksel konumunu temsil eder.

Örneğin, birinci yol çizgisinin 12. "h_samples" değerine (yani 280 piksel yüksekliğe) karşılık gelen yatay konumu 646 pikseldir.

METOT

Bu araştırmada, otonom araçlar için yol çizgisi takibi algoritmalarının geliştirilmesi ve performanslarının değerlendirilmesi amacıyla geleneksel görüntü işleme teknikleri kullanılmıştır. Yol çizgisi takibi, otonom araçların yolda güvenli bir şekilde seyredebilmesi için kritik bir bileşendir. Bu nedenle, yol çizgisi tespiti ve takibi için kullanılan yöntemlerin doğruluğu ve etkinliği, otonom araçların güvenliği ve yol tutuşu açısından büyük önem taşımaktadır.

Bu araştırma, otonom araçların güvenli bir şekilde seyredebilmesi için hayati önem taşıyan yol çizgisi takibi algoritmalarının geliştirilmesine odaklanmaktadır. Geleneksel görüntü işleme teknikleri kullanılarak yol çizgisi tespiti ve takibi için yeni yöntemler geliştirilmiş ve bu yöntemlerin performansı değerlendirilmiştir. Yol çizgisi takibi algoritmalarının doğruluğu ve etkinliği, otonom araçların güvenliği ve yol tutuşu üzerinde doğrudan etkili olduğundan, bu araştırmanın sonuçları otonom araç teknolojilerinin gelişimine önemli katkılar sağlayabilir.

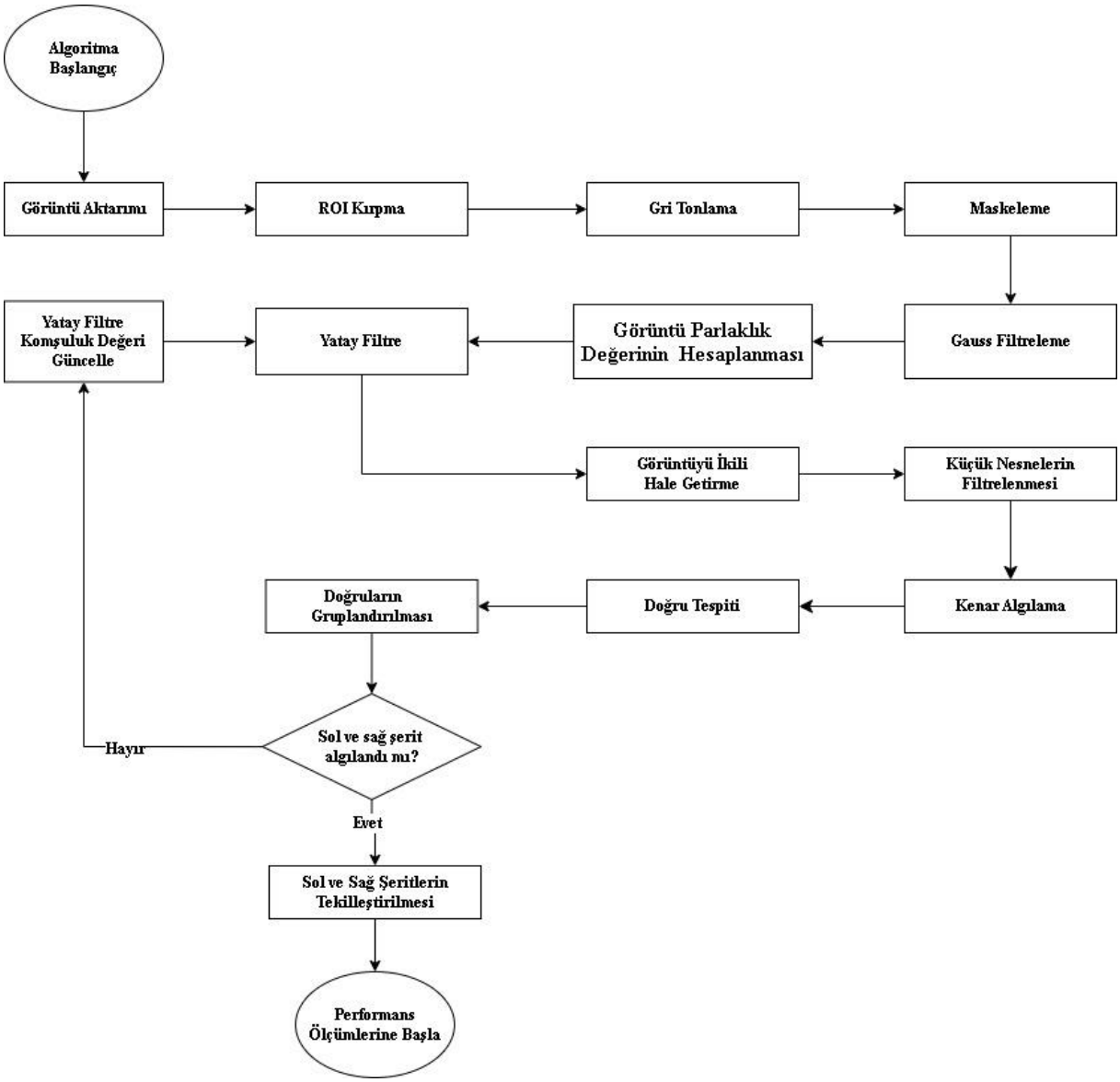
Çalışmada, yol çizgisi takibi algoritmasının performansını artırmak amacıyla adaptif τ yöntemi kullanılmıştır. Yatay filtre komşuluk değeri, görüntüdeki ortalama parlaklık seviyeleriyle dinamik olarak ilişkilendirilerek adaptif bir şekilde ayarlanmıştır. Bu yaklaşım, değişken ışık koşullarında daha sağlam ve doğru yol çizgisi tespiti sonuçları elde edilmesine olanak sağlamıştır.

Algoritmik hesaplamalar, tez çalışmaları boyunca Mathworks tarafından geliştirilen MATLAB[1] yazılımı kullanılarak gerçekleştirilmiştir. Bu yazılım, algoritmaların geliştirilmesi, test edilmesi ve doğrulanması süreçlerinde etkin bir araç olarak kullanılmıştır.

Çalışmada yer alan teknik çizimler ise açık kaynaklı bir diyagram yazılımı olan Draw.io[2] ile oluşturulmuştur.

[1] MATLAB: Mathworks firmasının tescilli bir ürünüdür.
[2] Draw.io: Diagrams.net tarafından geliştirilen açık kaynaklı bir diyagram yazılımıdır.

Bu araştırmada Şekil 0.1'deki algoritma akışı kullanılmıştır.



Şekil 0.1 Çalışmada kullanılan algoritma akışı

Araştırmada kullanılan algoritma akışı aşağıdaki adımlardan oluşmaktadır:

1. **Görüntü aktarımı:** Görüntü, işleme için MATLAB ortamına aktarılır.
2. **ROI kırpma:** Görüntü, önceden belirlenmiş ilgi alanı (ROI) parametrelerine göre kırpılır, böylece sadece ilgili bölge üzerinde işlem yapılır.
3. **Gri tonlama:** Renkli görüntü, gri tonlama yöntemiyle gri skalaya dönüştürülür.
4. **Maskeleme:** Görüntü üzerinde maskeleme işlemi uygulanır, böylece sadece yol çizgilerinin bulunduğu bölgeler üzerinde işlem yapılır.
5. **Gauss filtreleme:** Maskelenmiş görüntüye Gauss filtresi uygulanarak görüntüdeki gürültü azaltılır.
6. **Görüntü ortalama parlaklık değerinin hesaplanması:** Gauss filtresi uygulanmış görüntünün ortalama parlaklık değeri hesaplanır.
7. **Yatay filtre:** Görüntünün ortalama parlaklık değerine bağlı olarak adaptif bir yöntemle yatay filtre komşuluk değeri belirlenir ve yatay filtre uygulanır.
8. **Görüntüyü ikili hale getirme:** Yatay filtre uygulanmış görüntü, Otsu eşikleme yöntemi kullanılarak siyah-beyaz (ikilik) formata dönüştürülür.
9. **Küçük nesnelerin filtrelenmesi:** İkilik görüntü üzerindeki küçük nesneler (gürültüler) filtrelenir.
10. **Kenar algılama:** Görüntüdeki kenarlar, Canny kenar algılama yöntemi kullanılarak tespit edilir.
11. **Doğru tespiti:** Görüntüdeki doğrular, Hough dönüşümü yöntemi ile belirlenir.
12. **Doğruların gruplandırılması:** Elde edilen doğrular, sağ ve sol yol çizgisi adayları olarak gruplandırılır.
13. **Yatay filtre komşuluk değerinin güncellenmesi:** Eğer sağ ve sol yol çizgisi adayları bulunamamışsa, yatay filtre komşuluk değeri güncellenir ve algoritma 7. adımdan itibaren tekrar çalıştırılır.
14. **Sol ve sağ yol çizgilerinin tekilleştirilmesi:** Eğer sağ ve sol yol çizgisi adayları bulunmuşsa, bu adaylar tekilleştirilerek nihai sol ve sağ yol çizgisi bilgileri elde edilir.

1.7 Görüntü Aktarımı

MATLAB ortamında algoritma geliştirme ve test süreçlerinin gerçekleştirilebilmesi için, veri setindeki görüntülerin MATLAB ortamına aktarılması gerekmektedir. Bu çalışmada, veri setindeki görüntülerin MATLAB ortamına entegrasyonu, MATLAB'ın yerleşik "*imread"* fonksiyonu [10] aracılığıyla gerçekleştirilmiştir.

Bu fonksiyon, farklı formatlardaki (JPEG, PNG, vb.) görüntüleri okuyarak MATLAB'da işlenebilir matrislere dönüştürür. Bu fonksiyon, dosya yolunu girdi parametresi olarak alır ve belirtilen dosyada bulunan görüntüyü bir MATLAB matrisine dönüştürür. Elde edilen bu matris, görüntünün piksel değerlerini ve boyut bilgisini içerir. Fonksiyon, yaygın olarak kullanılan farklı görüntü formatlarının yanı sıra, matris ve vektör gibi çeşitli veri yapılarını da destekler.

MATLAB'ın *imread* fonksiyonunun desteklediği görüntü uzantıları aşağıda listelenmiştir [10]:

- BMP
- JPEG
- PNG
- CUR
- JPEG
- PPM
- GIF
- PBM
- RAS
- HDF4
- PCX
- TIFF
- ICO
- PGM
- XWD

Veri seti JPEG[3] formatında ".jpg" uzantılı görüntülerden oluşturulmaktadır. MATLAB yazılımı JPEG formatında bir görüntüyü çalışma ortamına aktarırken Tablo 0.1'deki gibi çıktılar üretir [10].

Tablo 0.1 MATLAB JPEG görüntü çıktı türü tablosu [10]

Piksel bileşeni başına bit sayısı	**Kayıplı Sıkıştırma**	**Kayıpsız Sıkıştırma**	**Çıktı Türü**
1	Destekleniyor	Destekleniyor	mantıksal
2 bit-8 bit	Destekleniyor	Destekleniyor	uint8 veya int8
9 bit-16 bit	Destekleniyor	Destekleniyor	uint16 veya int16

Veri setindeki görüntüler, her biri 0-255 piksel değerine sahip 8 bitten oluşan RGB (Red, Green, Blue) renk bileşenlerine sahiptir. MATLAB'ın *imread* fonksiyonu bu tür görüntüleri uint8 veri türünde içeri aktarır [10].

Bu çalışmada kullanılan görüntüler, MATLAB çalışma alanına 720x1280x3 boyutunda, uint8 formatında aktarılmıştır. Şekil 0.2'de MATLAB ortamına aktarılan bir örnek görüntü gösterilmektedir.

Şekil 0.2 MATLAB ortamına aktarılan örnek bir görüntü

[3] JPEG: Joint Photographic Experts Group tarafından geliştirilen ve Standartlaştırılan bir görüntü sıkıştırma standardıdır.

1.8 ROI Kırpma

Yol çizgisi takibi algoritmasının başarısı, doğru ROI (Region of Interest) seçilmesinden büyük ölçüde etkilenir. Doğru ROI alanının seçilmesi, algoritmanın iki temel yönden daha başarılı olmasını sağlar [11]:

- **İşlem Yükünün Azaltılması:** Görüntünün çözünürlüğünü düşürerek algoritmanın işlem yükünü önemli ölçüde azaltır. Bu, gerçek zamanlı uygulamalar için kritik bir faktördür.
- **Yanlış Yol Çizgisi Adaylarının Elenmesi:** Algoritmadaki yanlış yol çizgisi adaylarının elenmesine yardımcı olur. Bu, yol çizgisi takibinin doğruluğunu ve güvenilirliğini artırır.

Literatürdeki çalışmalar, dikdörtgen şeklinde ROI seçiminin diğer şekillere göre daha yüksek başarı sağladığını göstermektedir [12]. Bu nedenle, bu tezde sunulan yol çizgisi algılama algoritmasında da dikdörtgen şeklinde ROI seçilmiştir.

Bu tez kapsamında seçilen ROI, yatay ekseni tamamen kapsayacak şekilde tasarlanmıştır. Bu tasarım tercihinin temel nedeni, yol çizgisi bilgisinin yatay eksende herhangi bir noktadan başlayabilme potansiyelidir. Örneğin, Şekil 0.3'teki görüntüde sağ yol çizgisi sağ alt köşeden başlamaktadır.

Şekil 0.3 ROI seçiminde incelenen veri seti görüntüsü (1)

Buna ek olarak, Şekil 0.4'teki görüntüde sol yol çizgisi sol alt köşenin bile solundan başlamaktadır.

Şekil 0.4 ROI seçiminde incelenen veri seti görüntüsü (2)

Şekil 0.3 ve Şekil 0.4'ten anlaşılacağı üzere, ROI alanı seçiminin yatay ekseni tamamen içermesi gereklidir. Bu yaklaşım, algoritmanın farklı yol koşullarına ve yol çizgisi başlangıç noktalarına daha iyi uyum sağlamasını sağlar.

Tezde sunulan ROI'nin dikey başlangıç noktası, görüntünün dikey ekseninin minimum değeri olarak belirlenmiştir. Bu yaklaşımın temel gerekçesi, yol çizgisi bilgisinin dikey eksende fotoğrafın çekim yerine ve açısına göre en yakın pikselden başlayabilmesidir. (Şekil 0.3)

Yol çizgisi takibi çalışmalarında, yol çizgisi bilgisi genellikle görüntünün dikey ekseninde belirli bir yüzdeye karşılık gelmektedir. Veri setindeki yol görüntülerindeki yol çizgileri çizildiği durumlarda, yol çizgilerinin ufuk çizgisine göre konumlarının, görüntünün dikey eksende 420 piksel değerinin içinde kaldığı gözlemlenmiştir. (Şekil 0.5)

Şekil 0.5 Örnek görüntüdeki yol çizgisi bilgisi

Yol görüntülerinde ROI'nin dikey olarak alabileceği değerler deneysel olarak incelendiğinde aşağıdaki sonuçlar elde edilmiştir:

- ROI'nin dikey değeri 420'den büyük seçildikçe dış etkenlerin arttığı fark edilmiştir. Bunun sebebi genelde diğer araçların genelde dikeyde 420 piksel-720 piksel aralığında olmasıdır.
- ROI'nin dikey değeri 420'den küçük seçildikçe yol çizgisi verilerinin kaybedilmeye başlandığı fark edilmiştir.

Yukarıda bahsedilen sebeplerden ötürü, bu araştırmada ROI için dikey bitiş noktası, görüntünün altından itibaren 420 piksel olarak belirlenmiştir. Böylece, 720x1280x3 boyutundaki orijinal görüntüden 420x1280x3 piksel boyutunda bir ROI elde edilmiştir. Şekil 3.6'da kullanılan görüntü boyutları ve bu görüntüye ait ROI alanı gösterilmiştir. Şekilde beyaz ile gösterilen alan kullanılan görüntüyü, kırmızı ile gösterilen alan ise kullanılan ROI alanını ifade etmektedir.

Şekil 0.6 ROI alanının görüntü üzerinde gösterimi

Şekil 3.6'daki görüntünün ROI alanına göre kırpılması sonucu elde edilen görüntü Şekil 0.7'de görülebilir.

Şekil 0.7 ROI alanında kırpılmış yol görüntüsü

Bu işlem sonucunda, orijinal görüntünün dikey boyutunun %41,67'si çıkarılarak işleme tabi tutulan veri miktarı azaltılmıştır. Bu azalma, algoritmanın hesaplama karmaşıklığını doğrudan etkileyerek algoritmanın çalışma zamanını önemli ölçüde azaltmıştır.

1.9 Gri Tonlama

Veri setini oluşturan görüntüler Kırmızı (*Red*, R), Yeşil (*Green*, G) ve Mavi (*Blue*, B) olmak üzere üç bileşenden oluşmaktadır.

Tezde gri tonlamalı görüntü elde edilmeden önce görüntüdeki kırmız, yeşil ve mavi renk kanalları MATLAB'ın im2double fonksiyonu kullanılarak normalize edilmiştir. Görüntüdeki renk bileşenleri uint8 türünde olduğundan dolayı im2double fonksiyonu görüntüdeki her bir renk bileşeninin piksel değerini 255'e böler. Bu işlemin sonucunda her bir renk bileşeni 0-1 aralığına normalize edilmiştir.

RGB formatındaki görüntülerin gri tonlamalı görüntülere dönüştürülmesi için MATLAB yazılım programının *"rgb2gray"* fonksiyonu kullanılmıştır. Bu fonksiyon, Eşitlik (3.1)'de verildiği gibi, üç bileşenin ağırlıklı toplamını oluşturarak gri tonlama değerlerinin elde edilmesini sağlamaktadır.

Gri tonlamaya sahip normalize bir görüntüde piksel değerleri 0 ve 1 arasında değişmektedir. Piksel 0 değeri siyah renge, 1 değeri ise beyaz renge karşılık gelmektedir.

$$g(x,y)=0{,}299\cdot R(x,y)+0{,}587\cdot G(x,y)+0{,}114\cdot B(x,y) \qquad (3{,}1)$$

Eşitlik (3.1)'de renk bileşenlerinin katsayılarının farklı olmasının nedeni, insan gözünün bu renklere hassasiyetinin farklı seviyede olmasıdır. İnsan gözü yeşil renge daha hassas olduğundan, G(x,y) bileşeninin katsayısı daha yüksektir.

Tezde sunulan yol çizgisi algılama algoritmasında renk bilgisi kullanılmamaktadır. Buna bağlı olarak işlem yükünün azaltılması ve dolayısıyla işlem hızının artırılabilmesi için görüntüdeki her bir pikselin üç bileşen yerine tek bir gri yoğunluk seviyesi ile temsil edilmesi tercih edilmiştir. Gri tonlamalı görüntünün elde edilmesi, algoritmanın sonraki adımları açısından bir ön işlem olarak da değerlendirilebilir.

Bu işlem sonucunda, 420x1280x3 boyutundaki RGB (Red, Green, Blue) renk uzayındaki ROI görüntüsü, 420x1280 piksel boyutunda gri tonlamalı normalize bir görüntüye dönüştürülmüştür. Bu dönüşüm, renk bilgisinin ihmal edilmesiyle hesaplama karmaşıklığını önemli ölçüde azaltırken, yol çizgisi algılama için gerekli olan kontrast ve yoğunluk bilgisini korur. Bu sayede, algoritmanın daha verimli çalışması ve gerçek zamanlı uygulamalarda daha hızlı sonuçlar üretmesi sağlanır.

Şekil 0.7'de ROI'ye göre kırpılan görüntünün gri tonlama görüntüye çevrilmiş hali Şekil 0.8'de gösterilmiştir.

Şekil 0.8 Gri tonlamalı yol görüntüsü

1.10 Maskeleme

Görüntü maskeleme, yol çizgisi tespit algoritmalarının doğruluğunu ve verimliliğini artırmak için yaygın olarak kullanılan bir yöntemdir. Maskeleme, görüntünün yalnızca ilgili alanını (ROI) seçerek gereksiz bilgileri filtreler ve hesaplama maliyetlerini düşürür. Bu, özellikle karmaşık yol sahnelerinde veya değişken ışık koşullarında yol çizgisi tespitinin doğruluğunu ve hızını artırır.

Maskeleme teknikleri, yol çizgisi tespiti algoritmalarının doğruluğunu artırarak daha güvenli sürüş deneyimi sağlar. HG Zhu [13], yoğun trafikte ve karmaşık şehir içi yol koşullarında maskeleme tekniklerinin yol çizgisi tespit algoritmalarının doğruluğunu önemli ölçüde artırdığını göstermiştir.

Bu araştırmada, maskeleme görüntüsü oluşturulurken yol çizgilerinde bulunan perspektif etkisi dikkate alınmıştır. Perspektif etkisi, yoldaki yol çizgisi işaretleri gibi paralel çizgilerin uzaklaştıkça birleşiyormuş gibi görünmesine neden olur. Bu, doğrusal perspektifin temel bir ilkesidir; nesneler gözlemciden uzaklaştıkça daha küçük görünür. Düz bir yolda, yol çizgisi işaretleri ufukta bir kaybolma noktasında birleşir ve bu da perspektif etkisini oluşturur [14].

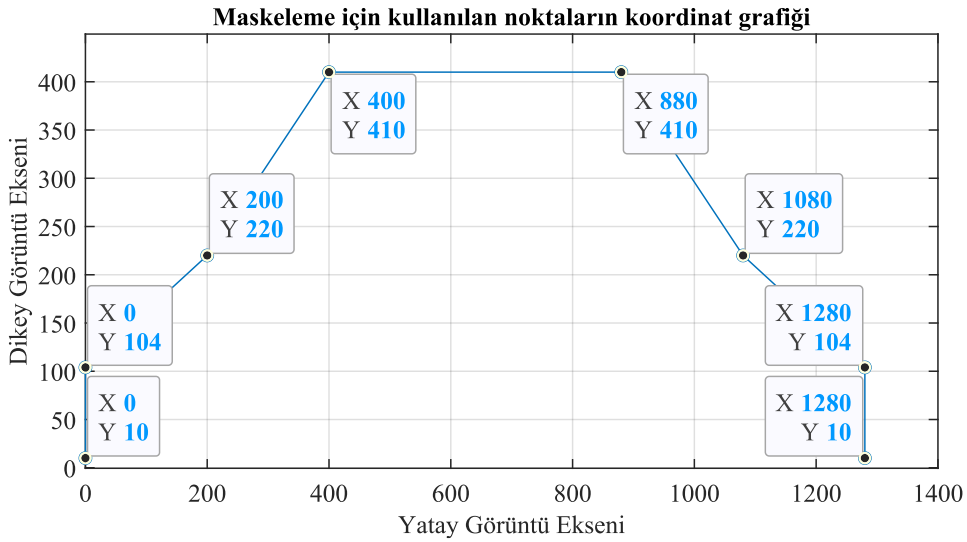
Perspektif etkisini hesaba katmak için, maskeleme görüntüsünde sol ve sağ yol çizgileri için ikişer adet doğru parçası kullanılmıştır. Bunun nedeni, yakındaki yol çizgilerinin daha az eğime sahip olması, uzaktaki yol çizgilerinin ise ufuk çizgisine yaklaştıkça daha fazla eğime sahip olmasıdır.

Maskeleme görüntüsünde ayrıca ROI alanının üst 10 ve alt 10 piksel alanı boş bırakılmıştır. Bu alanlarda daha fazla gürültü bulunabileceği göz önünde bulundurularak, bu bölgelerin maskeleme dışında tutulması algoritmanın performansını artırır. Görüntüde maskeleme yapabilmek için Tablo 0.2'deki yatay ve dikey koordinatlar kullanılmıştır. Tablo 0.2'de listelenen koordinatlar veri setindeki yol görüntülerinde sol ve sağ yol çizgilerinin bulunabildiği alanın deneysel olarak bulunması sonucunda elde edilmiştir.

Tablo 0.2 Maskeleme görüntüsünde kullanılan yatay ve dikey koordinatlar

Nokta	**Koordinat**
Nokta_1	(0,10)
Nokta_2	(0,104)
Nokta_3	(200,220)
Nokta_4	(400,410)
Nokta_5	(880,410)
Nokta_6	(1080,220)
Nokta_7	(1280,104)
Nokta_8	(1280,10)

Tablo 0.2'de kullanılan noktalardan elde edilen maskeleme alanının koordinat grafiği MATLAB ortamında çizdirildiğinde Şekil 0.9'un elde edildiği gözlemlenmiştir.



Şekil 0.9 Maskeleme için kullanılan noktaların koordinat grafiği

Tablo 0.2'de belirtilen noktalar kullanılarak oluşturulan maskeleme görüntüsü, veri setindeki tüm görüntülere uygulanmıştır. Maskeleme görüntüsü, ilgi alanı içindeki piksellerin 0, dışındaki piksellerin 1 olduğu ikili bir görüntüdür. Bu maskeleme görüntüsü, Şekil 0.10'da gösterilmiştir.

Şekil 0.10 Çalışmada kullanılan yol çizgisi maskeleme görüntüsü

Maskeleme işlemi, orijinal görüntü ile maskeleme görüntüsünün piksel bazında matris çarpımı ile gerçekleştirilmiştir. Maskeleme alanı içindeki pikseller 1 ile çarpılarak orijinal değerlerini korurken, maskeleme alanı dışındaki pikseller 0 ile çarpılarak sıfırlanmıştır. Bu sayede, sadece ilgi alanındaki bilgiler korunmuş, diğer bölgelerdeki gereksiz bilgiler ise elimine edilmiştir.

Şekil 0.8'de gösterilen gri tonlamalı görüntü Şekil 0.10'daki maskeleme görüntüsü ile maskelendiğinde Şekil 0.11'deki görüntü elde edilmiştir. Bu sonuç, sol ve sağ yol çizgileri dışındaki büyük bir alanın maskelenerek işleme dahil edilmediğini açıkça göstermektedir.

Şekil 0.11 Örnek görüntü maskeleme çıktısı

1.11 Gauss Filtreleme

Maskelenen görüntülerin ön işlem olarak gürültülerden arındırılması gerekmektedir. Bunun için görüntü işlemede yaygın olarak kullanılan Gauss filtre kullanılmıştır.

1.11.1 Gauss filtre hakkında

Gauss filtre, görüntü işlemede yaygın olarak kullanılan bir filtreleme yöntemidir. Bu filtre, görüntüdeki yüksek frekanslı bileşenleri azaltırken, düşük frekanslı bileşenleri korur. Gauss filtresi, adını ünlü matematikçi Carl Friedrich Gauss'dan alır ve normal dağılım eğrisi (Gauss eğrisi) ile ilişkilidir [7]. Özellikle gürültü azaltma ve kenar yumuşatma gibi işlemlerde oldukça etkilidir [15].

Yol çizgisi takibi uygulamalarında Gauss filtresi, görüntülerdeki istenmeyen gürültü ve kenar detaylarını azaltarak yol çizgilerinin daha net ve belirgin hale gelmesini sağlar [16]. Bu sayede, yol çizgilerini algılamak ve takip etmek için kullanılan algoritmalar daha doğru ve güvenilir sonuçlar verir. Gauss filtresinin hem görüntüdeki gürültüyü azaltarak hem de işlem verimliliğini yükselterek yol çizgisi takibi uygulamalarındaki başarıyı artırdığı gösterilmiştir [17].

Gauss filtrenin başlıca faydaları aşağıdaki gibidir:

- **Gürültü Azaltma:** Gerçek dünya görüntülerinde, kamera sensöründen kaynaklanan elektronik gürültü, ortam aydınlatmasındaki dalgalanmalar ve nesnelerin hareketi gibi faktörlerden dolayı gürültü oluşabilir. Bu gürültü, yol çizgilerinin kenarlarını bulanıklaştırarak ve şekillerini bozarak yol çizgisi takibi algoritmalarının performansını düşürebilir. Gauss filtre, bu tür gürültüleri etkili bir şekilde azaltarak yol çizgilerinin daha net ve belirgin hale gelmesini sağlar.
- **Kenar Yumuşatma:** Gauss filtresi, görüntülerdeki keskin kenarları yumuşatarak geçişleri daha pürüzsüz hale getirir. Bu işlem, yol çizgilerinin kenarlarının daha iyi tanımlanmasına ve yol çizgisi takibi algoritmalarının daha doğru çalışmasına yardımcı olur. Özellikle karmaşık görüntü ortamlarında, Gauss filtresi, yol çizgilerinin arka plandan ayrıştırılmasını kolaylaştırarak algılama işlemini optimize eder.
- **Ayarlanabilir Parametre:** Gauss filtresi, farklı parametreler kullanılarak farklı gürültü seviyelerine ve kenar yumuşatma ihtiyaçlarına göre uyarlanabilir. Bu sayede, farklı görüntü türleri ve aydınlatma koşulları için en uygun filtreleme işlemi gerçekleştirilebilir.

- **Hesaplama Verimliliği**: Gauss filtresi, nispeten basit bir matematiksel yapıya sahip olmasından dolayı hesaplama açısından oldukça verimlidir. Bu sayede, gerçek zamanlı yol çizgisi takibi gibi gecikme toleransı düşük olan uygulamalarda kullanılabilir.

Gauss filtre, bir görüntü üzerinde bir Gauss çekirdeği ile konvolüsyon işlemi gerçekleştirilmesini içermektedir. Bu çekirdek, Gauss fonksiyonunun ayrık bir yaklaşımı olup, genellikle simetrik ve pozitif değerler içerir [7]. Çekirdeğin boyutu ve standart sapması, filtreleme işleminin ne kadar güçlü olacağını belirler. Bu parametrelerin doğru bir şekilde seçilmesi, filtreleme işleminin etkinliğini ve sonuç görüntüsünün kalitesini doğrudan etkiler [15]. Özellikle büyük boyutlu çekirdekler, daha güçlü bir bulanıklık etkisi yaratırken, küçük boyutlu çekirdekler daha az bulanıklık sağlar [16]. Standart sapmanın uygun seçimi, kenar detaylarının korunması ve gürültünün azaltılması arasında bir denge kurulmasını sağlar [17].

Gauss filtrenin matematiksel ifadesi, normal dağılımın iki boyutlu bir versiyonu olarak tanımlanabilir. Gauss filtrenin çekirdeği G(x,y), Eşitlik (3.2) ile ifade edilmektedir:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-(\frac{x^2+y^2}{2\sigma^2})} \tag{3.2}$$

Eşitlik (3.2)'de x ve y, merkezi piksele göre konumları temsil ederken, σ ise standart sapmayı ifade eder ve filtre çekirdeğinin genişliğini belirler. Standart sapma değeri arttıkça, filtreleme etkisi de artar ve görüntü daha fazla yumuşatılır.

1.11.2 Gauss filtrenin uygulanması

Gauss filtre uygulanması amacıyla MATLAB'ın *"imgaussfilt"* fonksiyonu kullanılmıştır. Bu fonksiyon, görüntü işleme alanında sıkça kullanılan bir araç olup, özellikle gürültü azaltma ve kenar yumuşatma işlemlerinde etkili sonuçlar verir. Fonksiyonun temel kullanımı Eşitlik (3.3)'teki gibidir:

```
J = imgaussfilt(I, sigma,FilterSize,Padding);
```
(3.3)

MATLAB'ın *imgaussfilt* fonksiyonu Tablo 0.3'deki girdi ve çıktı parametrelerine sahiptir. Tablo 0.3'de girdi parametrelerinin çalışmada kullanılan değerleri listelenmiştir. Fonksiyonda kullanılan parametreler değerler deneysel olarak elde edilmiştir ve veri setindeki tüm görüntülerde bu değerler sabit olarak kullanılmıştır.

Tablo 0.3 MATLAB'ın *imgaussfilt* fonksiyonu girdi ve çıktı parametreleri

Parametre	**Parametre açıklaması**	**Türü**	**Çalışmada kullanılan değer**
I	Girdi görüntüsü. Bu görüntü, bir gri tonlamalı veya renkli olabilir.	Girdi	Maskeleme görüntüsü çıktısı
sigma	Gauss filtresinin standart sapmasıdır. Bu parametre, Gauss dağılımının yayılma derecesini belirler ve bulanıklaştırmanın derecesini kontrol eder.	Girdi	1
FilterSize	Filtrenin boyutunu belirler. Varsayılan olarak, `FilterSize` değeri `2*ceil(2*sigma) +1` olarak belirlenir.	Girdi	5
Padding	Kenarların nasıl doldurulacağını belirler. Aşağıdaki değerleri alabilir: • replicate: Kenar pikselleri tekrarlanır. • symmetric: Kenar pikselleri simetrik olarak yansıtılır. • circular: Kenar pikselleri dairesel olarak tekrarlanır. • Sabit bir değer: Kenarlar bu sabit değerle doldurulur.	Girdi	replicate
J	Çıktı görüntüsü. Bu, girdi görüntüsü `I`'ye Gauss filtresi uygulandıktan sonra elde edilen görüntüdür.	Çıktı	-

Şekil 0.11'da maskelenmiş görüntünün Gauss Filtre çıktısı Şekil 0.12'deki gösterilmiştir.

Şekil 0.12 Örnek görüntü Gauss filtre çıktısı

1.12 Görüntü Ortalama Parlaklık Değerinin Hesaplanması

Yatay filtreleme aşamasında, görüntünün ortalama parlaklık değeri referans olarak kullanılmaktadır. Bu amaç doğrultusunda, görüntünün ortalama parlaklık değerinin hesaplanması gerekmektedir.

1.12.1 Görüntü ortalama parlaklık etkileri hakkında

Geleneksel yol çizgisi algılama yöntemleri, yol ve yol çizgisi arasındaki kontrast farkını kullanarak yol çizgisi bilgilerini çıkartmaktadır. Bu yöntemlerde, yol çizgileri genellikle daha parlak, yol detayları ise daha koyu tonda görünmektedir. Bu kontrast farkı, kenar algılama algoritmalarının yol çizgilerini doğru bir şekilde tespit etme başarısını artırmaktadır. Ancak, çevresel koşullar bu süreci olumsuz etkileyebilmektedir. Özellikle gölgeler, yolda oluşan parlamalar ve diğer çevresel faktörler, kontrast farkının azalmasına veya değişmesine neden olarak yol çizgisi algılama performansını düşürebilir. Çevresel koşulların getirdiği zorluklar şu şekillerde özetlenebilir:

- **Gölge Etkileri:** Gölge alanları, yol çizgilerinin ve yolun kontrastını değiştirebilir, bu da yol çizgilerinin doğru bir şekilde algılanmasını zorlaştırabilir. Örneğin, bir ağaç veya bir bina tarafından yol üzerine düşen gölge, yol çizgisi algılama algoritmalarının hata yapmasına neden olabilir.
- **Yol Parlamaları:** Özellikle güneşli havalarda, yol yüzeyindeki yansımalar ve parlamalar, yol çizgilerinin algılanmasını güçleştirebilir. Parlamalar, yol çizgisi ve yol arasındaki kontrastı bozar ve algoritmaların yanılmasına yol açar.
- **Çeşitli Hava Koşulları:** Yağmur, kar ve sis gibi hava koşulları, yol yüzeyinin ve yol çizgilerinin görünürlüğünü azaltabilir. Bu durumlarda, kontrast farkı daha da belirsiz hale gelir ve yol çizgisi algılama performansı düşer [18].

Güneşin yoğun olduğu durumlarda yol yüzeyi yüksek ortalama parlaklık değerine sahip olmaktadır (Şekil 0.13). Bu durum yollarda yol çizgisi dışı alanların da parlak görünmesine neden olmaktadır. Bu alanlardaki beyaz noktaların ortalama parlaklık değerleri daha da yüksek olmaktadır. Ayrıca, araçlardan ve yoldan yansımalar da oluşabilmektedir.

Şekil 0.13 Güneş ışınlarının fazla olduğu örnek bir yol görüntüsü

Güneşin etkisinin az olduğu durumlarda yol görüntüleri daha az aydınlıktır. Bu durum, yollardaki yol çizgilerinin daha koyu görünmesine neden olur. Şekil 0.14'te, yol çizgilerinin normalde belirgin bir ortalama parlaklıkta olması beklenirken, daha koyu göründükleri gözlemlenmektedir. Bu durum, yol çizgilerinin algılanmasını zorlaştırarak hata oranlarını artırır.

Şekil 0.14 Işık oranı az ortamdaki yol çizgisi görüntüsü

1.12.2 Görüntü ortalama parlaklık hesaplanması

Gauss filtrenin uygulanması adımında elde edilen görüntünün ortalama parlaklık seviyesinin hesaplanabilmesi için MATLAB'ın *mean* fonksiyonu kullanılmıştır.

MATLAB'ın *mean* fonksiyonu, verilerin aritmetik ortalamalarını ölçmek için kullanılan temel bir araçtır. Aritmetik ortalama, veri noktalarının toplamının veri noktası sayısına bölünmesiyle elde edilir [19]. Mean fonksiyonu Eşitlik (3.4)'te gösterildiği şekilde kullanılır:

$$M = \text{mean}(I); \tag{3.4}$$

MATLAB'ın *mean* fonksiyonunun temel girdi ve çıktı parametreleri Tablo 0.4'daki gibidir. Çalışmada *mean* fonksiyonunda I görüntüsü I(:) şeklinde tek boyutlu diziye dönüştürülerek ortalama işlemine tabi tutulmuştur.

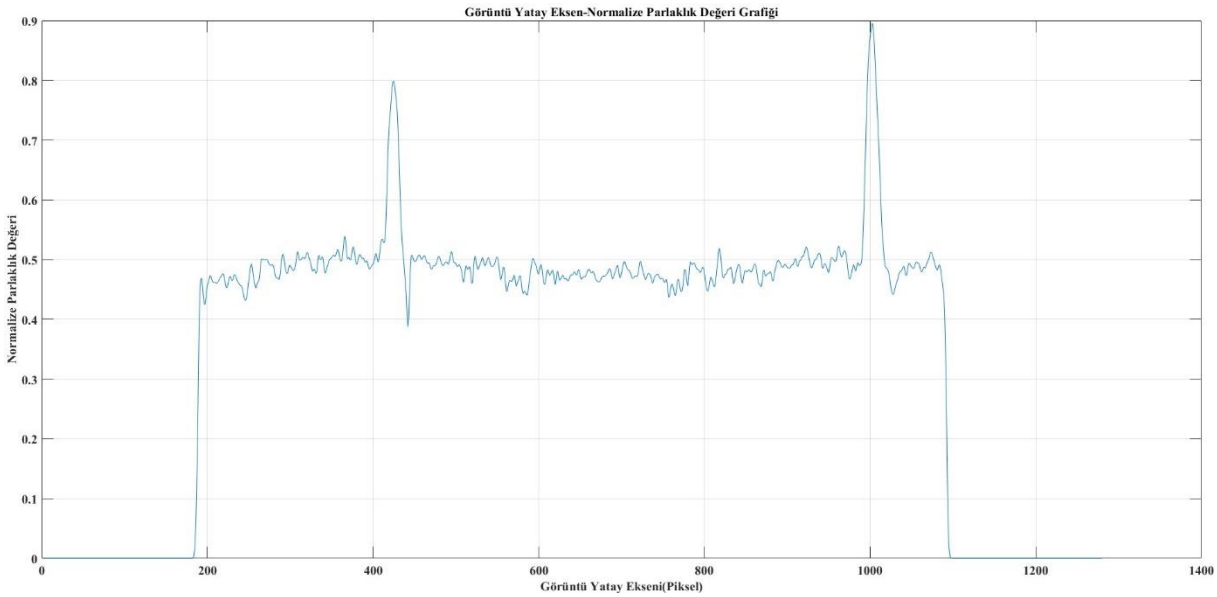
Tablo 0.4 Mean fonksiyonu girdi ve çıktı parametreleri

Parametre	**Parametre Tanımı**	**Türü**	**Çalışmada kullanılan değer**
I	Girdi görüntüsü.	Girdi	Gauss filtre çıktısı
M	Ortalama parlaklık değeri	Çıktı	-

Bu çalışmada, Gauss filtresi çıktısı, “*mean”* fonksiyonunun girdi parametresi olarak kullanılmıştır. Mean fonksiyonundan elde edilen ortalama parlaklık değeri, "Yatay filtre komşuluk değerinin ayarlanması" aşamasında kullanılmak üzere hesaplanmıştır. Şekil 0.12’de gösterilen görüntü için hesaplanan ortalama parlaklık değeri 0.3305 olarak belirlenmiştir.

1.13 Yatay Filtre

Yol çizgisi işaretleri, yatay eksende çevresindeki bölgelerden daha yüksek yoğunluk değerleri gösterme eğilimindedir. Bu durum, yol çizgisi çevresindeki işaretlerinin yüzeylerle olan kontrastından kaynaklanmaktadır. Örneğin, Şekil 0.15'te görüldüğü üzere, yol çizgisi işaretleri ile çevresindeki bölgeler arasındaki yoğunluk farkı belirgin bir şekilde gözlemlenmektedir.

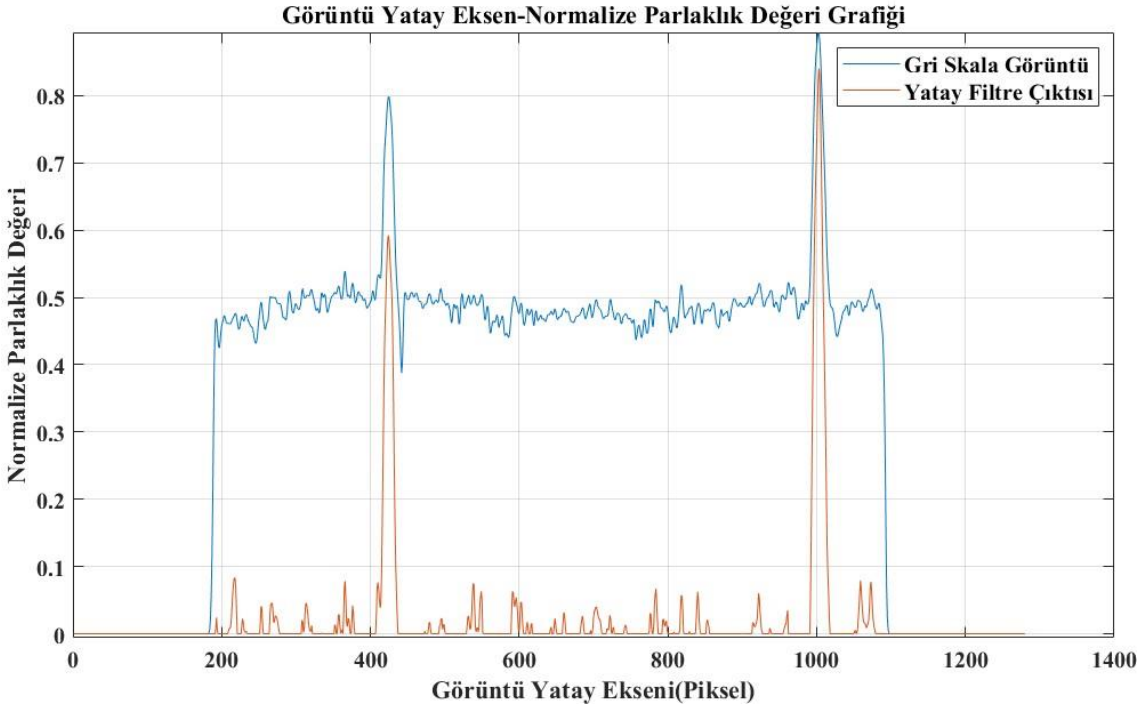


Şekil 0.15 Yol çizgili yatay eksenin normalize ortalama parlaklık değeri grafiği

Yatay eksende çalıştırılacak bir filtre, yol çizgisi olmayan alanların filtrelenerek yol çizgisi bilgilerinin ön plana çıkarılmasını amaçlamaktadır. Bu amaç doğrultusunda, bir dedektör her görüntü satırını bağımsız olarak piksel yoğunluk değerlerine göre filtreler. Nieto vd., [8] tarafından tasarlanan yatay filtre aşağıdaki hızlı ve başarılı sonuçlar vermektedir. Bahsedilen yatay filtre Eşitlik (3.5)’te verilmiştir.

$$N(i,j) = 2G(i,j) - (G(i-\tau,j) + G(i+\tau,j)) - |G(i-\tau,j) + G(i+\tau,j)| \quad (3.5)$$

Filtre, G ile temsil edilen gauss filtre çıktı görüntüsünü her bir dikey koordinat (j) için, her bir yatay koordinatta (i) yatay filtre komşuluk değerine (τ) göre pozitif ve negatif yönde filtreleyerek N görüntüsü elde eder. Filtre, aynı satırda τ mesafesinde sol ve sağ komşularından daha yüksek yoğunluk değerlerine sahip piksellere güçlü bir yanıt verir. Bu yaklaşım, yol çizgisi işaretlemelerinin daha belirgin hale gelmesini sağlamaktadır. Şekil 0.15'deki çizdirilen yatay eksen verisine τ=29 yatay filtre genişliğine sahip filtre uygulandığında Şekil 0.16'deki grafik elde edilmiştir. Şekilden de görüleceği üzere zirve noktaları (yol çizgisi bilgileri) dışında kalan noktalar baskılanmıştır.



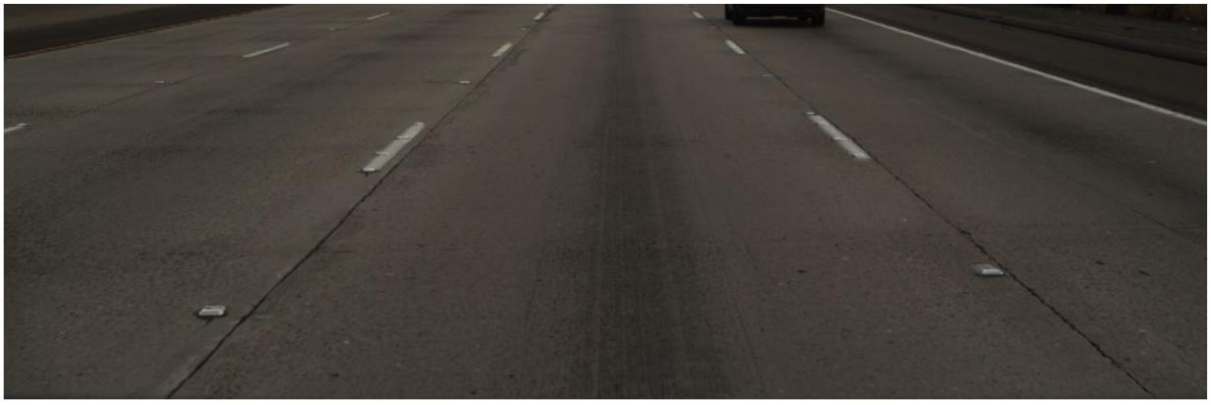
Şekil 0.16 Yol çizgili herhangi bir yatay eksenin ortalama parlaklık değeri grafiği

1.13.1 Yatay filtre komşuluk değerinin ayarlanması

Yatay filtre kapsamında MATLAB ortamında Eşitlik (3.5) gerçeklenmiştir. Yatay filtrede τ değeri parametresi bulunmaktadır. Literatürdeki çalışmalarda bu değer genellikle sabit bir değer seçilmiştir. Ancak görüntülerin içerdiği yapısal farklılıklar, optimal yatay filtreleme performansı için adaptif bir şekilde τ değerinin uyarlanmasını gerektirebilir.

Bu araştırmada adaptif bir yöntem belirlenebilmesi için koyu ve parlak görüntülerin farklı τ değerlerinde yatay filtre çıktıları incelenmiştir.

Koyu görüntü olarak Şekil 0.17'de görülen yol görüntüsü kullanılmıştır. Kullanılan görüntünün ortalama parlaklık değeri 0.1760 olarak hesaplanmıştır.



Şekil 0.17 Yatay filtre τ değeri testlerinde kullanılan koyu görüntü

Şekil 0.17'deki görüntü için τ değeri 10-80 aralığında her seferinde 10 arttırılarak yatay filtre çıktıları incelenmiştir. Her bir τ değeri için yatay filtre çıktısının şekil numaralarını içeren ilişki Tablo 0.5'de gösterilmiştir.

Tablo 0.5 Koyu görüntü için şekillerin τ değerleri tablosu

***τ* değeri**	**Şekil Numarası**
10	Şekil 0.18
20	Şekil 0.19
30	Şekil 0.20
40	Şekil 0.21
50	Şekil 0.22
60	Şekil 0.23
70	Şekil 0.24
80	Şekil 0.25

Şekil 0.18 Koyu görüntü τ =10

Şekil 0.19 Koyu görüntü τ=20

Şekil 0.20 Koyu görüntü τ=30

Şekil 0.21 Koyu görüntü τ=40

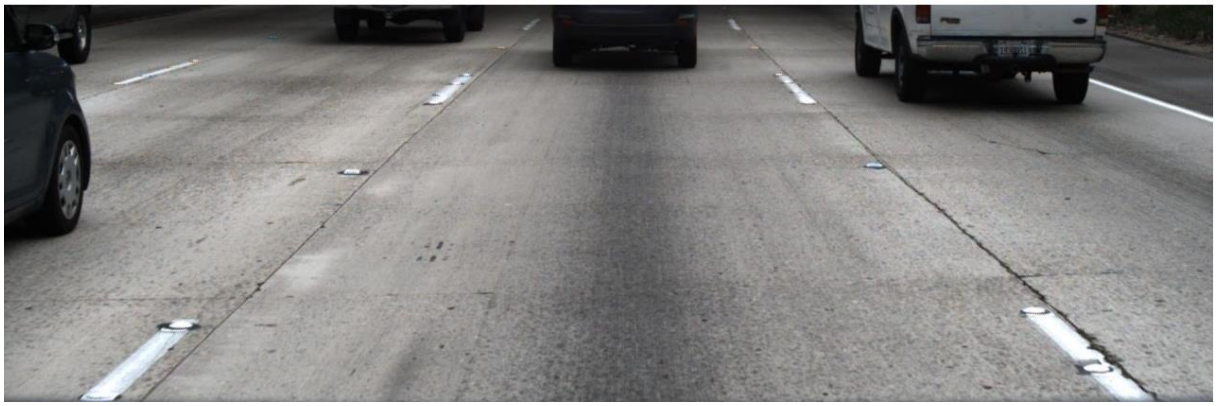
Şekil 0.22 Koyu görüntü τ=50

Şekil 0.23 Koyu görüntü τ=60

Şekil 0.24 Koyu görüntü τ=70

Şekil 0.25 Koyu görüntü τ=80

Parlak görüntü olarak Şekil 0.26'da görülen yol görüntüsü kullanılmıştır. Kullanılan görüntünün ortalama parlaklık değeri 0.3914 olarak hesaplanmıştır.

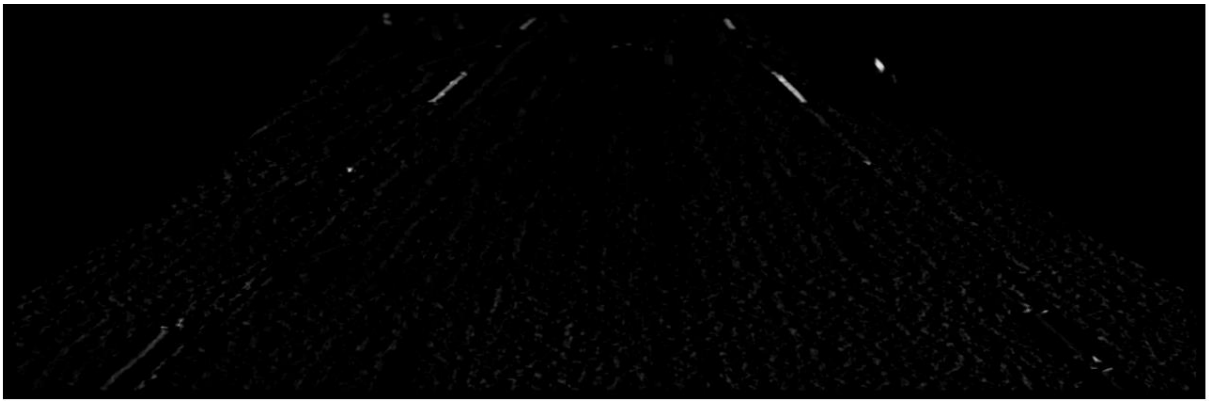


Şekil 0.26 Yatay filtre τ değeri testlerinde kullanılan parlak görüntü

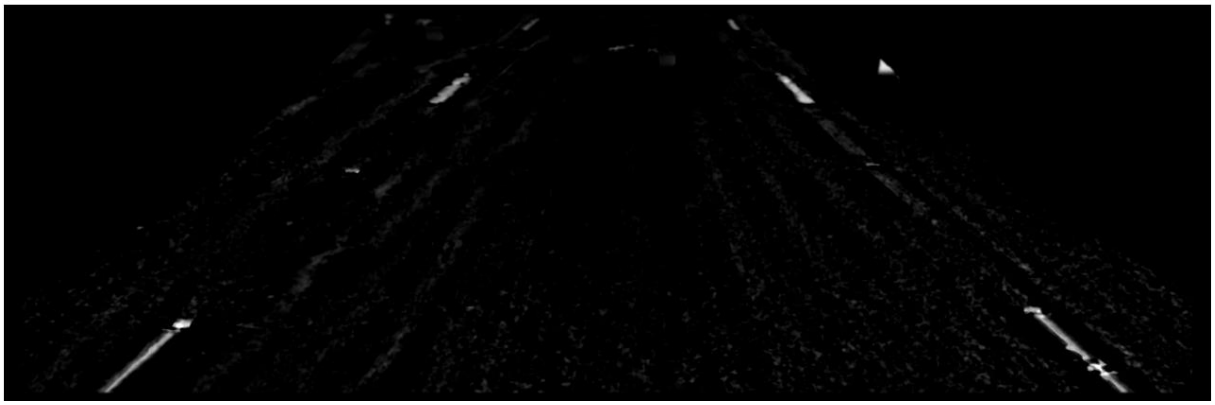
Şekil 0.26'daki görüntü için τ değeri 10-80 aralığında her seferinde 10 arttırılarak yatay filtre çıktıları incelenmiştir. Her bir τ değeri için yatay filtre çıktısının şekil numaralarını içeren ilişki Tablo 0.6'da gösterilmiştir.

Tablo 0.6 Parlak görüntü için şekillerin τ değerleri tablosu

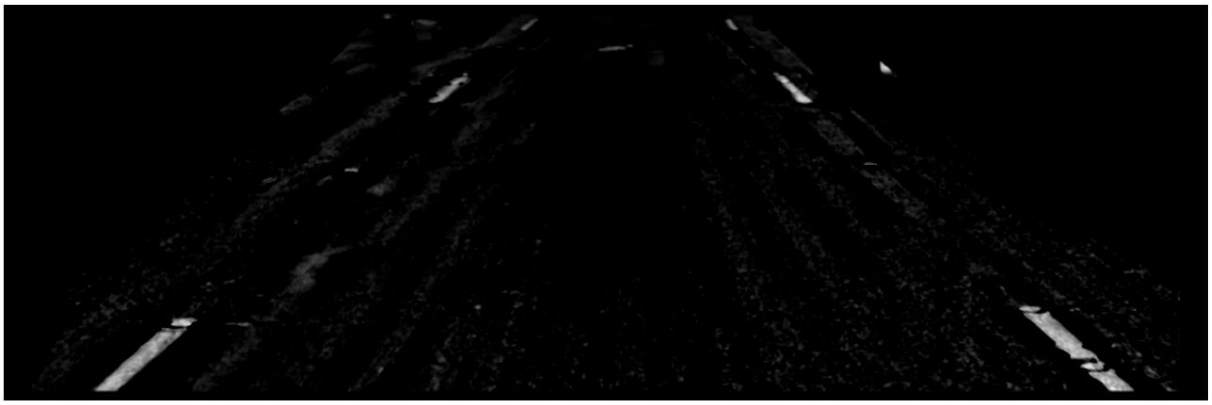
***τ* değeri**	**Şekil Numarası**
10	Şekil 0.27
20	Şekil 0.28
30	Şekil 0.29
40	Şekil 0.30
50	Şekil 0.31
60	Şekil 0.32
70	Şekil 0.33
80	Şekil 0.34



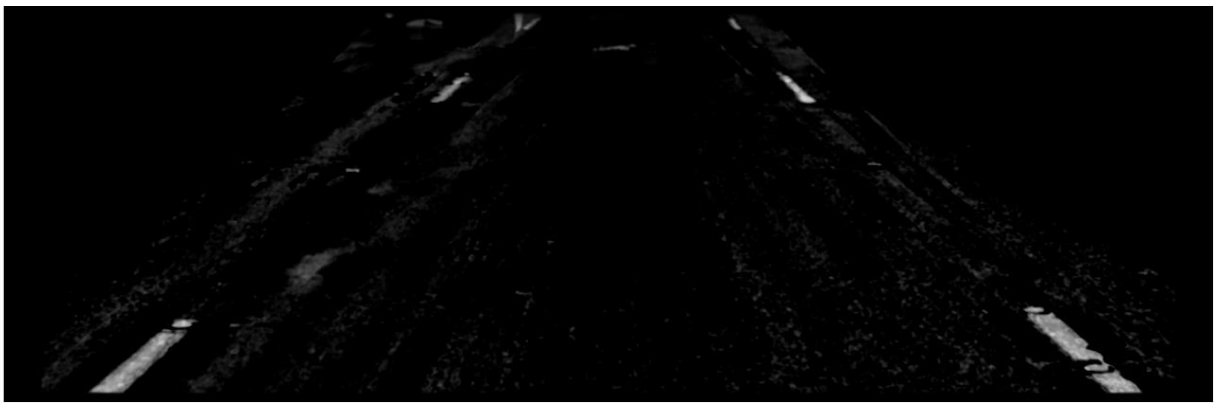
Şekil 0.27 Parlak görüntü τ=10



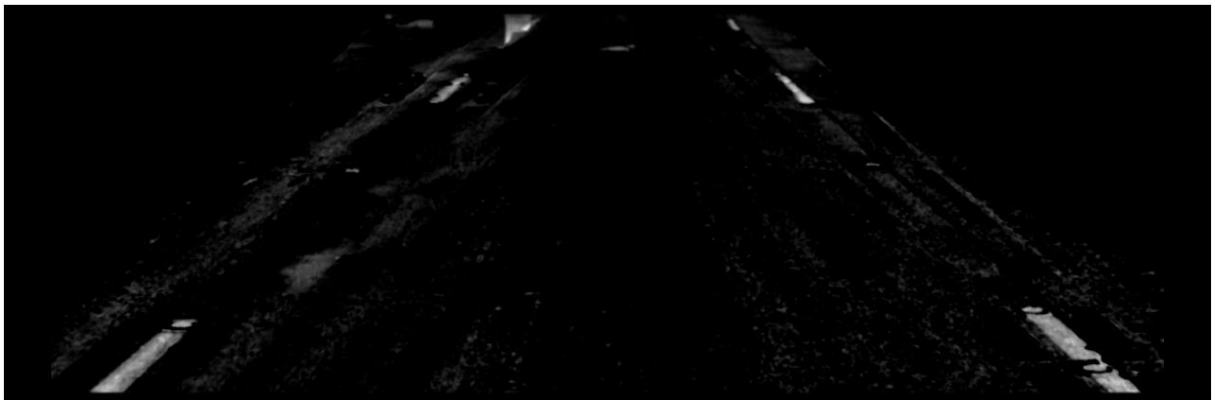
Şekil 0.28 Parlak görüntü τ=20



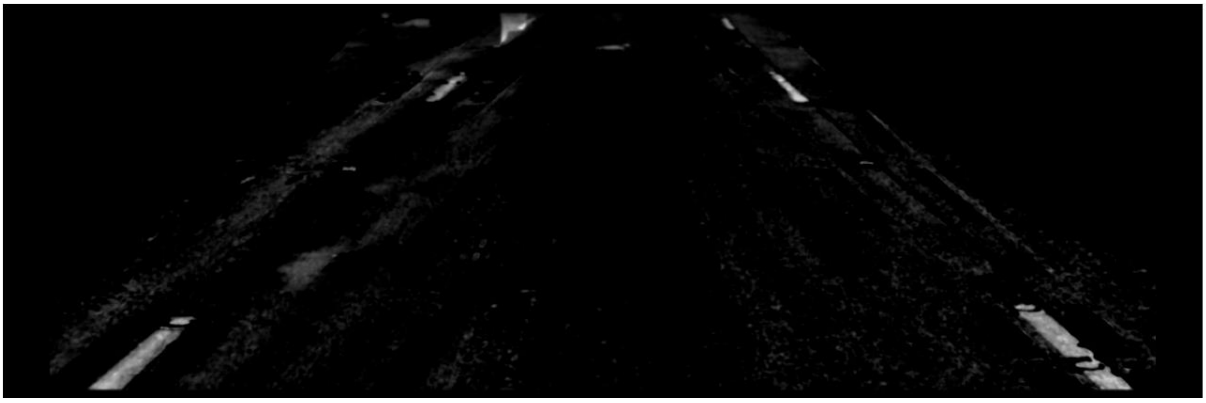
Şekil 0.29 Parlak görüntü τ=30



Şekil 0.30 Parlak görüntü τ=40

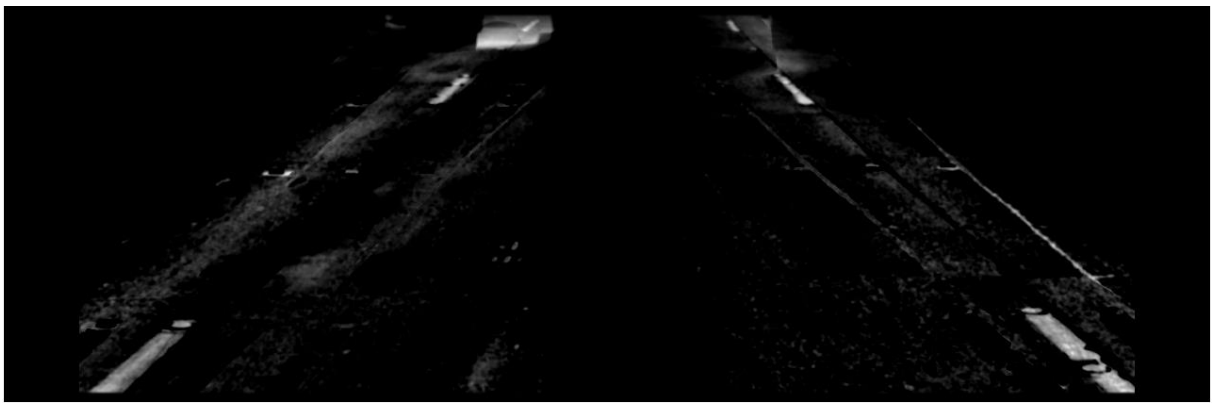


Şekil 0.31 Parlak görüntü τ=50



Şekil 0.32 Parlak görüntü τ=60

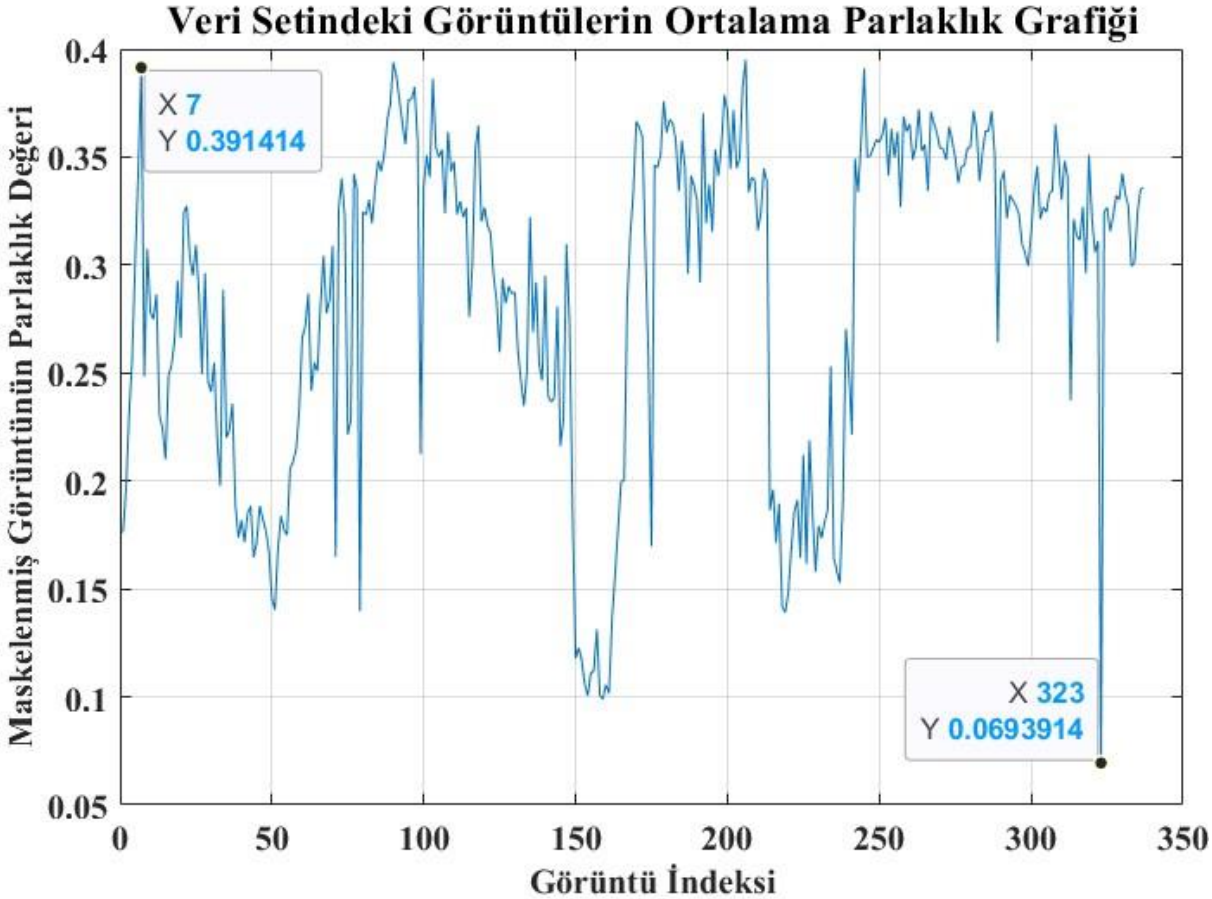
Şekil 0.33 Parlak görüntü τ=70



Şekil 0.34 Parlak görüntü τ=80

Veri setindeki görüntülerde bu analiz yapıldığında, τ parametresi ile görüntü parlaklığı arasında bir ilişki olduğu tespit edilmiştir. Bu ilişki, yol çizgisi tespiti ve doğruluğunu etkileyen önemli bir faktördür. Parlak görüntülerde düşük τ değerlerinin, koyu görüntülerde ise yüksek τ değerlerinin başarılı olduğu tespit edilmiştir. Burada buna sebep olan temel etken parlak görüntülerde yoldaki parlaklığın artmasıdır. Parlak görüntülerde τ değeri eğer gereğinden büyük seçilirse yolun yol çizgisi olmayan parlak kısımları yol çizgisi kısmının etrafına Şekil 0.34'teki gibi etki edecektir.

Bu tespitten yola çıkarak, çalışmada τ değeri görüntüdeki ortalama parlaklık seviyelerine ters orantılı olacak şekilde ayarlanmıştır. Ters orantı kurulurken maskelenmiş görüntülerin alabileceği maksimum ve minimum ortalama parlaklık değeri Şekil 0.35'te görüldüğü gibi çizdirilmiştir.



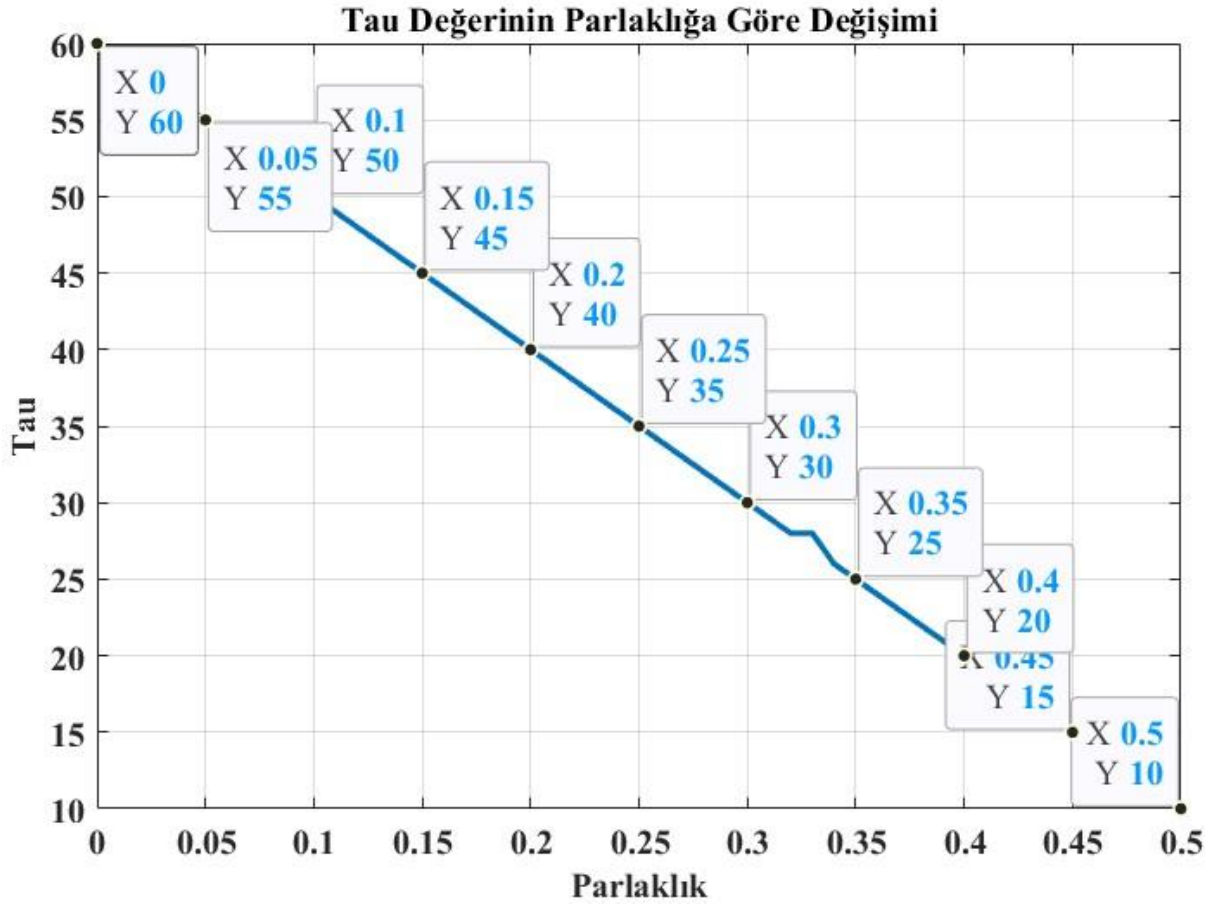
Şekil 0.35 Veri setindeki görüntülerin ortalama parlaklık grafiği

Şekil 0.35'ten de anlaşılacağı üzere ortalama parlaklık değeri $0.069 \leq \mu_{parlaklık} \leq 0.4$ aralığında olabilmektedir. Veri setindeki görüntülerde analiz yapıldığında ise τ değerinin optimal aralığının $20 \leq \tau \leq 52$ olduğu görülmüştür. Görüntünün ortalama parlaklık değeri-τ değeri arasında ters orantılı bir ilişki kurabilmek Eşitlik (3.6) kullanılmıştır. (Denklemde görülen

ortalama parlaklık değeri Görüntü Ortalama Parlaklık Değerinin Hesaplanması adımında elde edilen ortalama parlaklık değeridir.)

$$\tau=60-100*\mu_{parlaklık} \quad (3.6)$$

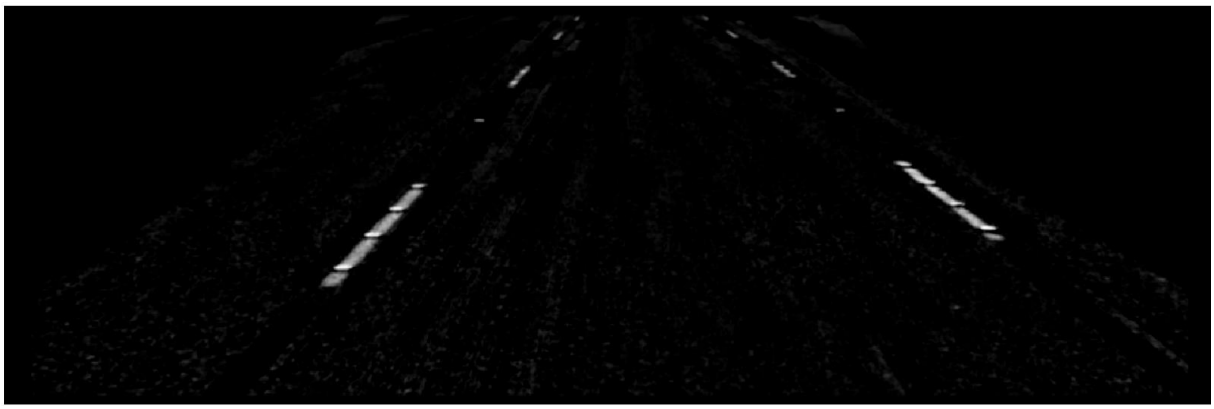
Eşitlik (3.6) kullanıldığında $0\leq\mu_{parlaklık}\leq0.5$ aralığındaki ortalama parlaklık değerlerine karşılık gelen τ değerlerini içeren grafik Şekil 0.36'daki gibidir.



Şekil 0.36 τ değerinin ortalama parlaklık değerine göre değişimi grafiği

1.13.2 Yatay filtrenin uygulanması

Yatay filtre komşuluk değerinin ayarlanması adımında elde edilen τ değeri kullanılarak görüntü Gauss Filtre çıktısı elde edilmiş görüntüler yatay filtreye tabi tutulmuştur. Şekil 0.12'deki Gauss Filtre çıktısında elde edilen görüntü Şekil 0.37'daki gibidir. Gauss filtre çıktısındaki görüntünün ortalama parlaklık değeri 0.32, τ değeri ise 28 olarak hesaplanmıştır.

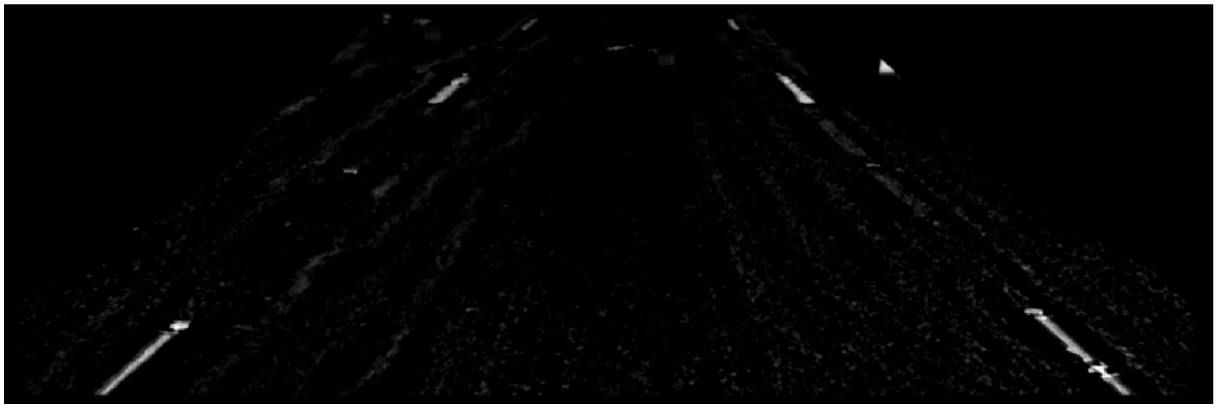


Şekil 0.37 Örnek görüntü yatay filtre çıktısı

Şekil 0.17'deki koyu görüntünün yatay filtre çıktısı Şekil 0.38'deki gibidir. Koyu görüntünün τ değeri ise 43 olarak hesaplanmıştır.

Şekil 0.38 Koyu görüntü yatay filtre çıktısı

Şekil 0.26'daki parlak görüntünün yatay filtre çıktısı Şekil 0.39'deki gibidir. Parlak görüntünün τ değeri ise 21 olarak hesaplanmıştır.



Şekil 0.39 Parlak görüntü yatay filtre çıktısı

1.14 Görüntüyü İkili Hale Getirme

Kenar algılama işleminin uygulanabilmesi için normalleştirilmiş görüntünün, ikili bir görüntü formatına dönüştürülmesi gerekmektedir. Bu amaç doğrultusunda, veri setindeki görüntüler eşikleme yöntemleri kullanılarak ikili hale getirilmiştir.

1.14.1 Eşikleme yöntemleri hakkında

Görüntü işlemede, görüntüyü ikili hale getirmek, yani her pikseli siyah veya beyaz olarak temsil etmek, birçok önemli işlem için temel bir adımdır. Bu işlem, nesneleri tespit etmek, çıkarmak, görüntüyü gruplandırmak ve daha fazlası gibi çeşitli görevlerde kullanılır. Görüntüyü ikili hale getirmek için eşikleme yöntemleri kullanılır. Eşikleme, bir görüntünün her pikselinin parlaklık değerine dayanarak siyah veya beyaz olarak sınıflandırılması işlemidir. Bu işlemde bir eşik değeri kullanılır. Kullanılan eşik değeri, hangi piksellerin siyah ve hangi piksellerin beyaz olacağını belirler. Eşitlemede çeşitli yöntemler bulunmaktadır. Başlıca eşikleme yöntemleri şunlardır:

- **Otsu'nun Eşikleme Yöntemi:** Bu yöntem, görüntüdeki iki sınıfa (ön plan ve arka plan) ait piksellerin ortalama yoğunlukları arasındaki farkı en üst düzeye çıkaran bir eşik değeri seçer. Bu yöntem, homojen arka plana sahip görüntüler için etkilidir [20].
- **Niblack'ın Eşikleme Yöntemi:** Bu yöntem, her piksel için çevresindeki piksellerin ortalama yoğunluğuna göre bir eşik değeri belirler. Bu yöntem, değişken arka plana sahip görüntüler için etkilidir [21].
- **Sauvola'nın Eşikleme Yöntemi:** Bu yöntem, Niblack [20] tarafından geliştirilen yönteme benzer, ancak piksellerin yerel varyansını da hesaba katar. Bu yöntem, gürültülü görüntüler için daha sağlamdır [22].
- **Kısmi Eşikleme:** Bu yöntem, görüntünün farklı bölgeleri için farklı eşik değerleri kullanır.
- **Adaptif Eşikleme:** Bu yöntem, her piksel için ayrı bir eşik değeri hesaplar.
- **Otsu'nun Modifiye Eşikleme Yöntemi:** Bu yöntem, Otsu'nun yönteminin bir varyasyonudur ve gürültülü görüntüler için daha sağlamdır.

1.14.2 Çalışmada kullanılan eşikleme yöntemi

Bu araştırmada sunulan algoritmada, MATLAB'ın *graythresh* fonksiyonu kullanılarak görüntülerde eşik değerleri hesaplanmıştır. MATLAB'ın *graythresh* fonksiyonu, Otsu yöntemi kullanarak gri tonlamalı görüntü I'den küresel bir eşik hesaplar [23]. Otsu'nun yöntemi, eşiklenmiş siyah ve beyaz piksellerin sınıf içi varyansını en aza indiren bir eşik seçer [24].

Bu fonksiyonun girdi parametreleri, çıktı parametreleri ve çalışmada kullanılan parametreler Tablo 0.7'de listelenmiştir.

Tablo 0.7 MATLAB'ın graythresh fonksiyonu girdi ve çıktı parametreleri

Parametre	**Parametre tanımı**	**Türü**	**Çalışmada kullanılan değer**
I	Kullanılan girdi görüntüsü	Girdi	Yatay filtre çıktısı
threshold	Eşik değeri	Çıktı	-

Bu adımda girdi görüntüsü olarak yatay filtre çıktısında elde edilen görüntü kullanılmıştır. Çıktı olarak elde edilen eşik seviyesi Çalışmada kullanılan ikili hale getirme yöntemi adımında kullanılacaktır.

1.14.3 Çalışmada kullanılan ikili hale getirme yöntemi

Bu araştırmada sunulan algoritmada, MATLAB'ın "*imbinarize"* fonksiyonu kullanılarak görüntüler ikili hale getirilmiştir. MATLAB'ın "*imbinarize"* fonksiyonu, görüntüyü ikili hale getirmek için çeşitli eşikleme yöntemlerinden birini kullanır. Bu fonksiyonun girdi parametreleri, çıktı parametreleri ve çalışmada kullanılan parametreler Tablo 0.8'de listelenmiştir. Fonksiyonda kullanılan parametreler deneysel olarak elde edilmiştir ve veri setindeki tüm görüntülerde bu değerler sabit olarak kullanılmıştır.

Tablo 0.8 MATLAB'ın imbinarize fonksiyonu girdi ve çıktı parametreleri

Parametre	**Parametre tanımı**	**Türü**	**Çalışmada kullanılan değer**
I	Kullanılan girdi görüntüsü	Girdi	Yatay filtre çıktısındaki görüntü
method	Kullanılan eşikleme yöntemi • "Global" (Varsayılan Değer) • "Adaptive"	Girdi	Global (Otsu)
threshold	Eşik Değeri	Girdi	MATLAB'ın *graythresh* fonksiyonu ile elde edilen eşik değeri
B	Elde edilen ikili görüntü	Çıktı	-

MATLAB "*imbinarize"* fonksiyonu, kullanıcının seçtiğine bağlı olarak farklı eşikleme yöntemlerinden birini kullanır. Varsayılan eşikleme yöntemi Otsu'nun yöntemidir. Diğer eşikleme yöntemleri, *"method"* argümanı ile manuel olarak seçilebilir.

Bu araştırmada görüntüyü ikili hale getirmek için Otsu'nun eşikleme yöntemi seçilmiştir. Bu yöntemin seçilmesinin sebebi, homojen arka plana sahip olan görüntülerde etkili olması ve gürültüye karşı nispeten dayanıklı olmasıdır.

Şekil 0.37'de görülen yatay filtre çıktısının ikili hale getirilmiş görüntüsü Şekil 0.40'daki gibidir.

Şekil 0.40 Görüntüyü ikili hale getirme sonucu

1.15 Küçük Nesnelerin Filtrelenmesi

Yol çizgisi analizi sürecinde, minimum yol çizgisi genişliği eşiğinin belirlenmesi ve bu eşiğin altındaki bilgilerin filtrelenmesi, algoritmanın performansını ve doğruluğunu optimize etmek için kritik bir adımdır. Literatürde, bu yaklaşımın gürültü azaltma ve özellik çıkarmada önemli bir rol oynadığı vurgulanmaktadır. Belirli bir boyutun altındaki nesnelerin (piksel kümeleri) genellikle gürültü olarak kabul edilmesi, görüntü işleme ve bilgisayarlı görü alanında yaygın bir uygulamadır [7].

Bu bağlamda, gürültü olarak sınıflandırılan küçük objeler, görüntüdeki küçük parlamalar, yol çizgisi dışı yabancı nesneler veya yol yüzeyindeki yansımalar, çakıl taşları gibi unsurları içerebilir. Bu tür unsurlar, yol çizgisi algılama algoritması tarafından hatalı pozitif sonuçlar olarak değerlendirilebilir. Dolayısıyla, bu hatalı pozitiflerin algoritmadan elenmesi, yol çizgisi algılama sürecini daha güvenilir kılar ve nihayetinde araçların yol çizgisi içinde kalmasını sağlayan karar mekanizmalarının doğruluğunu artırır.

Bu adımda belirli bir eşik değerinin altındaki nesneler filtrelenmiştir. Eşik değeri, yol çizgisi bilgisi olarak kabul edilecek minimum piksel sayısı olarak belirlenmiştir. Bu araştırmada MATLAB'ın *"bwareaopen"* fonksiyonu kullanılarak küçük nesneler etkili bir şekilde filtrelenmiştir. MATLAB'ın *"bwareaopen"* fonksiyonu, ikili görüntülerde belirli bir piksel sayısından daha küçük olan bağlı bileşenleri kaldırır [25].

Bu fonksiyonun girdi parametreleri, çıktı parametreleri ve çalışmada kullanılan parametreler Tablo 0.9'de listelenmiştir. Fonksiyonda kullanılan parametreler değerler

deneysel olarak elde edilmiştir ve veri setindeki tüm görüntülerde bu değerler sabit olarak kullanılmıştır.

Tablo 0.9 MATLAB'ın *bwareaopen* fonksiyonu girdi ve çıktı parametreleri

Parametre	**Parametre tanımı**	**Türü**	**Çalışmada kullanılan değer**
I	İkili görüntü	Girdi	İkili hale gelmiş görüntü
threshold	Eşik Değeri	Girdi	30
C	Küçük nesneler filtrelenmiş görüntü	Çıktı	-

Küçük noktaların filtrelenme çıktısı örneği Şekil 0.41'deki gibi görülebilir. Görüntüde Şekil 0.40'te görülen küçük noktaların başarıyla filtrelendiği görülmüştür.

Şekil 0.41 Küçük noktaların filtrelenmiş hali

1.16 Kenar Algılama

Görüntü işlemede kenar, farklı nesneler veya bölgeler arasındaki sınır olarak tanımlanır ve piksel yoğunluğundaki ani değişimlere karşılık gelir. Bu keskin geçişler, bir görüntünün yapısal özelliklerini belirleme ve nesneleri tanıma süreçlerinde kritik bir rol oynar. Kenar algılama, endüstriyel otomasyon, robotik ve tıbbi görüntüleme gibi çeşitli alanlarda hassas ölçümler, nesne tespiti ve sınıflandırması için temel bir araçtır. Örneğin, otomobil endüstrisinde montaj hattındaki parçaların doğru yerleştirilmesinin kontrolünde veya tıbbi görüntülemede tümörlerin sınırlarının ve boyutlarının belirlenmesinde kenar algılama algoritmaları kullanılabilir.

Kenar algılama, farklı algoritmalar aracılığıyla gerçekleştirilebilir. Yaygın olarak kullanılan Sobel, Prewitt operatörleri ve Canny kenar algılayıcısı, bu algoritmalara örnek olarak verilebilir. Sobel ve Prewitt operatörleri, görüntüyü yatay ve dikey yönlerde filtreleyerek

kenarları tespit ederken, Canny kenar algılayıcısı daha karmaşık bir yaklaşımla daha hassas sonuçlar elde etmeyi sağlar [26]. Canny algoritması, gürültü azaltma, gradyan hesaplama ve maksimum olmayan baskılama gibi adımları içerir. Bu adımlar, kenarları daha net bir şekilde belirlemeye ve hatalı pozitifleri azaltmaya yardımcı olur. Bu araştırmada, kenar algılama algoritması olarak Canny kenar algılama yöntemi tercih edilmiştir.

1.16.1 Canny kenar algılama

Canny kenar algılama yöntemi, görüntü işleme alanında yaygın olarak kullanılan etkili bir kenar tespit tekniğidir. 1986 yılında John Canny tarafından geliştirilen bu yöntem, gürültülü görüntülerde bile kenarları doğru bir şekilde tespit edebilme yeteneği sayesinde geniş kabul görmüştür [27]. Canny kenar algılama yöntemi, aşağıdaki adımları içerir [7]:

- **Gürültü Azaltma**

Görüntü üzerinde Gauss bulanıklaştırma işlemi uygulanarak gürültü azaltılır, bu da kenar algılama hassasiyetini artırır. Gauss filtresi, G(x, y) Eşitlik (3.7)'deki formülle ifade edilir:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.7}$$

Eşitlik (3.7)'de x ve y, merkezi piksele göre konumları, σ ise standart sapmayı ifade eder ve filtre çekirdeğinin genişliğini belirler. Standart sapma değeri arttıkça, filtreleme etkisi de artar ve görüntü daha fazla yumuşatılır.

- **Gradyan Hesaplama**

Gürültüsü azaltılmış görüntüdeki gradyanlar hesaplanır. Gradyan, görüntünün yoğunluk değişimlerini belirler. Gradyan büyüklüğü M(x,y) ve yönü ∅(x,y), Eşitlik (3.8) ve Eşitlik (3.9)'a göre hesaplanır:

$$M(x,y)=\sqrt{G_x^2+G_y^2} \tag{3.8}$$

$$\emptyset(x,y)=\arctan\left(\frac{G_y}{G_x}\right) \tag{3.9}$$

Burada G_x ve G_y sırasıyla x ve y yönlerindeki bileşenlerdir.

- **Maksimum Olmayan Bastırma**

Gradyan büyüklüğü haritasında yerel maksimumların korunması amacıyla, her pikselin gradyan yönündeki komşularıyla karşılaştırılması yapılır. Sadece yerel maksimum olan pikseller kenar olarak işaretlenir.

- **Çift Eşikleme**

Belirlenen iki eşik değeri (yüksek eşik T_H ve düşük eşik T_L) kullanılarak, gerçek kenar pikselleri ile olası kenar pikselleri ayrılır.

- **Histerezis Kenar İzleme**

Düşük eşik değerinin üzerindeki pikseller arasında yüksek eşik değerinin üzerindeki piksellerle bağlantılı olanlar kenar olarak kabul edilir. Böylece zayıf kenar pikselleri, güçlü kenar pikselleri ile bağlantılı ise korunur.

Bu adımlar sonucunda, Canny kenar algılama yöntemi ile görüntüdeki kenarlar başarılı bir şekilde tespit edilir. Canny algoritması, düşük sinyal/gürültü oranına sahip görüntülerde yüksek performans gösterir ve bu nedenle bilgisayarlı görü uygulamalarında sıkça tercih edilir.

1.16.2 Kenar algılanmanın uygulanması

Bu araştırmada kenar algılama işlemleri için MATLAB'ın "edge" fonksiyonundan yararlanılmıştır. Canny kenar algılama algoritması, gürültüye karşı dayanıklılığı ve kenarları doğru bir şekilde tespit edebilme yeteneği nedeniyle tercih edilmiştir. "Edge" fonksiyonunun Eşitlik (3,10)'da görüldüğü gibi kullanımı vardır [28]:

$$\text{BW = edge(I, 'Canny', threshold, sigma);} \tag{3.10}$$

Bu fonksiyonun girdi parametreleri, çalışmada kullanılan parametreler Tablo 0.10'da listelenmiştir. Fonksiyonda kullanılan parametreler deneysel olarak elde edilmiştir ve veri setindeki tüm görüntülerde bu değerler sabit olarak kullanılmıştır. Bu fonksiyonun çıktı parametreleri Tablo 0.11'de listelenmiştir.

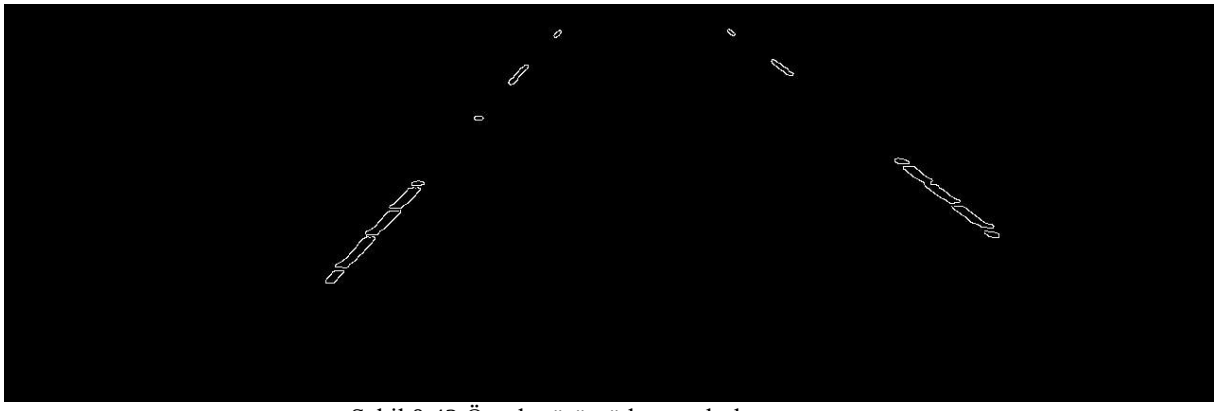
Tablo 0.10 MATLAB'ın edge fonksiyonu girdi parametreleri ve kullanılan değerler

Parametre	**Parametre Tanımı**	**Çalışmada kullanılan değer**
I	Girdi görüntüsü.	Küçük nesnelerin filtrelenmesi adımında elde edilen görüntü
method	Kenar algılama yöntemini belirtir. Aşağıdaki seçenekleri bulunur: • Sobel • Prewitt • Roberts • log • zerocross • Canny • approxcanny	Canny
threshold	Kenar algılama için kullanılan eşik değerleri. Kenar algılama yöntemi olarak Canny seçilirse bu değer, iki değerden oluşan bir vektördür. Vektör, yüksek eşik T_H ve düşük eşik T_L değerlerini içerir $[T_L \ T_H]$. Eğer threshold parametresi için tek bir değer belirtilirse, bu değer T_H olarak kullanılır ve T_L otomatik olarak hesaplanır.	[0.05 0.3]
sigma	Canny filtre için kullanılan standart sapma(σ) değeridir. Varsayılan değeri $\sqrt{2}$'dir. "Edge" fonksiyonu kullanılacak filtre genişliğini sigma değerine göre otomatik olarak ayarlar.	1

Tablo 0.11 MATLAB'ın edge fonksiyonu çıktı parametreleri

Parametre	**Parametre Tanımı**
C	Çıktı görüntüsü.
threshOut	Kenar algılama yöntemi olarak Canny seçilirse bu değer, iki değerden oluşan bir vektördür. Vektör, yüksek eşik T_H ve düşük eşik T_L değerlerini içerir[T_L T_H]. Diğer hesaplamalarda bu değer kullanılmamıştır.
G_x	Hesaplanan yatay gradyan değerini döndürür. I görüntüsü ile aynı boyutlardadır. Diğer hesaplamalarda bu değer kullanılmamıştır.
G_y	Hesaplanan dikey gradyan değerini döndürür. I görüntüsü ile aynı boyutlardadır. Diğer hesaplamalarda bu değer kullanılmamıştır.

Küçük Nesnelerin Filtrelenmesi adımında küçük nesneleri filtrelenmiş görüntülere kenar algılama uygulanmıştır ve kenarlara sahip görüntüler elde edilmiştir. Şekil 0.41'de gösterilen filtrelenmiş görüntüye kenar algılama işlemi uygulandığında, Şekil 0.42'deki gibi kenarları vurgulanmış bir görüntü elde edilmiştir.



Şekil 0.42 Örnek görüntü kenar algılama sonucu

1.17 Doğru Tespiti

Hough dönüşümü, doğru tespiti çalışmalarında sıklıkla kullanılan bir yöntemdir ve yol çizgisi takibi için de güçlü bir araçtır [29]. Yol çizgisi takibi sürecinde Hough dönüşümü, üç temel adımda gerçekleştirilir:

- **Hough Dönüşümü:** Görüntüdeki kenar noktaları, Hough uzayı adı verilen bir parametre uzayına dönüştürülür. Bu uzayda, her nokta potansiyel bir doğruyu temsil eder. Kenar noktaları, Hough uzayında oylama yaparak çizgi benzeri yapılara karşılık gelen yoğunlaşma bölgeleri oluşturur.

- **Zirve Noktalarının Hesaplanması:** Hough uzayındaki yoğunlaşma bölgeleri, potansiyel yol çizgilerini temsil eden zirve noktaları olarak belirlenir. Bu zirve noktaları, yol çizgilerinin konum ve yön bilgisi hakkında önemli ipuçları sağlar.
- **Zirve Noktalarının Doğru Haline Getirilmesi:** Tespit edilen zirve noktaları, gerçek yol çizgilerini en iyi şekilde temsil edecek şekilde optimize edilir.

1.17.1 Hough dönüşümü

Yol çizgisi takibi algoritmasında, kenarların algılanmasının ardından, Hough dönüşümü kullanılarak bu kenarların ifade ettiği şekiller tanımlanır. Bu yöntem, kenarların belirli bir matematiksel modele göre analiz edilerek görüntüdeki çizgilerin veya diğer geometrik şekillerin belirlenmesine olanak tanır [30]. Böylece, yol çizgileri elde edilebilir [31].

Standard Hough Dönüşümü (SHT), dijital görüntü işlemelerinde, geometrik şekillerin belirlenmesinde yaygın olarak kullanılan bir algoritmadır. İlk kez 1962 yılında Paul Hough tarafından önerilen bu teknik, özellikle çizgilerin ve eğrilerin tanınmasında etkilidir [29]. Bu algoritmanın temel prensibi, görüntü uzayındaki noktaların parametre uzayındaki eğrilere dönüştürülmesi ve bu eğrilerin kesişim noktalarının, ilgili geometrik şekillerin tespiti olarak değerlendirilmesidir [30].

Hough dönüşümü, başlangıçta Duda and Hart [31] tarafından doğrular, daireler ve çeşitli çokgenler için geliştirilmiştir. Daha sonra Ballard [30] tarafından farklı şekiller için iyileştirilmiştir.

Hough dönüşümünün yol çizgisi takibi için kullanılmasının temel nedenleri şunlardır:

- **Gürültüye ve Eksik Veriye Karşı Dayanıklılık:** Hough dönüşümü, görüntüdeki gürültü ve bazı piksellerin eksik olması gibi durumlarda bile şekilleri tespit etmede başarılıdır. Bu, özellikle gerçek dünya görüntülerinde sıkça karşılaşılan bir sorundur ve Hough dönüşümünün bu tür zorluklara karşı dayanıklı olması onu tercih edilir kılar [31].
- **Basit ve Etkili Bir Yöntem:** Hough dönüşümü, matematiksel olarak basit bir prensibe dayanır ve bu sayede kolayca anlaşılabilir ve uygulanabilir. Aynı zamanda, şekil tespiti konusunda oldukça etkili bir yöntemdir [32].
- **Parametrik Şekillerin Tespiti:** Hough dönüşümü, doğru, daire, elips gibi parametrik olarak ifade edilebilen şekilleri tespit etmede oldukça başarılıdır. Bu, özellikle endüstriyel uygulamalarda ve nesne tanıma gibi alanlarda önemli bir avantajdır [33].

- **Farklı Şekiller İçin Genişletilebilirlik:** Hough dönüşümü, sadece doğru ve daireler için değil, diğer parametrik şekiller için de genişletilebilir. Bu, farklı şekillerin tespit edilmesi gereken uygulamalarda esneklik sağlar [30].

Hough dönüşümü aşağıdaki adımlardan oluşur:

- **Kenar Tespiti:** Görüntüdeki kenarları tespit etmek için bir kenar algılama algoritması kullanılır. Bu çalışmada, kenar tespiti amacıyla Canny algoritması kullanılmıştır.
- **Parametrik Uzay:** Hough dönüşümü, şekillerin parametrik bir temsilini kullanır. Hough dönüşümünde, bir doğru için Eşitlik (3.11) kullanılır.

$$\rho = x\cos(\theta) + y\sin(\theta) \tag{3.11}$$

Eşitlik (3.11)'de:

- ρ : Doğrunun orijine olan en kısa mesafesidir (normal mesafesi).
- θ : Normalin açısıdır (0 ile 180 derece arasında).
- (x, y) : Görüntü uzayındaki bir noktanın koordinatlarıdır.

- **Akümülatör (Toplayıcı):** Görüntüdeki her bir kenar noktası (x, y) için $A(\rho,\theta)$ akümülatör matrisi Eşitlik (3.11)'e göre hesaplanır. Bu işlem, tüm olası θ değerleri için ρ değeri hesaplanarak gerçekleştirilir.

Bu araştırmada, Hough dönüşümünün uygulanması amacıyla MATLAB'ın "*hough"* fonksiyonu kullanılmıştır [34]. Bu fonksiyon, ikili görüntüdeki doğruları bulmak için Hough dönüşümü uygular.

Bu fonksiyonun girdi parametreleri, çıktı parametreleri ve çalışmada kullanılan parametreler Tablo 0.12'da listelenmiştir. Fonksiyonda kullanılan parametreler deneysel olarak elde edilmiştir ve veri setindeki tüm görüntülerde bu değerler sabit olarak kullanılmıştır.

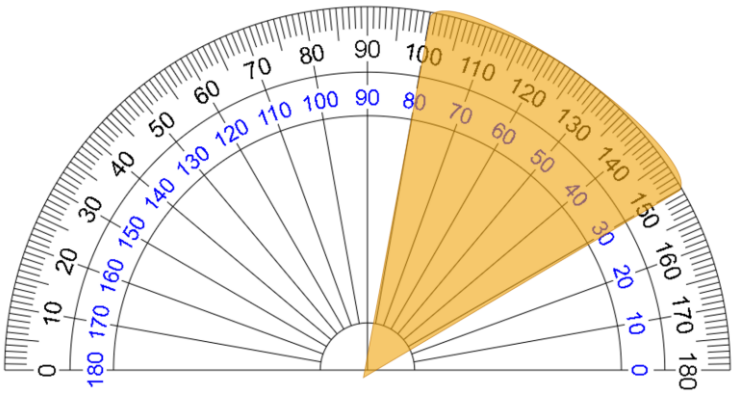
Tablo 0.12 MATLAB'ın hough fonksiyonu girdi ve çıktı parametreleri

Parametre	**Parametre Tanımı**	**Türü**	**Çalışmada kullanılan değer**
BW	Bir ikili görüntü matrisi. Görüntüdeki beyaz pikseller, doğruların geçebileceği noktaları belirtir.	Girdi	Görüntü

Theta	İsteğe bağlı bir parametre. Doğruların açılarının aralığını belirten bir vektördür. Varsayılan değeri -90:0.5:89.5 derecedir.	Girdi	-80:1:-30 ve 30:1:80
RhoResolution	İsteğe bağlı bir parametre. Doğruların uzaklıklarının çözünürlüğünü belirten bir sayısal değerdir. Varsayılan değeri 1 pikseldir.	Girdi	1
H	Hough dönüşümünün sonucunu içeren bir matristir. Hough matrisi, parametre uzayındaki her bir hücrenin oylarını içerir. Hough matrisinin boyutları, girdi olarak verilen açı aralığı ve çözünürlüğüne bağlıdır. Hough matrisinin satırları, doğruların uzaklıklarını, sütunları ise doğruların açılarını temsil eder.	Çıktı	-
Theta	Girdi olarak verilen veya varsayılan olarak kullanılan doğru açılarının aralığını içeren bir vektördür.	Çıktı	-
Rho	Hough matrisinin satırlarına karşılık gelen doğru uzaklıklarını içeren bir vektördür.	Çıktı	-

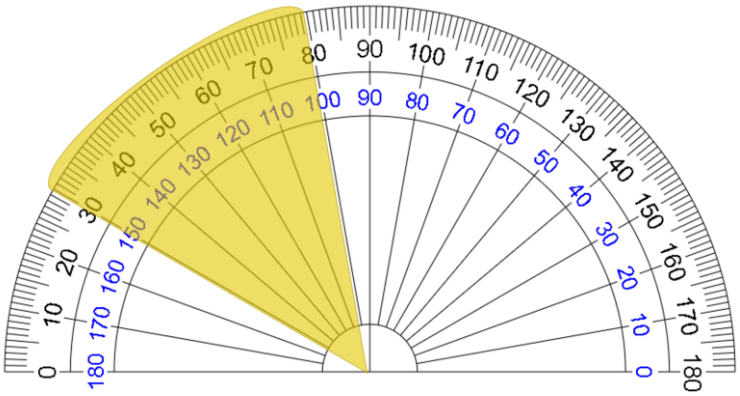
MATLAB'ın "*hough"* fonksiyonu, elde edilecek çizgilerin beklenen açı aralıklarını θ parametresiyle girdi olarak alabilmektedir. Bu özellik sayesinde, belirlenen aralık dışında kalan açı değerlerine sahip doğrular, fonksiyon tarafından otomatik olarak filtrelenir.

Şekil 0.43'de görüldüğü gibi, sol yol çizgisi için 30° ile 80° arasındaki θ değerine sahip doğrular yol çizgisi adayı olarak hesaba katılmıştır. Bu aralık dışındaki pozitif θ değerlerine sahip doğrular, "*hough"* fonksiyonunun çıktısında yer almamıştır.



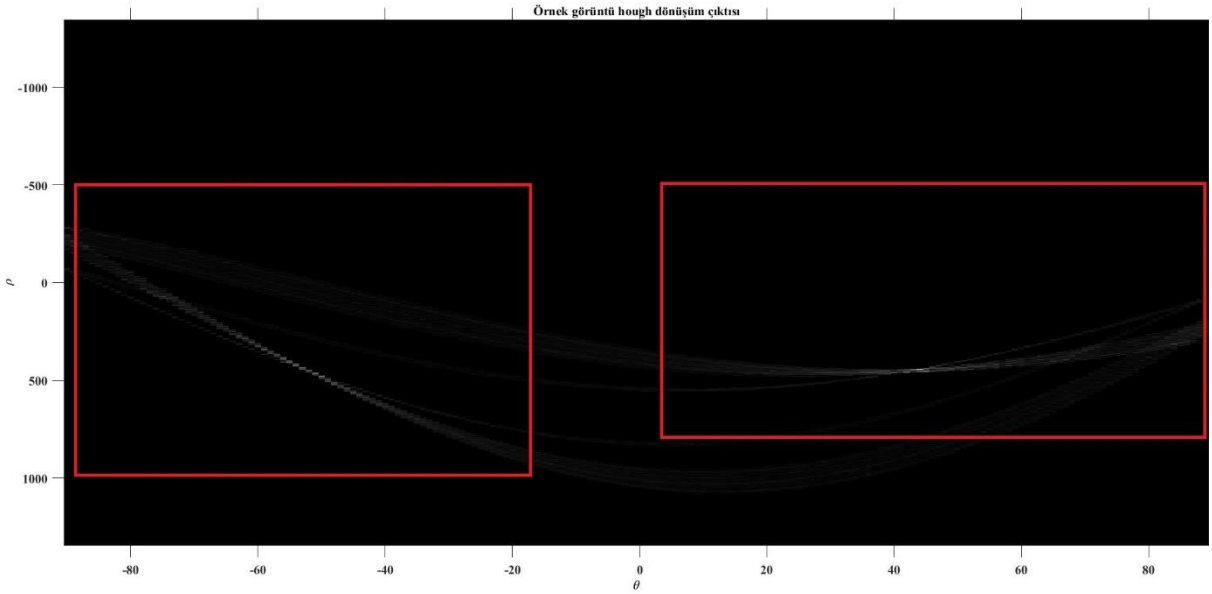
Şekil 0.43 Sol yol çizgisi için kullanılan açı aralığı

Şekil 0.44'de görüldüğü gibi, sağ yol çizgisi için -30° ile -80° arasındaki (100° ile 150° arasındaki) θ değerine sahip doğrular yol çizgisi adayı olarak hesaba katılmıştır. Bu aralık dışındaki negatif θ değerlerine sahip doğrular, "*hough*" fonksiyonunun çıktısında yer almamıştır.



Şekil 0.44 Sağ yol çizgisi için kullanılan açı aralığı

Şekil 0.42'de gösterilen kenar algılama sonucu elde edilen görüntü, "*hough*" fonksiyonuna girdi olarak verildiğinde, Şekil 0.45'te görülen Hough dönüşüm çıktısı elde edilmiştir. Bu çıktı, potansiyel yol çizgilerini temsil eden ρ ve θ değerlerini içermektedir. Hough dönüşümü sonucunda elde edilen çıktı üzerinde tespit edilen doğrular, sol ve sağ şerit çizgilerini temsil etmektedir. Söz konusu sol ve sağ şerit çizgilerini ifade eden örnek iki doğru, kırmızı alan içerisinde işaretlenmiştir.



Şekil 0.45 Hough Dönüşüm Çıktısı

1.17.2 Zirve noktalarının hesaplanması

Hough dönüşümü sürecinde, doğruların uzunluk ve açı bilgilerini barındıran yerel bir akümülatör matrisi oluşturulmuştur. Bununla birlikte, bu matriste yer alan hatalı değerlerin ayıklanması gerekmektedir. Bu bağlamda, yerel maksimum noktaların tespiti önem arz etmektedir [31].

Yerel maksimum noktalarının bulunması amacıyla MATLAB'ın "houghpeaks" fonksiyonu kullanılmıştır [35]. Bu fonksiyon, Hough dönüşümü matrisini tarayarak en yüksek yoğunluğa sahip noktaları bulur. Ardından, bu noktaların komşuluklarını inceleyerek yerel maksimum olup olmadıklarını kontrol eder. Eğer bir nokta komşularından daha yüksek bir yoğunluğa sahipse, yerel maksimum olarak kabul edilir. Bu noktalar, Hough dönüşümü ile tespit edilen doğru veya daire gibi şekillerin parametrelerini belirlemektedir.

MATLAB'ın "*houghpeaks"* fonksiyonu, doğruların hassas bir şekilde algılanabilmesi için kullanıcıdan maksimum zirve noktası sayısı, eşik değeri gibi parametrelerin girilmesine imkân sağlamaktadır.

Bu fonksiyonun girdi parametreleri ve çalışmada kullanılan parametreler Tablo 0.13'de listelenmiştir. Fonksiyonda kullanılan parametreler deneysel olarak elde edilmiştir ve veri setindeki tüm görüntülerde bu değerler sabit olarak kullanılmıştır. Bu fonksiyonun çıktı parametreleri Tablo 0.14'te listelenmiştir.

Tablo 0.13 MATLAB'ın houghpeaks fonksiyonu girdi parametreleri ve kullanılan değerler

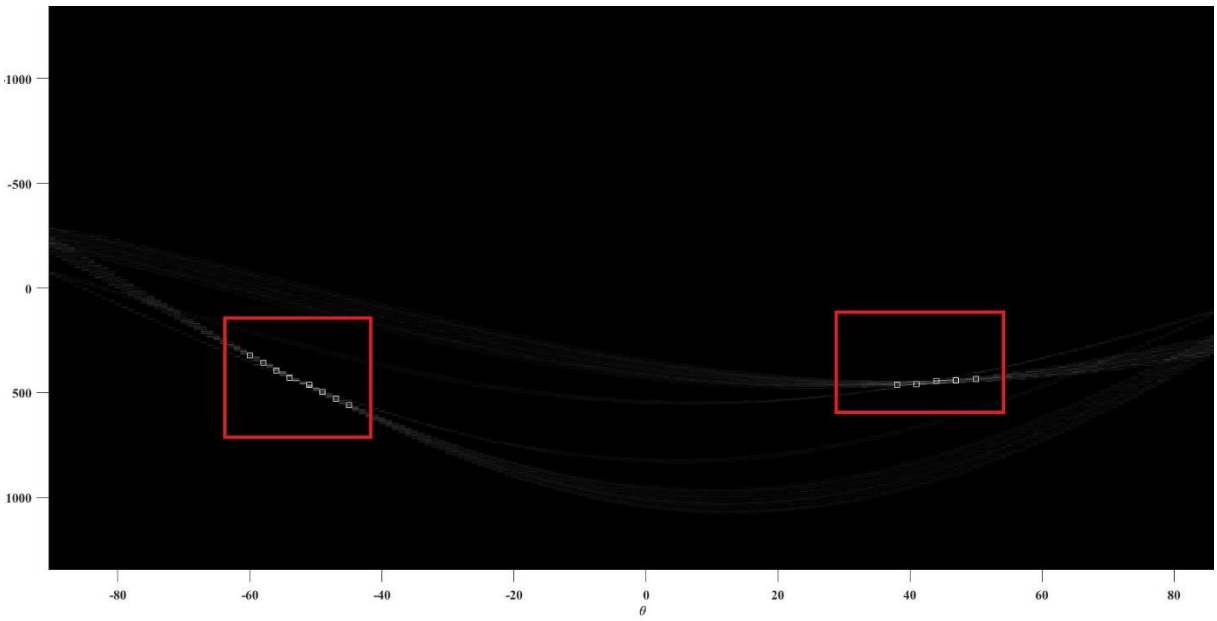
Parametre	**Parametre Tanımı**	**Çalışmada kullanılan değer**
H	Hough dönüşümü ile elde edilen akümülatör matrisidir. M x N boyutunda bir matris olmalıdır. H matrisinin her bir elemanı, belirli bir θ ve ρ değeri için Hough uzayındaki oylama sayısını gösterir. H matrisinin boyutu, θ ve ρ vektörlerinin uzunluğuna bağlıdır.	Doğru Tespiti adımında elde edilen H matrisidir.
numpeaks	Bulunacak maksimum zirve noktası sayısıdır. Varsayılan değeri 1'dir. Bu parametre, akümülatör matrisindeki en yüksek oylama sayısına sahip numpeaks kadar noktayı döndürür. Eğer numpeaks değeri, akümülatör matrisindeki eşik değerinden büyük olan noktaların sayısından küçükse, fonksiyon sadece eşik değerinden büyük olan noktaları döndürür.	40
threshold	Akümülatör matrisindeki noktaların seçilmesi için bir alt sınır belirler. Varsayılan değeri, akümülatör matrisinin maksimum değerinin yarısıdır. Bu parametre, akümülatör matrisindeki noktaların eşik değerinden büyük olup olmadığına göre filtrelenmesini sağlar. Eşik değeri, görüntüdeki gürültüyü azaltmak ve istenmeyen şekilleri elemek için kullanılabilir.	ceil(0.2*max(H(:)))
NHoodSize	Yerel maksimum arama için kullanılan komşuluk boyutudur. Varsayılan değeri, akümülatör matrisinin boyutunun [2/100 2/100] katıdır. Bu parametre, akümülatör matrisindeki her bir noktanın, NHoodSize boyutunda bir komşuluk içindeki diğer noktalarla karşılaştırılmasını belirler. Eğer bir nokta, komşuluk içindeki en büyük değere sahipse, yerel maksimum olarak kabul edilir. NHoodSize değeri, akümülatör matrisindeki yerel maksimum sayısını etkiler. NHoodSize değeri küçüldükçe, yerel maksimum sayısı artar. NHoodSize değeri büyüdükçe, yerel maksimum sayısı azalır.	[21 31]

Tablo 0.14 MATLAB'ın houghpeaks fonksiyonu çıktı parametreleri

Parametre	**Parametre Tanımı**
Peaks	Bulunan zirve noktalarının satır ve sütün koordinatlarını ifade eder.

MATLAB'ın *"houghpeaks"* fonksiyonu ile zirve noktalarının hesaplanması neticesinde, akümülatör matrisinin yerel maksimum noktaları tespit edilmiştir.

Şekil 0.45'te yer alan Hough dönüşümü çıktısından elde edilen tepe noktalarının yerel maksimum noktaları tespit edilmiştir. Bu noktaların, Hough dönüşüm çıktısı üzerindeki görselleştirilmesi Şekil 0.46'da sunulmuştur. Hough dönüşümü çıktısı üzerinde tespit edilen noktalar, sol ve sağ şerit çizgilerine karşılık gelen maksimum noktaları temsil etmektedir. Belirlenen bu örnek görüntüye ait maksimum noktalar, kırmızı bölgeler içerisinde görselleştirilmiştir.



Şekil 0.46 Elde edilen yol çizgisi adaylarının hough dönüşümü üzerine çizimi

Şekilden de görüleceği üzere, sağ ve sol yol çizgileri için birden fazla zirve noktası elde edilebilmektedir. Bu zirve noktalarının doğrulara dönüştürülmesi gerekmektedir.

1.17.3 Zirve noktalarının doğru haline getirilmesi

Hough dönüşümü sürecinde elde edilen zirve noktalarının, sol ve sağ yol çizgisi adayları olarak doğrulara dönüştürülmesi gerekmektedir. Bu dönüşüm için MATLAB'ın *"houghlines"* fonksiyonu kullanılmıştır [36]. Bu fonksiyon, "*houghpeaks*" fonksiyonu ile elde edilen zirve noktalarından ve verilen parametrelerden yararlanarak doğrular oluşturur.

Bu fonksiyonun girdi parametreleri ve çalışmada kullanılan parametreler Tablo 0.15'de listelenmiştir. Fonksiyonda kullanılan parametreler deneysel olarak elde edilmiştir ve veri setindeki tüm görüntülerde bu değerler sabit olarak kullanılmıştır. Bu fonksiyonun çıktı parametreleri Tablo 0.16'da listelenmiştir.

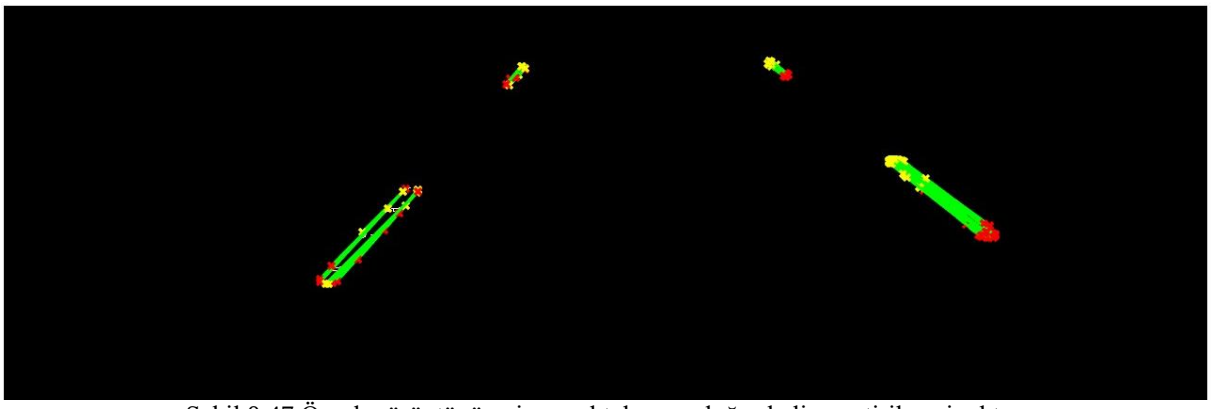
Tablo 0.15 MATLAB'ın houghlines fonksiyonu girdi parametreleri ve kullanılan değerler

Parametre	**Parametre Tanımı**	**Çalışmada kullanılan değer**
H	Hough dönüşümü adımında elde edilen akümülatör matrisi.	Hough dönüşümü adımında elde edilen akümülatör matrisi.
Theta	X ekseni ve ρ vektörü arasındaki doğru dönüş açısının derece biriminde matrisi.	Hough dönüşümü adımında elde edilen θ matrisi
Rho	Piksel cinsinden orijinden uzaklık.	Hough dönüşümü adımında elde edilen ρ matrisi
peaks	Zirve noktalarının hesaplanması adımında elde edilen zirve noktaları.	Zirve noktalarının hesaplanması adımında elde edilen peaks matrisi
Fillgap	Hough dönüşümü sonucunda aynı grupta yer alan iki çizgi parçası arasındaki mesafe bu değerden küçükse, bu çizgi parçalarını birleştirilir.	150
MinLenght	Piksel cinsinden elde edilebilecek minimum yol çizgisi uzunluğu. Bu uzunluğun altında kalan yol çizgileri elenir.	5

Tablo 0.16 MATLAB'ın houghlines fonksiyonu çıktı parametreleri

Parametre	Parametre Tanımı
lines	Elde edilen çizgi bilgileridir. Aşağıdaki alanları içeren bir veri yapısıdır: • point1:Doğrunun başlangıç noktasının [X Y] cinsinden koordinatlarını içerir. • point2: Doğrunun bitiş noktasının [X Y] cinsinden koordinatlarını içerir. • θ: Hough dönüşüm parçasının derece cinsinden açı değeri. ρ: Hough dönüşüm parçasının ρ eksenindeki koordinatıdır.

Şekil 0.46'da belirtilen zirve noktaları, "houghlines" fonksiyonu aracılığıyla doğrusal yapılara dönüştürülmüştür. Elde edilen bu doğrular, Şekil 0.42'de görülen kenar algılama uygulanmış görüntünün üzerine yerleştirildiğinde Şekil 0.47 elde edilmiştir.



Şekil 0.47 Örnek görüntünün zirve noktalarının doğru haline getirilmesi çıktısı

1.18 Doğruların Gruplandırılması

Bu aşamada, elde edilen doğrular filtrelenerek sol ve sağ yol çizgisi olmak üzere iki gruba ayrılmıştır. Bu ayrımın temel amacı, sol ve sağ yol çizgisi gruplarının ilerleyen adımlarda kendi içlerinde ağırlıklı ortalama işlemine tabi tutulmasını sağlamaktır. Çizgilerin gruplandırılması sırasında, yol çizgisi adayı olmayan doğrular da filtrelenerek elenmiştir.

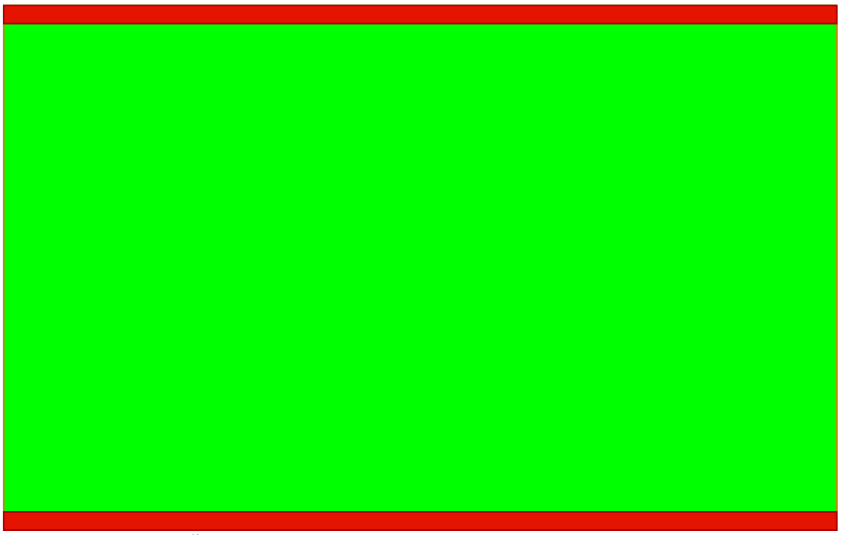
Zirve noktalarının doğrulara dönüştürülmesi sürecinde elde edilen veri kümesi, sol ve sağ yol çizgilerine ait olmayan doğruları da içerebilmektedir. Yol çizgisi takibinin etkin bir şekilde gerçekleştirilebilmesi için bu doğruların ayıklanması gerekmektedir. Bu amaçla, ekstrapolasyon uygulanmış doğrular önceden belirlenmiş kriterlere göre filtrelenerek elenmiştir.

1.18.1 Doğruların ekstrapolasyonu

Önceki aşamalarda elde edilen çizgiler, farklı uzunluklara sahip olup, başlangıç ve bitiş koordinatları değişkenlik göstermektedir. Aynı eğime sahip olmalarına rağmen, farklı yatay merkezlere sahip olan bu çizgiler, çeşitli nedenlerle Şekil 0.48'dek gibi farklı konumlarda yer alabilmektedirler.

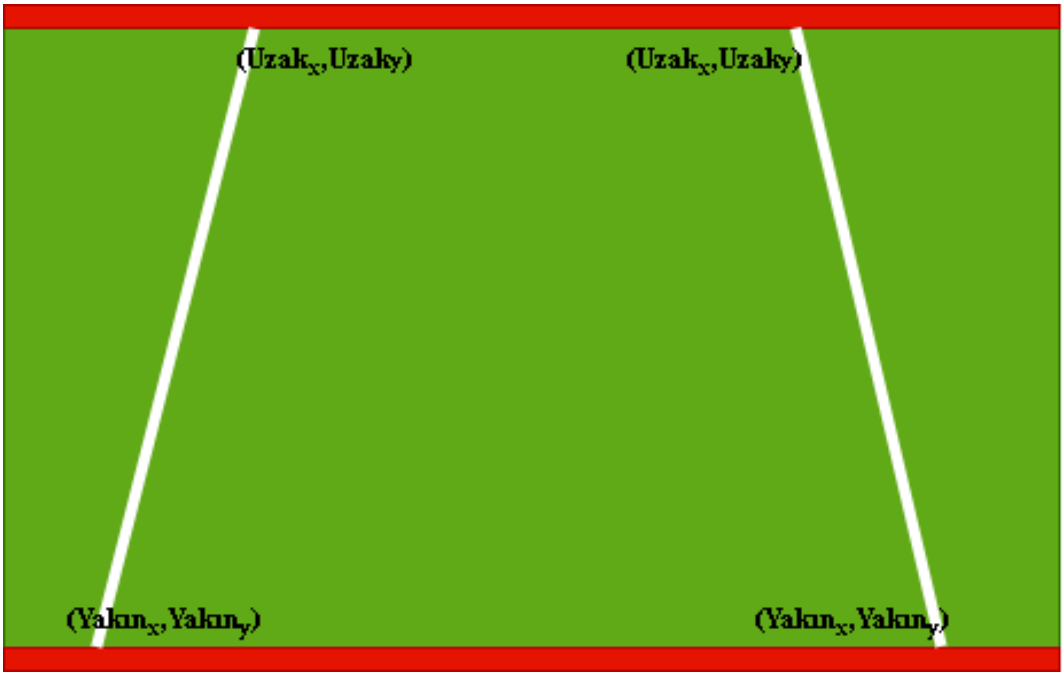
Şekil 0.48 Ekstrapolasyon uygulanmamış sol yol çizgisi adayları

Bu durum, çizgilerin sol ve sağ yol çizgileyle tutarlı bir şekilde ilişkilendirilmesini engellemektedir. Bunun önüne geçebilmek amacıyla doğrular belirlenen dikey maksimum ve minimum dikey koordinata ulaşana kadar ekstrapolasyon uygulanmıştır. Ekstrapolasyonda dikkat edilen bir husus maske görüntüsündeki üst ve alttan 10'ar piksellik güvenli alan payının burada da uygulanarak aynı dikey aralığın kullanılması olmuştur. Şekil 0.49'da bu güvenli alan payları kırmızı alanlarla görülmektedir.



Şekil 0.49 Üst ve alttan 10'ar piksel bırakılan güvenli alan payları

Ekstrapolasyon sürecinde, Şekil 0.50'de gösterildiği gibi dikey eksene yakın noktalar $(\text{Yakın}_x,\text{Yakın}_y)$ koordinatları ile , dikey eksene uzak noktalar ise $(\text{Uzak}_x,\text{Uzak}_y)$ koordinatları ile ifade edilmiştir. (Görüntü koordinat sistemi ve matrisleme indeksi arasındaki farktan dolayı uzak noktaların koordinatları yakın noktaların koordinatlarından daha küçüktür [7].)



Şekil 0.50 Ekstrapolasyon işleminde yakın ve uzak noktalar

Çizgilerin $(\text{Yakın}_x,\text{Yakın}_y)$ ile $(\text{Uzak}_x,\text{Uzak}_y)$ noktaları arasında ekstrapolasyonu için sırasıyla aşağıdaki adımlar uygulanmıştır:

- **Eğim değeri(m):** Her bir doğru için eğim değeri (m) Eşitlik (3.12) kullanılarak hesaplanmıştır.

$$m=\frac{(y_2-y_1)}{(x_2-x_1)} \tag{3.12}$$

- **Dikey ekseni kesişim noktası değeri (b):** Her bir doğru için dikey ekseni kesişim noktası değeri (b) Eşitlik (3.14) kullanılarak hesaplanmıştır.

$$y_1=mx_1+b \tag{3.13}$$

$$b=y_1-mx_1 \tag{3.14}$$

- **Yakın$_y$ koordinatı:** Her bir doğrunun Yakın$_y$ koordinatı için görüntünün üst kısmından itibaren 10 piksellik bir güvenlik marjı belirlenmiştir. Bu doğrultuda, doğrunun Yakın$_y$ koordinatı, Eşitlik (3.15) kullanılarak hesaplanmıştır:

$$\text{Yakın}_y=\text{Görüntü\_Yükseklik}-10=410 \tag{3.15}$$

- **Yakın$_x$ koordinatı:** Her bir doğru için hesaplanan Yakın$_y$ koordinat değerleri, Eşitlik (3.16) aracılığıyla Yakın$_x$ koordinatının hesaplanmasında kullanılmıştır.

$$\text{Yakın}_x=\text{round}\left(\frac{\text{Yakın}_y-b}{m}\right) \tag{3.16}$$

- **Uzak$_y$ koordinatı:** Her bir Uzak$_y$ koordinatı için görüntünün alt kısmından itibaren 10 piksellik bir güvenlik marjı belirlenmiştir. Bu doğrultuda, doğrunun Uzak$_y$ koordinatı Eşitlik (3.17) kullanılarak hesaplanmıştır:

$$\text{Uzak}_y=10 \tag{3.17}$$

- **Uzak$_x$ koordinatı:** Her bir doğru için hesaplanan Uzak$_y$ koordinat değerleri, Eşitlik (3.18) aracılığıyla Uzak$_x$ koordinatının hesaplanmasında kullanılmıştır.

$$\text{Uzak}_x=\text{round}\left(\frac{\text{Uzak}_y-b}{m}\right) \tag{3.18}$$

Ekstrapolasyonu tamamlanan doğrular açısal ve koordinatsal filtrelemeye tabi tutulmuştur.

1.18.2 Eğimsel gruplandırma kuralının oluşturulması

Karayolu ulaşım altyapısında, sol ve sağ yol çizgileri belirli açısal değer aralıklarında yer almaktadır. Bu nedenle, elde edilen doğrusal verilerden söz konusu açısal değer aralıklarının dışında kalanların filtrelenmesi gerekmektedir. Sağ ve sol yol çizgilerinin tespiti amacıyla Hough dönüşümü sırasında Şekil 0.43 ve Şekil 0.44'de görüldüğü gibi belirli açı aralığı kullanılmıştır. Kullanılan açı aralıkları Tablo 0.17'da listelenmiştir.

Tablo 0.17 Sol ve Sağ yol çizgileri için θ°_{min} ve θ°_{max}değerleri

Yol Çizgisi	$\boldsymbol{\theta^{\circ}_{min}}$	$\boldsymbol{\theta^{\circ}_{max}}$
Sol	30°	80°
Sağ	-80°	-30°

Tablo 0.17'da sunulan açı değerleri derece(°) cinsinden ifade edilmiştir. Eğim aralıklarının hesaplanabilmesi için, bu değerlerin Eşitlik (3.19)'da yer alan formül kullanılarak dereceden radyana dönüştürülmesi gerekmektedir.

$$\theta^{Radyan}=\theta^{\circ}\left(\frac{\pi}{180}\right) \tag{3.19}$$

Eşitlik (3.19) kullanılarak, Tablo 0.17'da belirlenen açı değerleri derece biriminden radyan birimine dönüştürülmüştür. Radyan cinsinden elde edilen açı aralıkları Tablo 0.18'de listelenmiştir.

Tablo 0.18 Sol ve Sağ yol çizgileri için $\theta^{\text{Radyan}}_{\text{max}}$ ve $\theta^{\text{Radyan}}_{\text{max}}$ değerleri

Yol Çizgisi	$\boldsymbol{\theta^{Radyan}_{min}}$	$\boldsymbol{\theta^{Radyan}_{max}}$
Sol	0.5236	1.3963
Sağ	-1.3963	-0.5236

Yol çizgilerinin alabileceği açı değerleri radyan birimine dönüştürüldükten sonra ilgili radyan açı değerleri için yatay ekseni ile yapılan eğim değeri (m), Eşitlik (3.20) kullanılarak hesaplanmıştır.

$$m=\tan\left(\theta^{Radyan}\right) \tag{3.20}$$

Radyan biriminde hesaplanan açı aralıkları, Eşitlik (3.20) kullanılarak yatay eksene göre eğim (m) aralıklarına dönüştürülmüştür. Elde edilen yatay eksenle eğim aralıkları Tablo 0.19'de listelenmiştir.

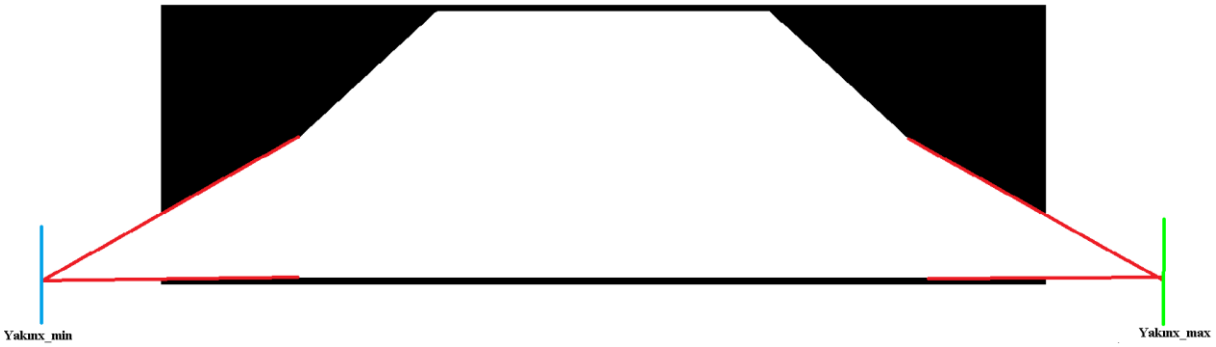
Tablo 0.19 Sol ve sağ yol çizgileri için yatay eksenle eğim aralıkları

Yol Çizgisi	**m_{min}**	**m_{max}**
Sol	0.5774	5.6713
Sağ	-5.6713	-0.5774

Bu çalışmada, yol çizgisi takibi hesaplamalarında kullanılacak doğruların eğim değerlerinin Tablo 0.19'de belirtilen aralıklarında olması gerekmektedir. Bu aralıklar dışında kalan doğrular, Doğruların eğimsel ve koordinatsal olarak gruplandırılması adımında elenecektir.

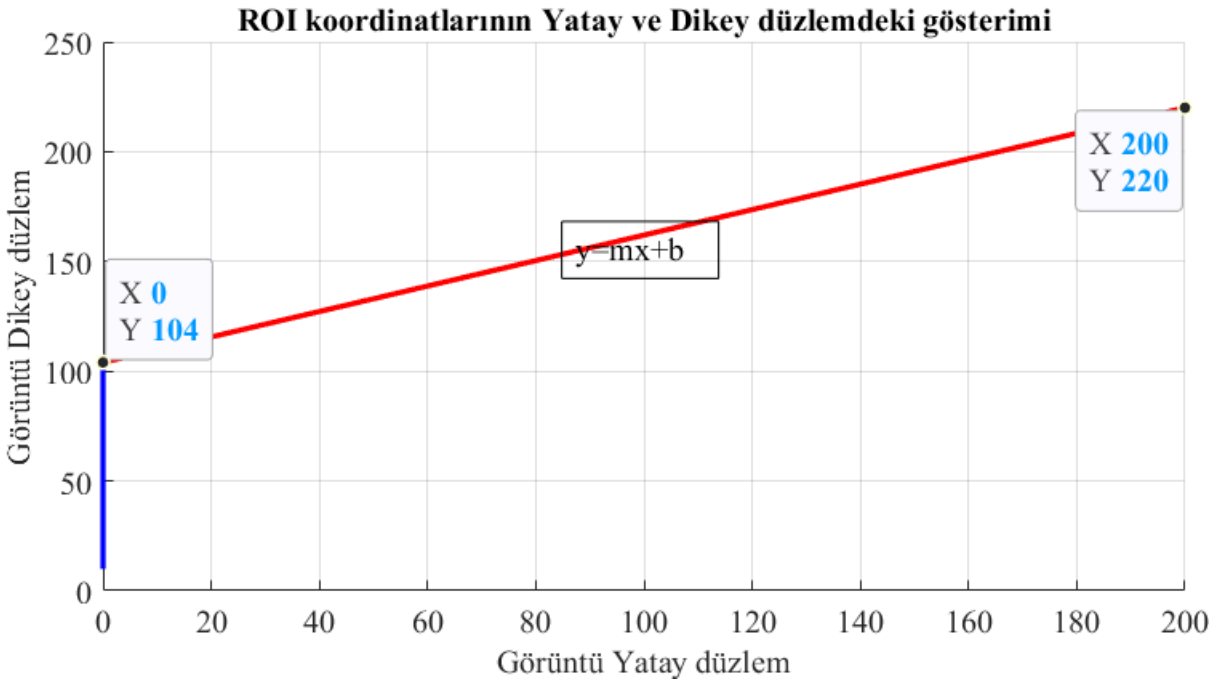
1.18.3 Koordinatsal gruplandırma kuralının oluşturulması

Maskeleme görüntüsü incelendiğinde, doğruların ekstrapolasyon sonucunun yatay koordinatlarının Şekil 0.51'de olduğu gibi yatay eksende görüntü dışına taşabildiği gözlemlenmiştir. Bu taşmadan dolayı, yakın eksendeki yatay koordinat değerinin $Yakın_{x_{min}}$ ile $Yakın_{x_{max}}$ aralığında olabileceği tespit edilmiştir. Yol çizgilerinin $Yakın_x$ değerlerinin bu aralığın dışına çıkmaması gerekmektedir.



Şekil 0.51 Ekstrapolasyon sonucunun görüntü dışına çıkması

$Yakın_{x_{min}}$ ve $Yakın_{x_{max}}$ değerlerinin tespiti için temel doğru denkleminden faydalanılmıştır. Temel doğru denklemi, Eşitlik (3.13) ile ifade edilmektedir. Bu denklemde m ve b değerlerinin hesaplanabilmesi için minimum iki koordinat bilgisi gerekmektedir. $Yakın_{x_{min}}$ noktasının belirlenmesinde, Şekil 0.52'de gösterilen kırmızı doğrunun geçtiği (0,104) ve (200,220) noktaları kullanılmıştır.



Şekil 0.52 Doğru denklemi için kullanılan noktaların grafiği

- **Kırmızı doğru eğim değeri(*m*):** İlgili doğrunun eğim değerinin tespit edilebilmesi amacıyla, belirlenen noktalar doğru denklemine yerleştirilmiştir. Hesaplamalar sırasında (x_1,y_1) ve (x_2,y_2) noktalarına Eşitlik (3.21) ve Eşitlik (3.22)'de belirtilen değerler atanmıştır.

$$(x_1,y_1)=(0,104) \tag{3.21}$$

$$(x_2,y_2)=(200,220) \tag{3.22}$$

Temel doğru denklemine (x_1,y_1) ve (x_2,y_2) noktaları yerleştirilmiştir.

$$y_2 = mx_2 + b \tag{3.23}$$

$$y_1 = mx_1 + b \tag{3.24}$$

Denklemlerdeki b bilinmeyeninin etkisini ortadan kaldırmak amacıyla, iki doğru denklemi Eşitlik (3.25)'te gösterildiği gibi birbirinden çıkarılmıştır:

$$(y_2-y_1)=m(x_2-x_1)+b-b \tag{3.25}$$

$$(y_2-y_1)=m(x_2-x_1) \tag{3.26}$$

Denklemlerdeki m değeri, Eşitlik (3.27)'de görüldüğü üzere 0.58 olarak hesaplanmıştır:

$$m=\frac{(y_2-y_1)}{(x_2-x_1)}=\frac{(220-104)}{(200-0)}=0.58 \quad (3.27)$$

- **Kırmızı doğru dikey eksen kesişim noktası değeri(*b*):** Elde edilen eğim değeri (*m*) ve (0,104) noktası kullanılarak, doğru denkleminde doğrunun dikey ekseni kestiği nokta (*b*), Eşitlik (3.28), Eşitlik (3.29) ve Eşitlik (3.30) aracılığıyla hesaplanmıştır. Dikey eksen kesişim noktası değeri *(b)* Eşitlik (3.30)'da görüldüğü üzere 104 olarak hesaplanmıştır.

$$y_1=mx_1+b \quad (3.28)$$

$$104=m0+b \quad (3.29)$$

$$b=104 \quad (3.30)$$

- **Yakın$_{x_{min}}$ koordinatı:** Doğru denklemi elde edildikten sonra, y=10 koordinatına karşılık gelen Yakın$_{x_{min}}$ noktası Eşitlik (3.31), Eşitlik (3.32) ve Eşitlik (3.33) kullanılarak hesaplanmıştır.

$$10=0.58x+104 \quad (3.31)$$

$$0.58x=10-104 \quad (3.32)$$

$$0.58x=\frac{-94}{0.58}\cong -162 \quad (3.33)$$

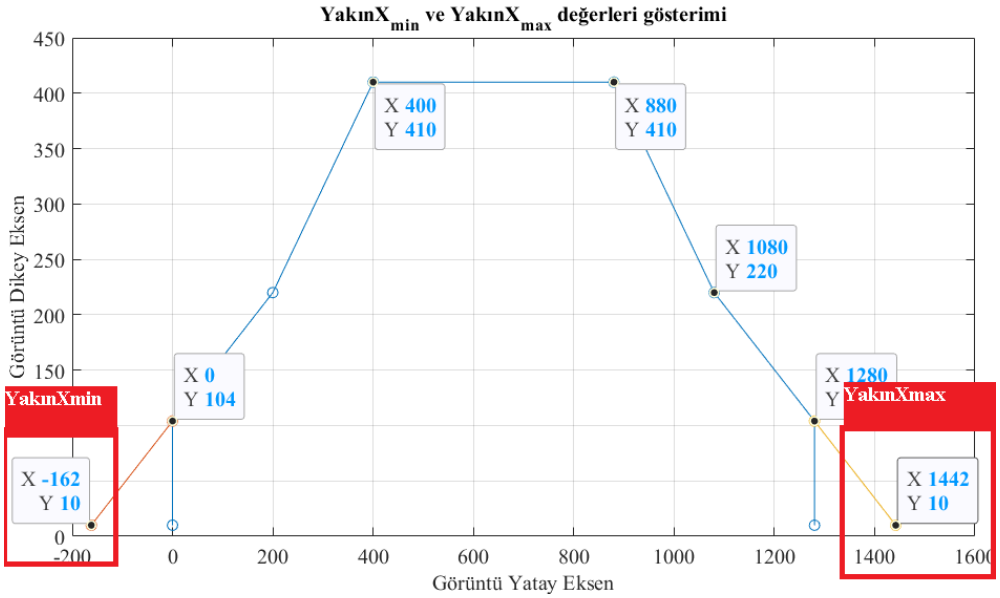
Yakın$_{x_{min}}$ koordinatı, Eşitlik (3.34)'te görüldüğü gibi -162 olarak hesaplanmıştır.

$$\text{Yakın}_{x_{min}}=-162 \quad (3.34)$$

- **Yakın$_{x_{max}}$ koordinatı:** Yakın$_{x_{max}}$ koordinatı, Eşitlik (3.35) kullanılarak 1442 olarak hesaplanmıştır.

$$\text{Yakın}_{x_{max}}=\text{Görüntü Genişlik}+\lfloor\text{Yakın}_{x_{min}}\rfloor=(1280+\lfloor -162\rfloor)=1442 \quad (3.35)$$

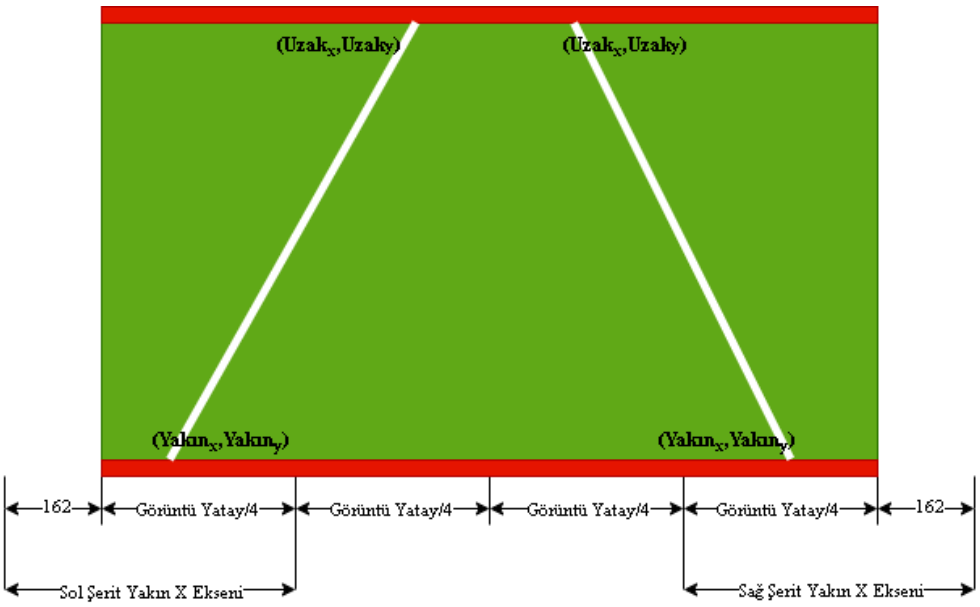
Sol ve sağ yol çizgisi için elde edilen $Yakın_{x_{min}}$ ve $Yakın_{x_{max}}$ koordinatları, Doğruların eğimsel ve koordinatsal olarak gruplandırılması adımında kullanılacaktır. Elde edilen koordinatların maskeleme görüntüsü koordinatları üzerinde gösterimi Şekil 0.53'teki gibidir.



Şekil 0.53 $Yakın_{x_{min}}$ ve $Yakın_{x_{max}}$ değerleri gösterimi

Yapılan çalışmalarda ekstrapolasyon uygulanmış yol çizgisi bilgilerinin dikey eksende en yakın ve en uzak noktalarının yatay eksende belirli bir aralıkta olabileceği görülmüştür.

Sol ve sağ yol çizgilerinin $Yakın_x$ değerleri belirli bir aralıkta olabilmektedir. Bunun için görüntünün yatay ekseni dört eşit parçaya bölünmüştür. Yol çizgilerinin dikey eksende en yakın noktaları için yatay eksende alabileceği değerler Şekil 0.54'te gösterilmiştir.



Şekil 0.54 Dikey eksende en yakın noktalar için kullanılan aralıklar

Şekil 0.54'e göre yol çizgilerinin $Yakın_x$ noktalarının alabileceği değerler aşağıdaki adımlar doğrultusunda hesaplanmıştır:

- **Sol yol çizgisi minimum $Yakın_x$ değeri:** Eşitlik (3.36)'da görüldüğü üzere -162 olarak hesaplanmıştır.

$$Sol\_Yakın_{x_{min}} = Yakın_{x_{min}} = -162 \tag{3.36}$$

- **Sol yol çizgisi maksimum $Yakın_x$ değeri:** Eşitlik (3.37) kullanılarak 320 olarak hesaplanmıştır.

$$Sol\_Yakın_{x_{max}} = \left(\frac{GörüntüGenişlik}{4}\right) = 320 \tag{3.37}$$

- **Sağ yol çizgisi minimum $Yakın_x$ değeri:** Eşitlik (3.38) kullanılarak 960 olarak hesaplanmıştır.

$$Sağ\_Yakın_{x_{min}} = \left(\frac{GörüntüGenişlik}{4}\right) x3 = 960 \tag{3.38}$$

- **Sağ yol çizgisi maksimum $Yakın_x$ değeri:** Eşitlik (3.39) kullanılarak 1442 olarak hesaplanmıştır.

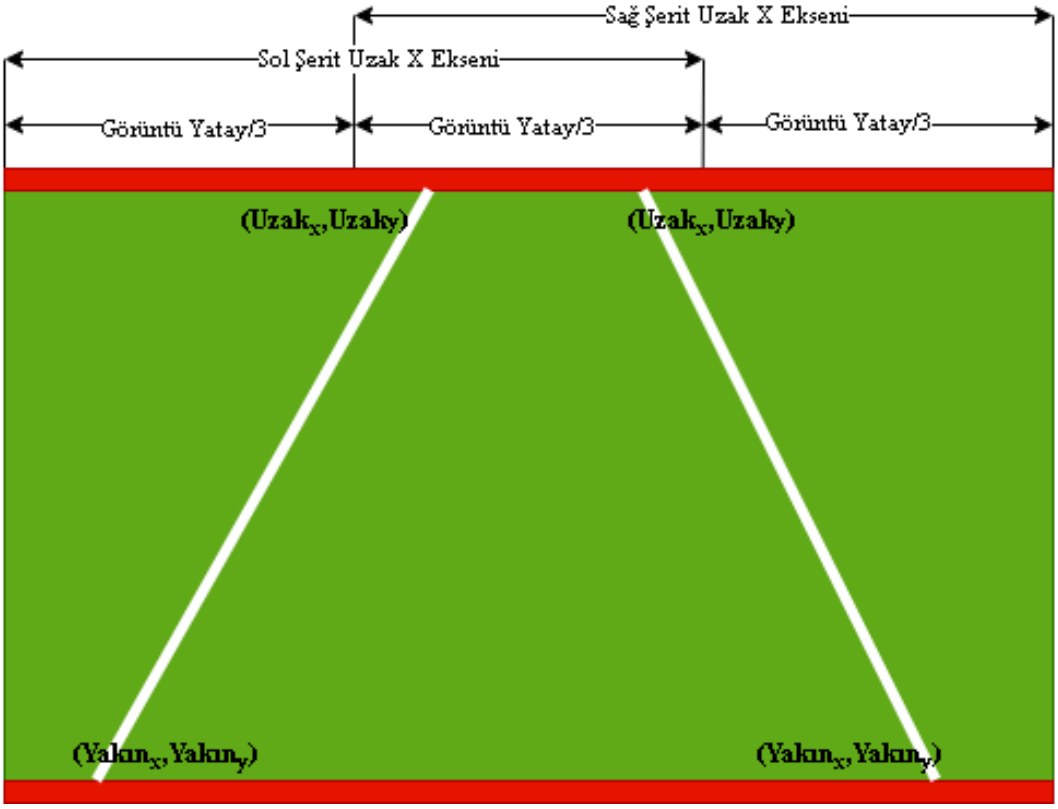
$$Sağ\_Yakın_{x_{max}} = Yakın_{x_{max}} = 1442 \tag{3.39}$$

Sol ve sağ yol çizgisi için elde edilen $Yakın_{x_{min}}$ ve $Yakın_{x_{max}}$ değerleri Tablo 0.20'de gösterilmiştir.

Tablo 0.20 Sol ve sağ yol çizgisi için elde edilen $Yakın_{x_{min}}$ ve $Yakın_{x_{max}}$ değerleri tablosu

Yol Çizgisi	**$Yakın_{x_{min}}$**	**$Yakın_{x_{max}}$**
Sol	-162	320
Sağ	960	1442

Sol ve sağ yol çizgileri için $Uzak_x$ koordinatları belirli bir aralıkta olabilmektedir. Bunun için görüntünün yatay ekseni üç eşit parçaya bölünmüştür. Yol çizgilerinin $Uzak_x$ koordinatlarının yatay eksende alabileceği değerler Şekil 0.55'da görselleştirilmiştir.



Şekil 0.55 Dikey eksende en uzak noktalar için kullanılan aralıklar

Şekil 0.55'a göre yol çizgilerinin $Uzak_x$ noktalarının alabileceği değerler aşağıdaki adımlar doğrultusunda hesaplanmıştır:

- **Sol yol çizgisi minimum $Uzak_x$ değeri:** Eşitlik (3.40) kullanılarak 0 olarak hesaplanmıştır.

$$Sol\_Uzak_{x_{min}} = 0 \tag{3.40}$$

- **Sol yol çizgisi maksimum $Uzak_x$ değeri:** Eşitlik (3.41) kullanılarak 853 olarak hesaplanmıştır.

$$Sol\_Uzak_{x_{max}} = 2\left(\frac{GörüntüGenişlik}{3}\right) \cong 853 \tag{3.41}$$

- **Sağ yol çizgisi minimum $\mathbf{Uzak_x}$ değeri:** Eşitlik (3.42) kullanılarak 427 olarak hesaplanmıştır.

$$\text{Sağ\_Uzak}_{x_{min}} = \left(\frac{GörüntüGenişlik}{3}\right) \cong 427 \tag{3.42}$$

- **Sağ yol çizgisi maksimum $\mathbf{Uzak_x}$ değeri:** Eşitlik (3.43) kullanılarak 1280 olarak hesaplanmıştır.

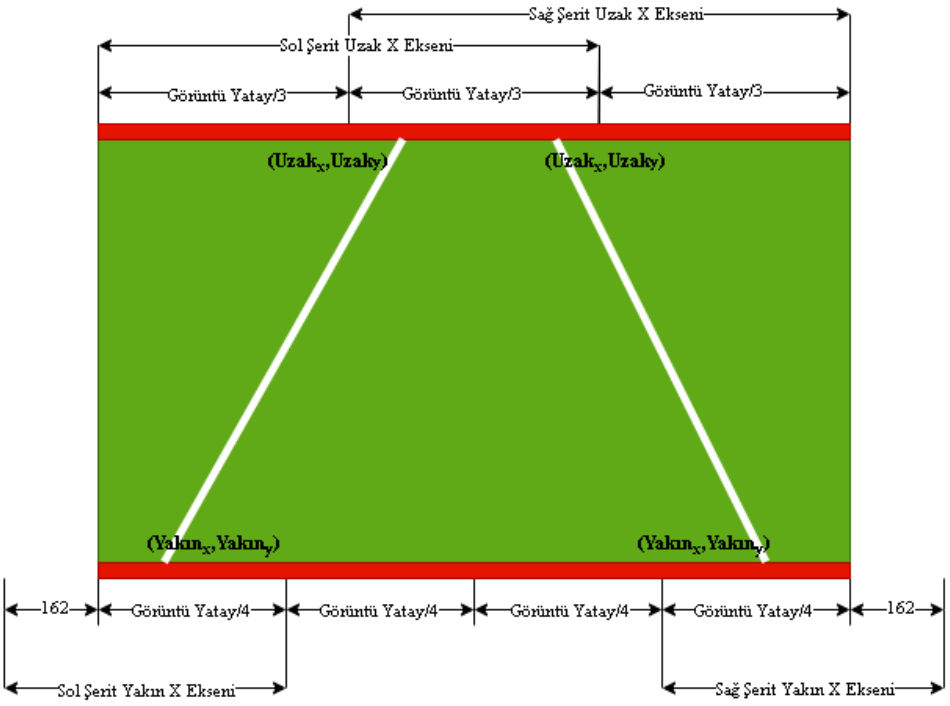
$$\text{Sağ\_Uzak}_{x_{max}} = 1280 \tag{3.43}$$

Sol ve sağ yol çizgileri için $Uzak_{x_{min}}$ ve $Uzak_{x_{max}}$ değerleri Tablo 0.21'da gösterilmiştir.

Tablo 0.21 Sol ve sağ yol çizgisi için $Uzak_{x_{min}}$ ve $Uzak_{x_{max}}$ değerleri tablosu

Yol Çizgisi	**$\mathbf{Uzak_{x_{min}}}$**	**$\mathbf{Uzak_{x_{max}}}$**
Sol	0	853
Sağ	427	1280

Sonuç olarak dikey eksen uzak ve yakın noktalar için kullanılan koordinat aralıkları Şekil 0.56'da görülmektedir.



Şekil 0.56 Dikey eksen uzak ve yakın noktalar için kullanılan koordinat aralığı kuralı

Ekstrapolasyon uygulanmış çizgilerin Şekil 0.56'ya göre filtrelenebilmesi için hesaplanan Yakın_x ve Uzak_x yatay eksen koordinat aralıkları Tablo 0.22'de listelenmiştir.

Tablo 0.22 Dikey yakın ve dikey uzak noktalar için yatay eksen aralıkları

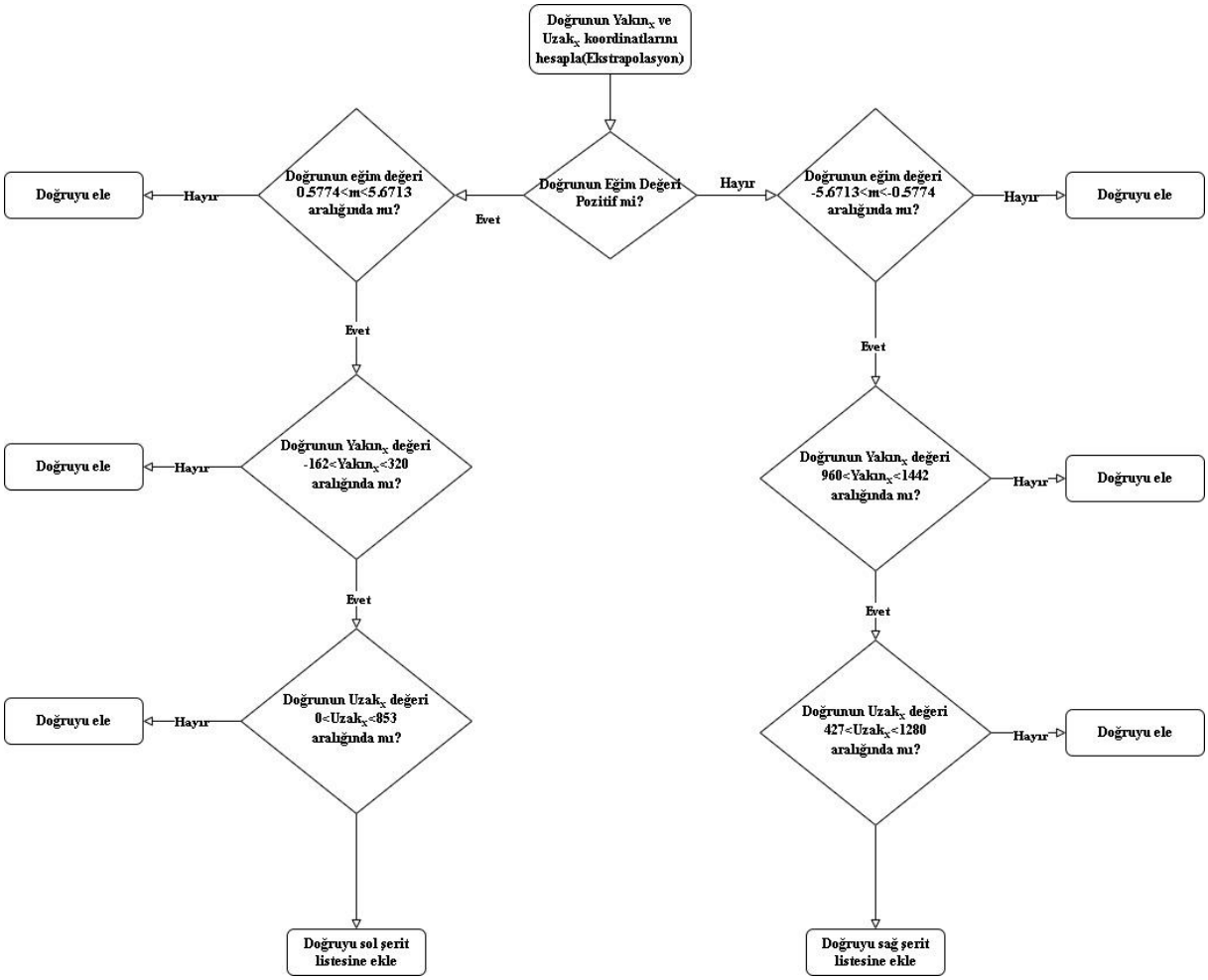
Yol Çizgisi	**Dikey Yakın**		**Dikey Uzak**	
	$\text{Yakın}_{x_{min}}$	$\text{Yakın}_{x_{max}}$	$\text{Uzak}_{x_{min}}$	$\text{Uzak}_{x_{max}}$
Sol	-162	320	0	853
Sağ	960	1442	427	1280

1.18.4 Doğruların eğimsel ve koordinatsal olarak gruplandırılması

Doğruların ekstrapolasyonu aşamasında, elde edilen doğrular eğim ve koordinat değerlerine göre gruplandırılmıştır. Bu gruplandırma işlemi, sol ve sağ yol çizgisi adaylarının belirlenmesini ve ilerleyen aşamalarda ağırlıklı ortalama hesaplamalarının doğru şekilde yapılmasını amaçlar. Gruplandırma adımları aşağıda listelenmiştir:

- **Pozitif Eğimli Doğrular (Sol Yol Çizgisi Adayları):**
  - Eğim değerinin (m) 0.5774 ile 5.6713 arasında olup olmadığı kontrol edilmiştir. Bu aralık dışındaki doğrular elenmiştir.
  - Kalan doğruların Yakın_x koordinatının -162 ile 320 arasında olup olmadığı kontrol edilmiştir. Bu aralık dışındaki doğrular elenmiştir.
  - Son olarak, kalan doğruların Uzak_x koordinatının 0 ile 853 arasında olup olmadığı kontrol edilmiştir. Bu aralık dışındaki doğrular elenmiştir.
- **Negatif Eğimli Doğrular (Sağ Yol Çizgisi Adayları):**
  - Eğim değerinin (m) -5.6713 ile -0.5774 arasında olup olmadığı kontrol edilmiştir. Bu aralık dışındaki doğrular elenmiştir.
  - Kalan doğruların Yakın_x koordinatının 960 ile 1442 arasında olup olmadığı kontrol edilmiştir. Bu aralık dışındaki doğrular elenmiştir.
  - Son olarak, kalan doğruların Uzak_x koordinatının 427 ile 1280 arasında olup olmadığı kontrol edilmiştir. Bu aralık dışındaki doğrular elenmiştir.

Bu adımların algoritmik akışını gösteren diyagram Şekil 0.57'de sunulmuştur.



Şekil 0.57 Çizgilerin gruplandırılması akış diyagramı

Yol çizgisi gruplandırma işlemi sonucunda, sol ve sağ yol çizgilerine ait doğruların yer aldığı iki ayrı liste elde edilmiştir. Her bir listenin en az bir doğru içermesi gerekmektedir. Aksi takdirde, sol ve/veya sağ yol çizgisi tespit edilememiş demektir. Bu durumda, yatay filtre parametreleri güncellenerek, Yatay filtrenin uygulanması adımı ve takip eden adımlar, her iki listede de en az bir doğru bulunana kadar tekrarlanmıştır.

1.19 Yatay Filtre Komşuluk Değerinin Güncellenmesi

Doğruların sol ve sağ yol çizgisi olarak gruplandırılması sonucunda, sol veya sağ yol çizgisi listesinde eleman bulunmaması durumunda ilgili yol çizgisinin algılanamadığı sonucuna varılır. Bu durumda, Eşitlik (3.44) kullanılarak yatay filtre komşuluk değeri güncellenir:

$$\tau_{n+1}=\tau_n*2 \tag{3.44}$$

Eşitlik (3.44)'te yer alan "n" değeri, döngünün kaçıncı iterasyonunda olunduğunu ifade eder. Eğer "n" değeri 5'e ulaşırsa, yol çizgilerinin bulunamadığı kabul edilir ve döngü sonlandırılır.

Güncellenen yatay komşuluk değeri ile algoritma, Yatay filtrenin uygulanması aşamasından itibaren tekrar başlatılır.

1.20 Sol Ve Sağ Yol Çizgilerinin Tekilleştirilmesi

Sağ ve sol yol çizgisi listelerinin elde edilmesinin ardından, özellikle kesintili yol çizgilerinde sıklıkla karşılaşıldığı gibi, her iki yol çizgisi için birden fazla aday doğru ortaya çıkabilmektedir. Bu durum, birden fazla eğim ve y ekseni kesişim noktası anlamına gelir. Yol çizgilerinin Eşitlik (3.13)'te belirtilen temel doğru modeline uygun hareket ettiği göz önünde bulundurularak, sol ve sağ yol çizgisi için tek bir eğim (m) ve y ekseni kesişim noktası (b) değeri elde edilmesi amaçlanmıştır. Bu işlem, iki temel adımda gerçekleştirilir:

- Yol çizgisi dışı doğruların elenmesi
- Sol ve sağ yol çizgilerinin kendi içlerinde ağırlıklı ortalamalarının hesaplanması

1.20.1 Yol çizgisi dışı doğruların elenmesi

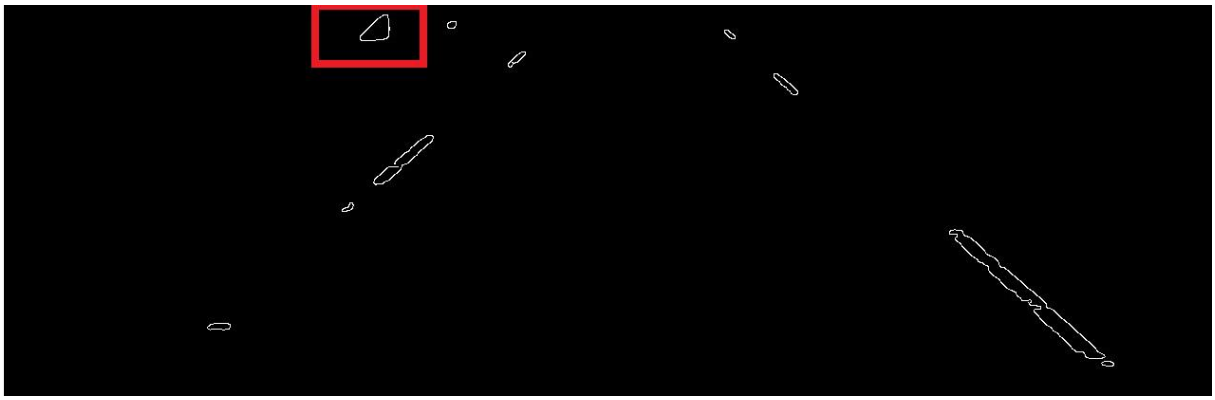
Doğruların gruplandırılması aşamasında, eğimsel ve koordinatsal filtrelemelerden geçen doğrular, zaman zaman sol ve sağ şeride ait olmayan doğruları da içerebilir. Bu durumun başlıca nedenleri şunlardır:

- **Maskeleme Alanı Dışındaki Yol Çizgilerinin Etkisi:**

Sol ve sağ yol çizgisinin dışındaki yol çizgileri, eğer maskeleme alanı içerisine girerlerse, doğru oluşturabilirler. Şekil 0.58'de gösterilen yol görüntüsü bu duruma örnek olarak verilebilir. Şekildeki kırmızı kutu içerisine alınan yol çizgisi maskeleme alanı içerisine girmiştir.

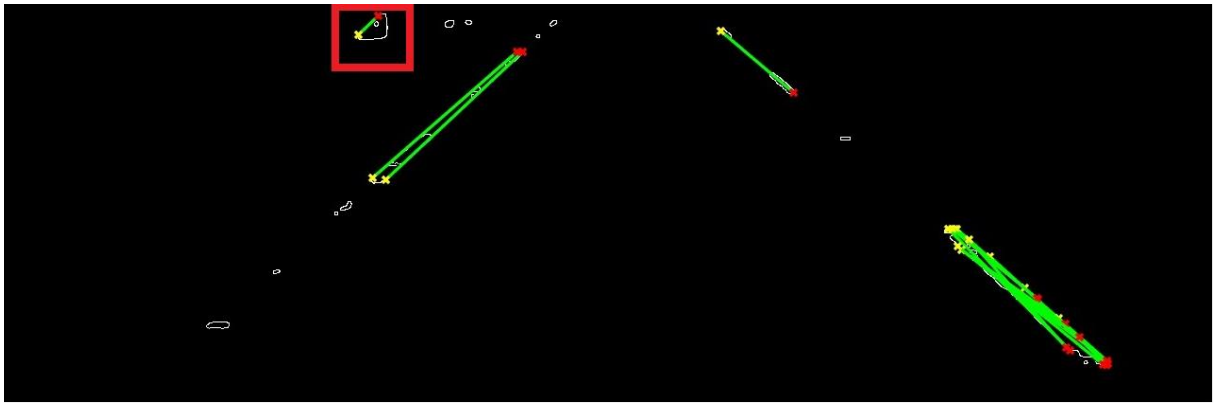
Şekil 0.58 Sol yol çizgisinin dışındaki maskelenmiş alana giren yol çizgisi

Şekil 0.58'de yer alan yol görüntüsü üzerinde gerçekleştirilen kenar algılama işleminin sonucu, Şekil 0.59'da sunulmuştur. Şekilde kırmızı kutu ile işaretlenen alanda, dış yol çizgisi gözlemlenebilmektedir.



Şekil 0.59 Sol yol çizgisinin dışındaki maskelenmiş alana giren kenar algılama sonucu

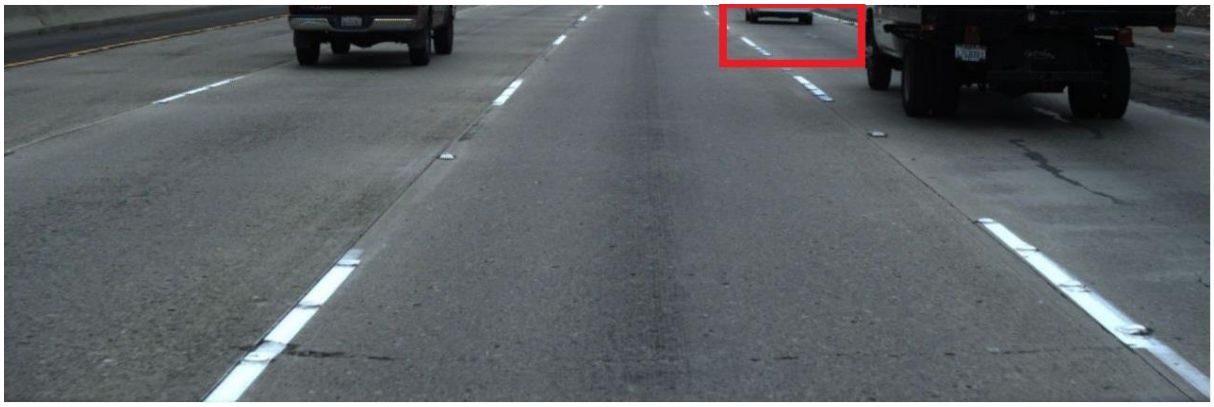
Şekil 0.59'da görülen maskelenmiş alana giren yol çizgisi doğru tespit sonucu Şekil 0.60'da gösterilmiştir. Şekilde kırmızı kutu içerisinde yer alan dış yol çizgisi, filtrelenmesi gereken bir veri olarak öne çıkmaktadır.



Şekil 0.60 Sol yol çizgisinin dışındaki maskelenmiş alana giren doğru tespiti sonucu

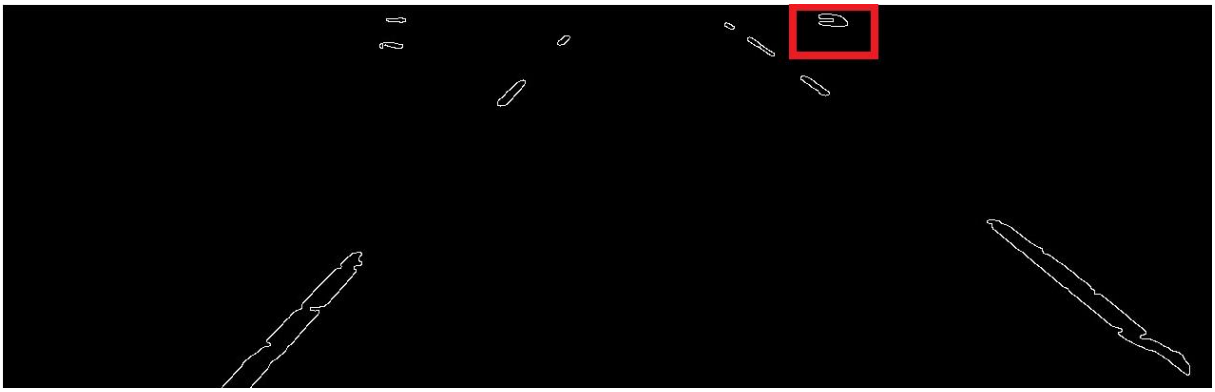
- **Yoldaki araçlar veya diğer etkenler:**

Dış yol çizgilerinin yanı sıra, yoldaki araçlar veya diğer unsurlar da sağ/sol yol çizgileyle ilişkili olmayan doğruların ortaya çıkmasına neden olabilmektedir. Bu doğrular, eğimsel ve koordinatsal gruplandırma filtrelerinden geçebilme potansiyeline sahiptir. Örnek olarak, Şekil 0.61'de kırmızı kutu ile işaretlenen araçtan kaynaklanan durum, bu duruma örnek teşkil etmektedir.



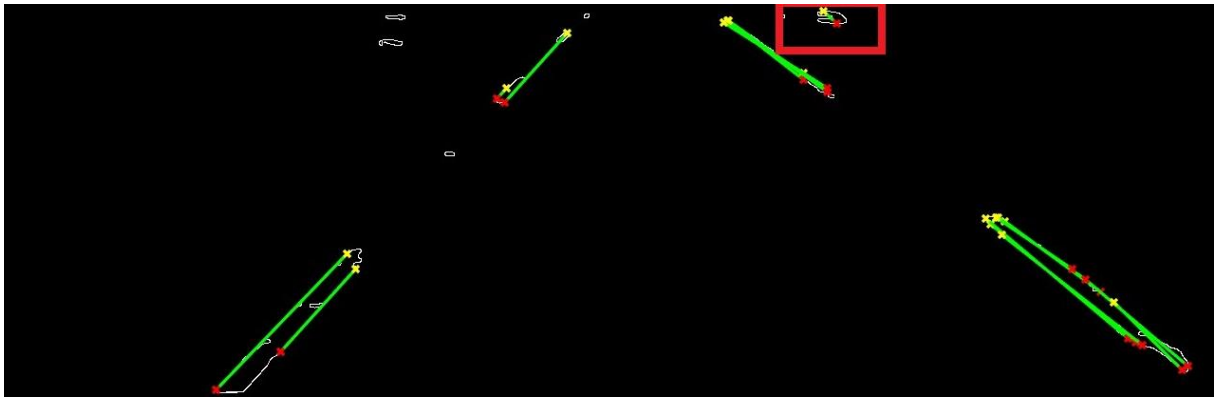
Şekil 0.61 Yanlış doğru oluşturan araç görüntüsü

Şekil 0.61‘de gösterilen yol görüntüsüne kenar algılama uygulandığında elde edilen görüntü Şekil 0.62’de sunulmuştur. Araçtan kaynaklanan istenmeyen kenar görüntüsü kırmızı kutu içerisinde işaretlenmiştir.



Şekil 0.62 Yanlış doğru oluşturan araç görüntüsü kenar algılama sonucu

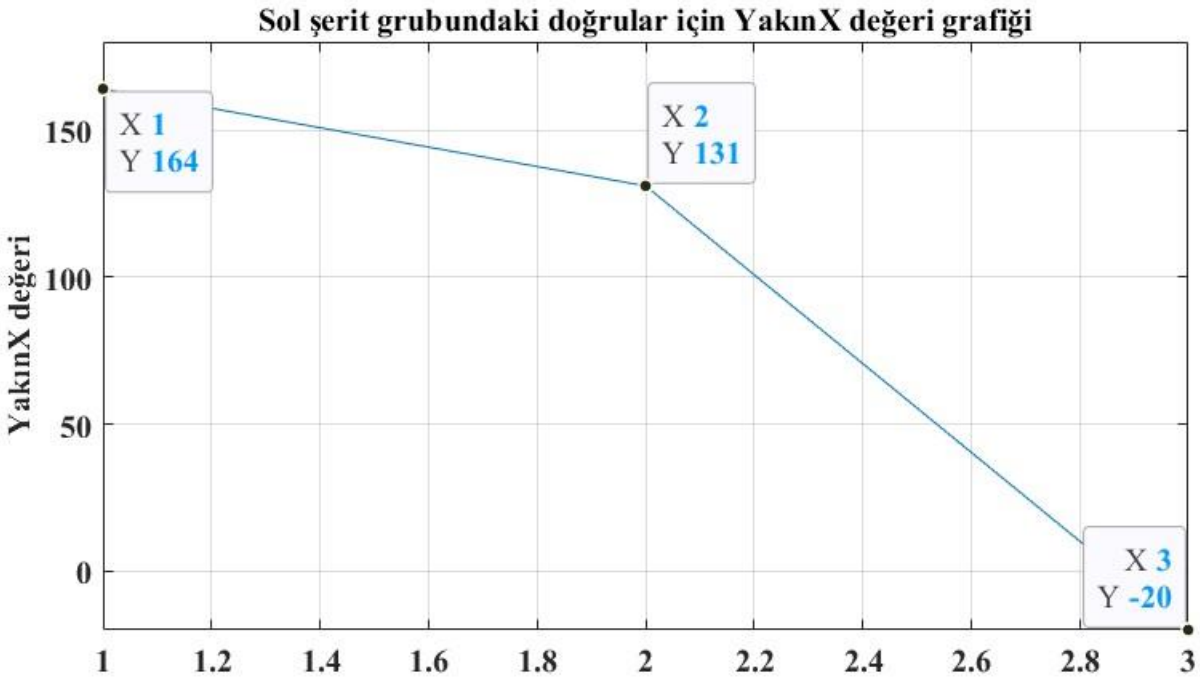
Şekil 0.62’deki kenar algılama sonucuna doğru tespiti uygulandığında Şekil 0.63’te görülen çıktı elde edilmiştir. Şekil 0.63’te kırmızı kutu içerisinde gösterilen doğru, araçtan kaynaklanan ve istenmeyen bir veridir.



Şekil 0.63 Yanlış doğru oluşturan araç görüntüsü doğru tespiti sonucu

Yukarıda bahsedilen nedenlerden dolayı, sol ve sağ yol çizgisinden bağımsız doğruların elenmesi amacıyla bir filtreleme işlemi gerçekleştirilmiştir. Bu filtre, aşağıdaki gözlemlere dayanarak tasarlanmıştır:

- Sol şeride ait doğruların $Yakın_x$ ve $Uzak_x$ koordinat değerleri birbirine yakın olacaktır.
- Sağ şeride ait doğruların $Yakın_x$ ve $Uzak_x$ koordinat değerleri birbirine yakın olacaktır.
- Sol ve sağ şeride ait olmayan doğruların $Yakın_x$ ve/veya $Uzak_x$ koordinat değerleri sol ve sağ yol çizgisi doğrularına yakın olmayacaktır. Örnek olarak Şekil 0.60'daki görüntünün sol yol çizgisi grubu için $Yakın_x$ grafiği çizdirildiğinde Şekil 0.64 elde edilmiştir. Şekilde 3 indeksli noktanın değerinin diğer iki doğrunun(1 ve 2 indeksli doğruların) $Yakın_x$ değerlerinden çok uzak olduğu görülmüştür.



Şekil 0.64 Sol yol çizgisi grubundaki doğrular için $Yakın_x$ değeri grafiği

Bu bağlamda, sol ve sağ yol çizgileri için diğer noktalardan uzak olan $Yakın_x$ ve $Uzak_x$ değerlerinin ayıklanmasına karar verilmiştir. Bu amaçla, Z-skoru yönteminden yararlanılmıştır [37].

Z-skoru, bir veri noktasının dağılım ortalamasından kaç standart sapma uzaklıkta olduğunu gösteren bir ölçüdür ve veri noktalarının normal dağılım içinde ne kadar uçta olduğunu belirlemek için kullanılır. Z-skoru hesabı için Eşitlik (3.45) kullanılmıştır. Bu eşitlik, sol ve sağ yol çizgilerinin $Yakın_x$ *ve* $Uzak_x$ değerleri için ayrı ayrı uygulanmıştır.

$$Z=\frac{(X-\mu)}{\sigma} \tag{3.45}$$

Eşitlik (3.45)'te:

- Z: X noktası için Z-skoru
- X: Z skoru hesaplanacak veri
- μ: X değerlerin ortalaması
- σ: X değerlerin standart sapması
- **$Yakın_x$ değerleri için Z-skoru:** Sol ve sağ yol çizgisi listelerinin her bir elemanı için $Yakın_x$ Z-skoru, Eşitlik (3.46)'ya göre ayrı ayrı hesaplanmıştır.

$$Z_{\text{Yakın}_x} = \frac{(X_{\text{Yakın}_x} - \mu_{\text{Yakın}_x})}{\sigma_{\text{Yakın}_x}} \tag{3.46}$$

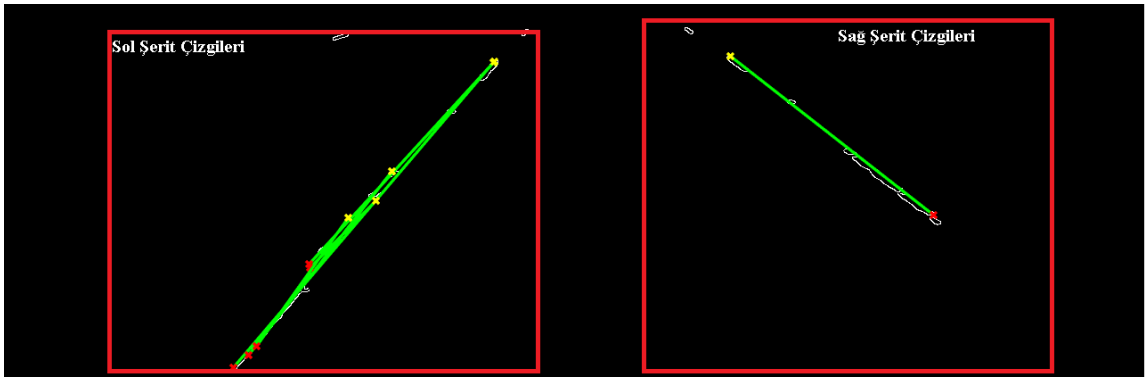
- **Uzak$_x$ değerleri için Z skoru:** Sol ve sağ yol çizgisi listelerinin her bir elemanı için Uzak$_x$ Z skoru, Eşitlik (3.47)'ye göre ayrı ayrı hesaplanmıştır.

$$Z_{\text{Uzak}_x} = \frac{(X_{\text{Uzak}_x} - \mu_{\text{Uzak}_x})}{\sigma_{\text{Uzak}_x}} \tag{3.47}$$

Sol ve sağ yol çizgisi listeleri için ayrı ayrı Z-skoru yöntemi uygulanmıştır. Her bir listedeki doğruların Yakın$_x$ ve Uzak$_x$ değerleri için Z-skorları hesaplanmıştır. Mutlak değeri 1.2'den büyük olan Z-skorlarına sahip noktalar, ilgili yol çizgisi için yol çizgisi dışı noktalar olarak değerlendirilerek elenmiştir.

1.20.2 Sol ve sağ yol çizgilerinin ağırlıklı ortalaması

Yol çizgilerinin gruplandırılması işlemi sonucunda, sol ve sağ yol çizgileri için birden fazla doğru parçası elde edilebilmektedir. Örnek olarak Şekil 0.65'te, sol yol çizgisiyle ilişkili birden fazla doğru parçasının varlığı gözlemlenebilmektedir.



Şekil 0.65 Sol ve sağ yol çizgisi adaylarının gösterimi

Yol çizgisi dışı doğruların elenmesi adımı sonrasında, eğim ve dikey eksen kesim noktaları birbirine yakın değerler göstermektedir. Ancak, sol ve sağ yol çizgileri için temsili birer doğru elde etmek amacıyla, doğrular üzerinde ağırlıklı ortalama işlemi uygulanmıştır. Ağırlıklı ortalama işleminin temel prensibi, daha uzun doğruların hesaplama üzerindeki etkisini artırarak daha doğru bir kestirim elde etmektir.

Bu amaçla, her bir doğrunun uzunluğu Eşitlik (3.48) kullanılarak hesaplanmıştır:

$$L = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3.48}$$

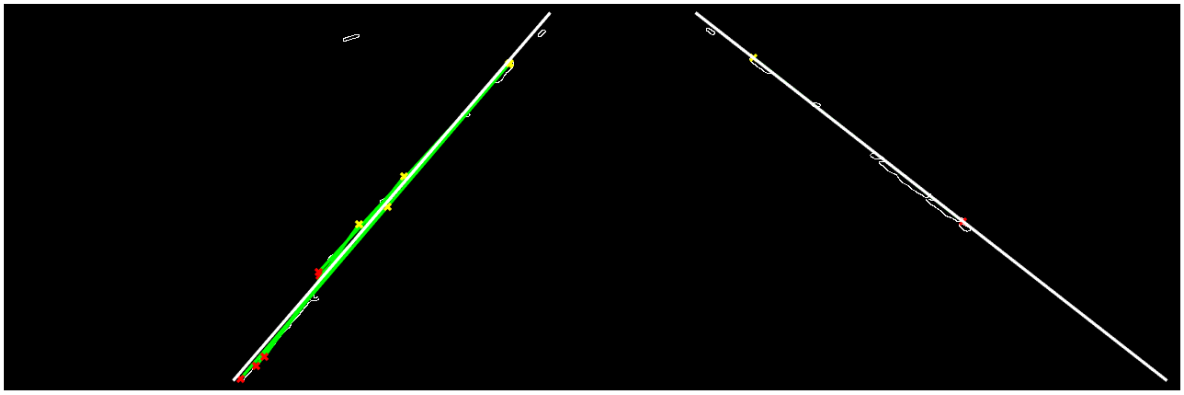
Elde edilen doğru uzunlukları, ağırlıklandırılmış eğim değerinin Eşitlik (3.49) ile hesaplanmasında kullanılmıştır:

$$m_{ortalama} = \frac{\sum_{i=1}^{n} m_i \times L_i}{\sum_{i=1}^{n} L_i} \tag{3.49}$$

Benzer şekilde, ağırlıklandırılmış dikey eksen kesişim noktası değeri de Eşitlik (3.50) ile hesaplanmıştır:

$$b_{ortalama} = \frac{\sum_{i=1}^{n} b_i \times L_i}{\sum_{i=1}^{n} L_i} \tag{3.50}$$

Şekil 0.65'te gösterilen sol ve sağ yol çizgisileri, ağırlıklı ortalama işlemine tabi tutulduğunda elde edilen tekilleştirilmiş sol ve sağ yol çizgisileri Şekil 0.66'da sunulmaktadır. Şekilde yeşil renkli çizgiler ağırlıklı ortalama işleminden önceki sol ve sağ çizgileri, beyaz renkli çizgiler ağırlıklı ortalama sonucunda elde edilen sol ve sağ yol çizgilerini temsil etmektedir.



Şekil 0.66 Tekilleştirilmiş sol ve sağ yol çizgilerinin çıktısı

SONUÇ

1.21 Gerçek Yol Çizgisi Noktalarının Elde Edilmesi

Veri seti parametre dosyasında, yoldaki tüm yol çizgilerine ait bilgiler yer almaktadır. Bu bilgiler, Veri Setindeki Yol Çizgisi Koordinat Bilgileri başlığı altında ayrıntılı olarak açıklanmıştır. Veri setindeki görüntülerin incelenmesi sonucunda, aracın konumuna göre sol yol çizgisinin solunda ve sağ yol çizgisinin sağında da yol çizgilerinin bulunabileceği tespit edilmiştir. Örnek olarak Şekil 0.1'de yer alan yol görüntüsünde birden fazla yol çizgisi gözlemlenmektedir.

Şekil 0.1 Birden Fazla Yol Çizgisi Bilgisi İçeren Yol Görüntüsü

Bu çalışmada, aracın konumuna göre iç kısımda yer alan sol ve sağ yol çizgilerine odaklanılmış olup, dış kısımda bulunan yol çizgileri değerlendirme dışı bırakılmıştır. Bu doğrultuda, veri setindeki yol çizgileri arasından iç kısımda yer alan sol ve sağ yol çizgileri belirlenmiştir. İç yol çizgilerinin belirlenmesi sürecinde, veri setindeki yol çizgilerine ait koordinatlar MATLAB ortamında analiz edilmiştir.

MATLAB programlama ortamında bulunan *"polyfit"* fonksiyonu kullanılarak, her bir yol çizgisiiçin eğim (m) ve dikey eksen kesişim noktaları (b) hesaplanmıştır [38]. Bu fonksiyon, verilen "x" ve "y" koordinatları için n. dereceden bir polinom uydurarak, ilgili yol çizgisinin matematiksel modelini elde etmeyi sağlar. MATLAB'ın "*polyfit*" fonksiyonunun kullanımı, Eşitlik (4.1)'de detaylı olarak açıklanmıştır.

$$p = polyfit(x,y, n); \quad (4.1)$$

MATLAB'ın polyfit fonksiyonunun girdi parametreleri Tablo 0.1'de listelenmiştir.

Tablo 0.1 MATLAB'ın polyfit fonksiyonu girdi parametreleri

Parametre	**Parametre Tanımı**
x	Polinomun X eksenindeki noktaları.
y	Polinomun Y eksenindeki noktaları.
n	Polinomun derecesi

MATLAB'ın polyfit fonksiyonunun çıktı parametreleri Tablo 0.2'de listelenmiştir.

Tablo 0.2 MATLAB'ın polyfit fonksiyonu çıktı parametreleri

Parametre	**Parametre Tanımı**
P	Polinom katsayılarını verir.

Bu çalışmada, veri setinden elde edilen yol çizgilerine ait "x" ve "y" koordinatları kullanılmıştır. Veri analizi sürecinde kullanılan "polyfit" fonksiyonunun girdi parametreleri Tablo 0.3'te listelenmiştir.

Tablo 0.3 Çalışmada kullanılan "polyfit" fonksiyonu girdi parametreleri

Değişken	**Kullanılan Değer**
x	Veri setindeki x ekseni noktaları
y	Veri setindeki y ekseni noktaları
n	1

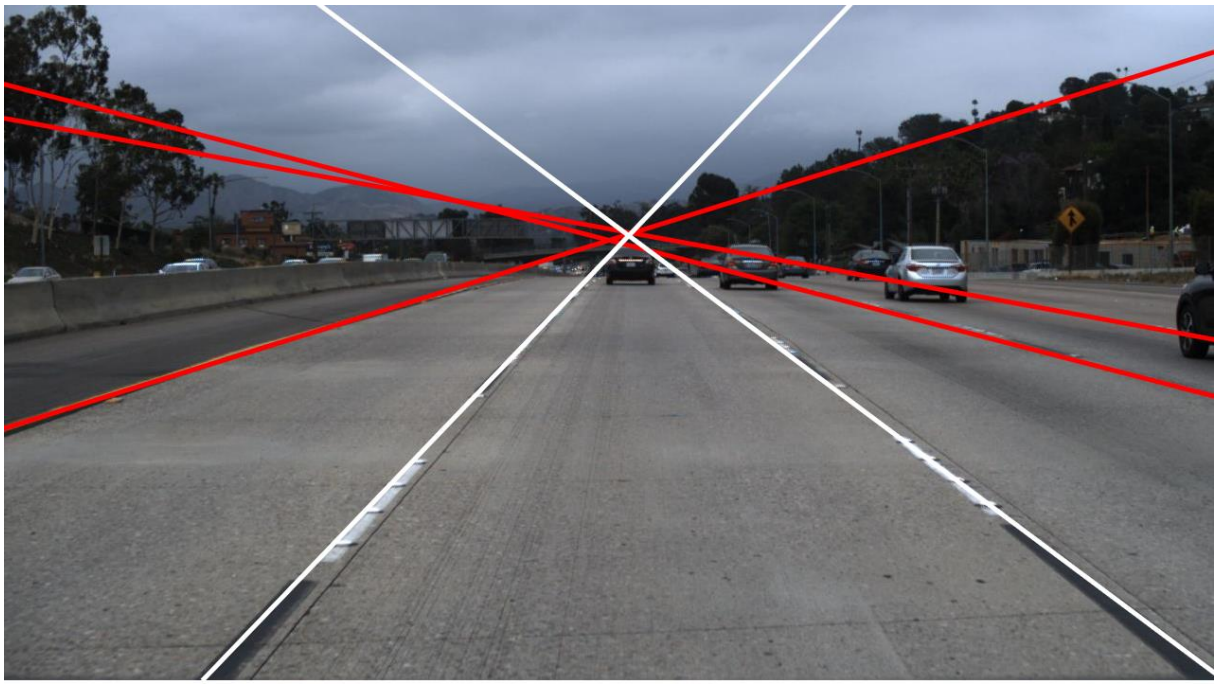
MATLAB'ın "*polyfit"* fonksiyonu birinci dereceden denklem kullanıldığı için temel doğru denklemini oluşturur. Bunun sonucunda eğim*(m)* ve y ekseni kesim noktası*(b)* değerlerini döndürür. Sonuç olarak, her bir doğru için eğim değeri elde edilmiştir.

Her bir doğru eğim değerlerine göre sıralandığında en büyük eğime sahip olan yol çizgisi sol yol çizgisi, en küçük değere sahip olan yol çizgisi sağ yol çizgisi olarak hesaba olduğu tespit edilmiştir. Bu yöntemin sebebi aşağıdaki gibidir:

- Sağ yol çizgileri negatif eğime sahiptir. İç kısımda yer alan sağ yol çizgisi, diğer sağ yol çizgilerine kıyasla daha dik bir açıya sahiptir (Şekil 0.2). Bu nedenle eğim değeri, diğer sağ yol çizgilerinden daha küçüktür (mutlak değer olarak daha büyüktür).

- Sol yol çizgileri pozitif eğime sahiptir. İç kısımda yer alan sol yol çizgisi, diğer sol yol çizgilerine kıyasla daha dik bir açıya sahiptir (Şekil 0.2). Bu nedenle eğim değeri, diğer sol yol çizgilerinden daha büyüktür.
- Şekil 0.2'de görüldüğü gibi, en küçük eğim değerine sahip doğru iç kısımda yer alan sağ şeridi, en büyük eğim değerine sahip doğru ise iç kısımda yer alan sol şeridi temsil etmektedir.

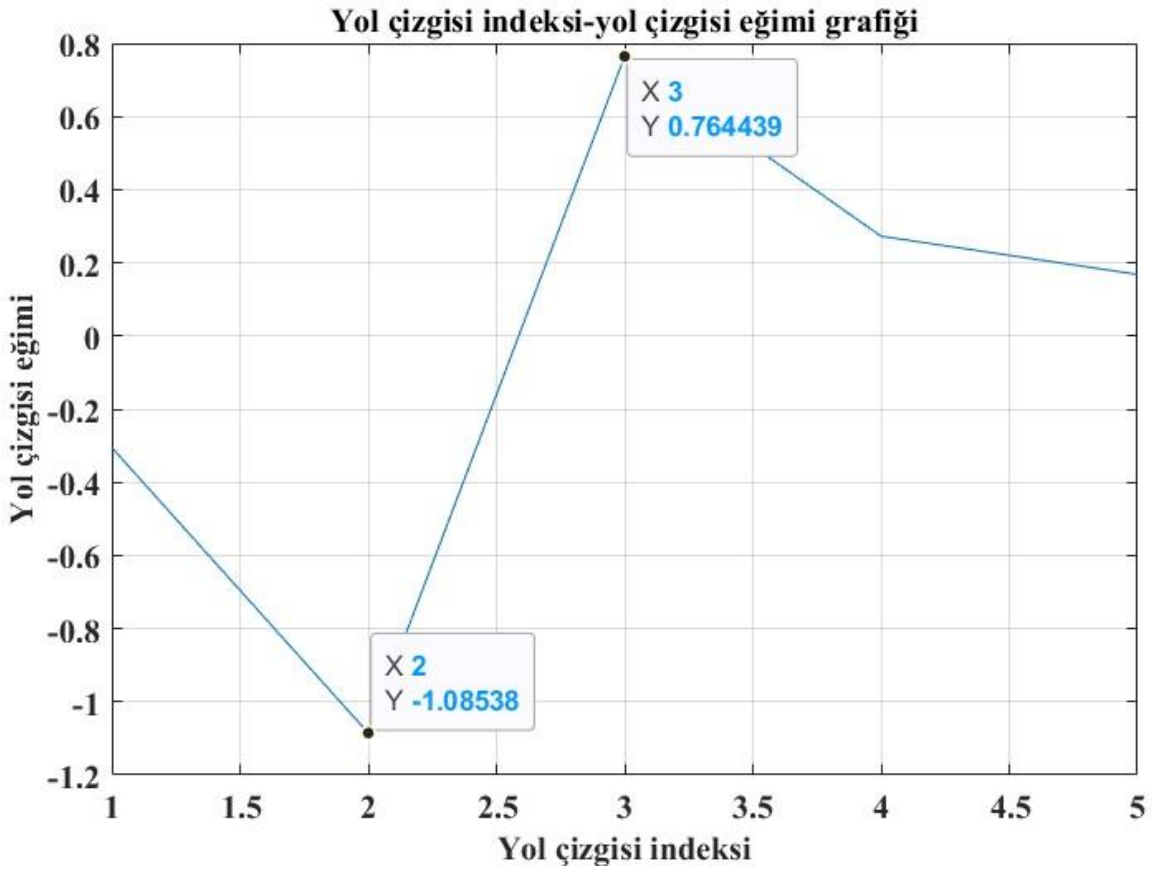
Şekil 0.2'de beyaz renkle çizilen yol çizgileri iç yol çizgileri, kırmızı renkli yol çizgileri ise dış yol çizgileridir.



Şekil 0.2 Yol çizgilerinin yol üzerinde gösterimi

Şekil 0.2'de yol çizgilerinin çizdirildiği yol görüntüsü için yol çizgisi numarası-yol çizgisi eğimi grafiği

Şekil 0.3'te görülebilmektedir. Grafikte yatay eksen, ilgili yol çizgisinin gerçek yol çizgisi noktalarını içeren veri seti dosyasındaki indeksini belirtmektedir. Grafik incelendiğinde, veri seti dosyasındaki 2 indeksli yol çizgisinin en düşük eğime sahip olduğu ve bu nedenle sağ şeridi temsil ettiği görülmektedir. Benzer şekilde, veri seti dosyasındaki 3 indeksli yol çizgisi en yüksek eğime sahip olduğundan sol şeridi ifade etmektedir.

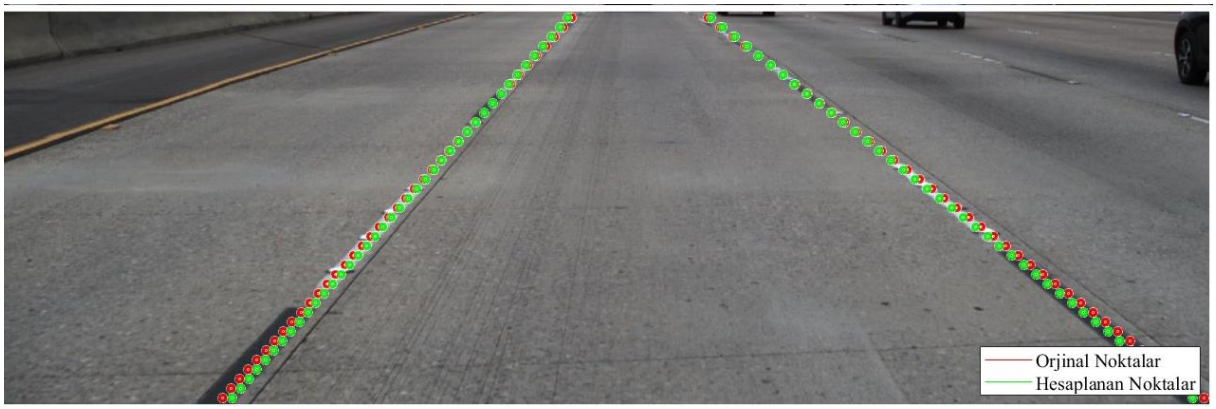


Şekil 0.3 Yol çizgisi numarası-yol çizgisi eğimi grafiği

Sağ ve sol yol çizgilerinin belirlenmesinin ardından, ROI (İlgi Alanı) boyunca temel doğru denklemi kullanılarak hem gerçek hem de hesaplanan noktalar için sol ve sağ yol çizgilerine ait birer doğru elde edilmiştir. Bu doğrulardan elde edilen noktalar, algoritma çıktıları ile karşılaştırma amacıyla kullanılacaktır.

1.22 Algoritma Başarısının Hesaplanması

Sol ve sağ yol çizgisi için her bir dikey koordinata karşılık gelen yatay noktalar arasındaki gerçek ve hesaplanan yatay koordinat değerleri hesaplanmıştır.



Şekil 0.4 ROI üzerinde orijinal ve hesaplanan noktalar gösterimi

Doğru ve yanlış noktaların belirlenmesinde, TuSimple veri setinin önerdiği kriterler esas alınmıştır [39]. Bu doğrultuda, orijinal noktalar ile hesaplanan noktalar arasındaki farkın mutlak değerinin 20'den büyük olması durumunda noktalar hatalı, 20'den küçük olması durumunda ise doğru olarak kabul edilmiştir. Algoritmanın başarısı, aşağıdaki Eşitlik (4.2) ile hesaplanmıştır:

$$\text{Başarı} = \frac{\text{Sol ve Sağ Yol çizgilerindeki Doğru Hesaplanan Nokta Sayısı}}{\text{Sol ve Sağ Yol çizgilerindeki Toplam Nokta Sayısı}} \tag{4.2}$$

Başarı değerlendirmesi, 0 ile 1 arasında normalize edilmiş bir metrik kullanılarak yapılmıştır. Sol ve sağ yol çizgilerindeki hiçbir noktanın doğru tespit edilemediği durumlar 0 başarı oranına karşılık gelirken, tüm noktaların doğru tespit edildiği ideal durum 1 başarı oranına denk gelmektedir.

Şekil 0.5 Örnek görüntü üzerinde orijinal noktalar ve hesaplanan noktalar gösterimi

Bu çalışmada, TuSimple veri setinin eğitim görüntüleri kümesinden 336 eğitim görüntüsü(rastgele olarak) seçilmiş ve analiz süreçlerinde kullanılmıştır.

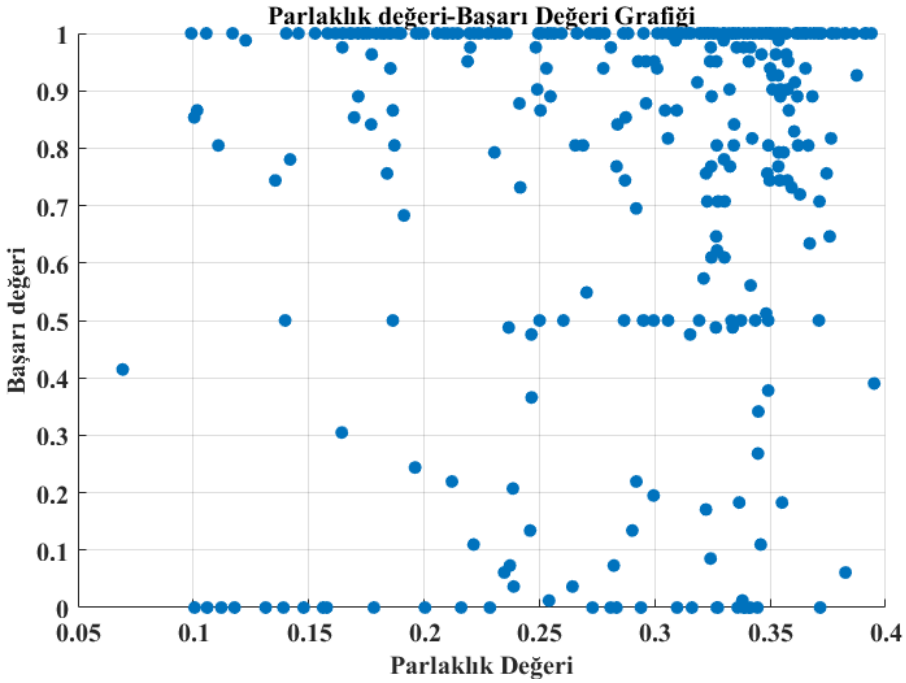
Bu çalışmada geliştirilen algoritma, görüntü kümesindeki tüm görüntülere uygulanmış ve başarılı sonuçlar elde edilmiştir. Her bir yöntem için tüm görüntülerin başarı sonuçlarının ortalaması alınarak, veri seti genelinde ortalama başarı performansı hesaplanmıştır. Elde edilen sonuçlar Tablo 0.4'te sunulmuştur. Tablo incelendiğinde, τ değerinin adaptif olarak seçildiği yöntemin, sabit τ değeri ile çalışan diğer yöntemlere kıyasla önemli ölçüde daha yüksek performans sergilediği görülmektedir.

Tablo 0.4 Yatay filtre komşuluk değeri seçim yöntemi başarı sıralaması sonuçları

Yöntem	**Ortalama Başarı**
Adaptif τ	0.76466
τ=20	0.6878
τ=40	0.67687
τ=60	0.634
τ=80	0.5979

Her bir yöntemin, veri setindeki görüntülerin ortalama parlaklık değerlerine bağlı olarak yol çizgisi tespit başarısı analiz edilmiş ve elde edilen sonuçlar grafikler aracılığıyla sunulmuştur.

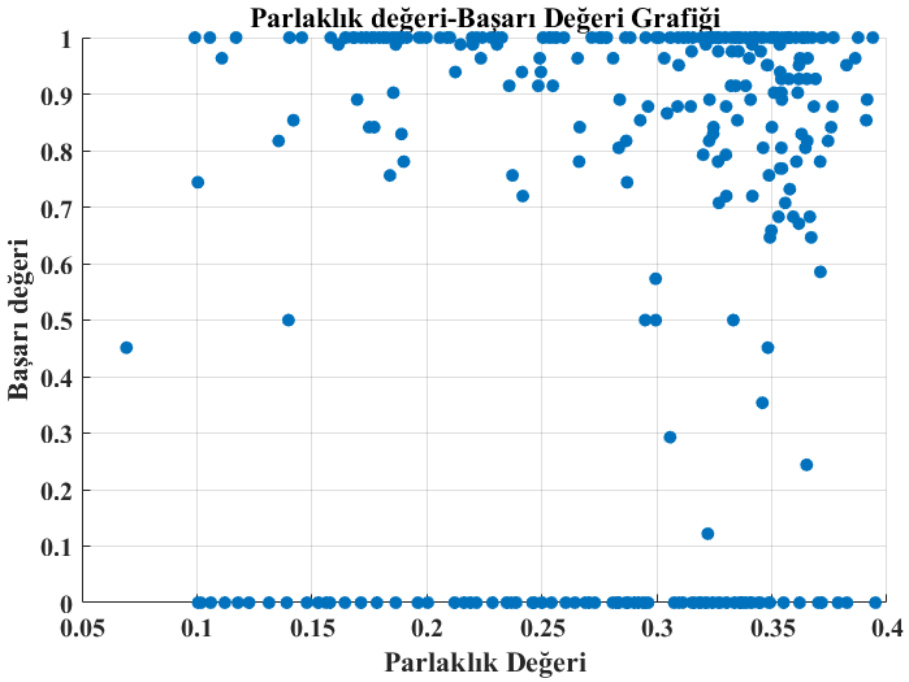
Şekil 0.6'da, adaptif τ yöntemi kullanılarak elde edilen yol çizgisi tespit başarısının, farklı ortalama parlaklık değerlerine göre dağılımı gösterilmektedir. Grafik incelendiğinde, adaptif τ yönteminin genel olarak farklı ortalama parlaklık değerlerinde istikrarlı ve yüksek bir başarı performansı sergilediği gözlemlenmiştir.



Şekil 0.6 Adaptif τ yöntemi için ortalama parlaklık değeri-başarı değeri grafiği

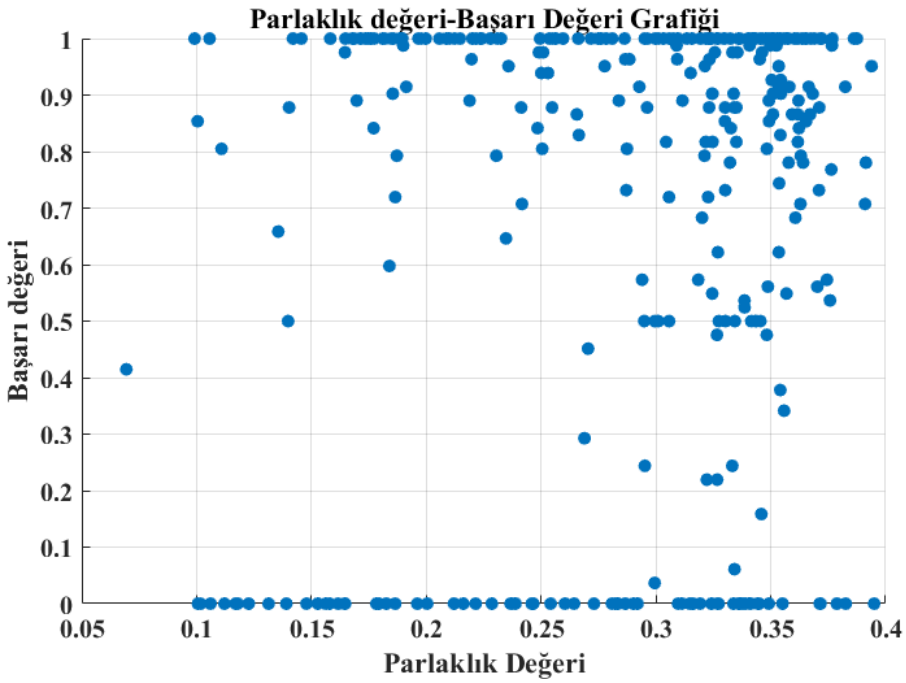
Şekil 0.7'de, sabit τ yöntemi kullanılarak τ=20 değeri için elde edilen yol çizgisi tespiti başarısının farklı ortalama parlaklık değerlerine göre dağılımı sunulmaktadır. Grafik incelendiğinde, sabit τ yönteminin τ=20 değeri için yüksek ortalama parlaklık seviyesine sahip

görüntülerde başarılı sonuçlar verdiği, ancak düşük ortalama parlaklık seviyesine sahip görüntülerde başarımının azaldığı gözlemlenmiştir.



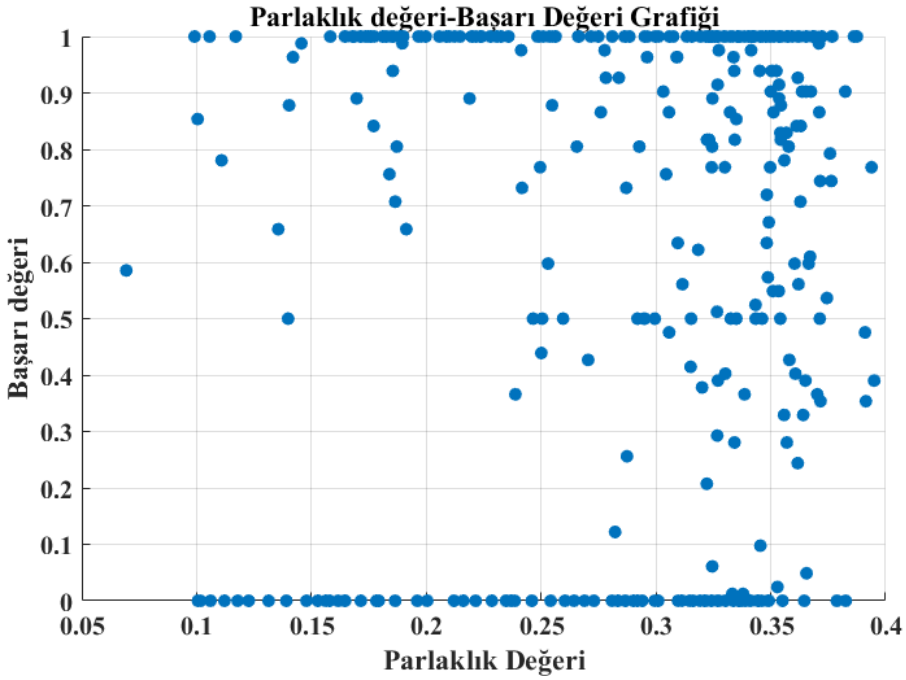
Şekil 0.7 Sabit τ yöntemi için(τ=20) ortalama parlaklık değeri-başarı değeri grafiği

Şekil 0.8'de sabit τ yöntemi kullanılarak τ=40 değeri için elde edilen yol çizgisi tespiti başarısının farklı ortalama parlaklık değerlerine göre dağılımı gösterilmektedir. Grafik incelendiğinde, sabit τ yönteminin τ=40 değeri için parlak görüntülerde ve koyu görüntülerde ortalama sonuçlar verdiği gözlemlenmiştir.



Şekil 0.8 Sabit τ yöntemi için(τ=40) ortalama parlaklık değeri-başarı değeri grafiği

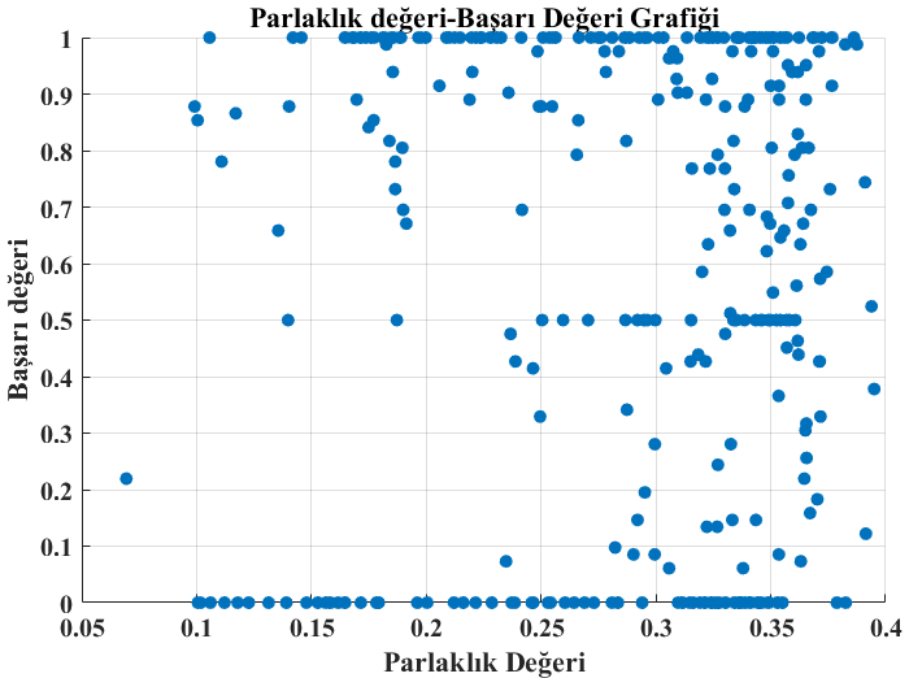
Şekil 0.9'da sabit τ yöntemi kullanılarak τ=60 değeri için elde edilen yol çizgisi tespiti başarısının farklı ortalama parlaklık değerlerine göre dağılımı gösterilmektedir. Grafik incelendiğinde, sabit τ yönteminin τ=60 değeri için parlak görüntülerde düşük sonuçlar, koyu görüntülerde ortalama sonuçlar verdiği gözlemlenmiştir.



Şekil 0.9 Sabit τ yöntemi için(τ=60) ortalama parlaklık değeri-başarı değeri grafiği

Şekil 0.10'da sabit τ yöntemi kullanılarak τ=80 değeri için elde edilen yol çizgisi tespiti başarısının farklı ortalama parlaklık değerlerine göre dağılımı gösterilmektedir. Grafik

incelendiğinde, sabit τ yönteminin τ=80 değeri için parlak görüntülerde düşük sonuçlar, koyu görüntülerde ortalama sonuçlar verdiği gözlemlenmiştir.



Şekil 0.10 Sabit τ yöntemi için(τ=80) ortalama parlaklık değeri-başarı değeri grafiği

Denenen yöntemler için elde edilen başarı grafikleri incelendiğinde adaptif τ seçimi yönteminin hem koyu hem de parlak görüntülerde belirli bir başarı seviyesine ulaşabildiği görülmüştür.

1.23 Çalışmada Karşılaşılan Sorunlar

Çalışma kapsamında, veri seti ile ilgili üç temel sorun tespit edilmiştir. Bu sorunlar, algoritma performansını önemli ölçüde etkilemiş ve ideal algoritma parametrelerinin belirlenmesini güçleştirmiştir. Karşılaşılan sorunlar şu şekildedir:

- Yol çizgisi bilgisi içermeyen yol çizgileri
- Çizgisel olmayan yol çizgileri
- Orijinal yol çizgisi koordinatlarındaki yanlış değerler

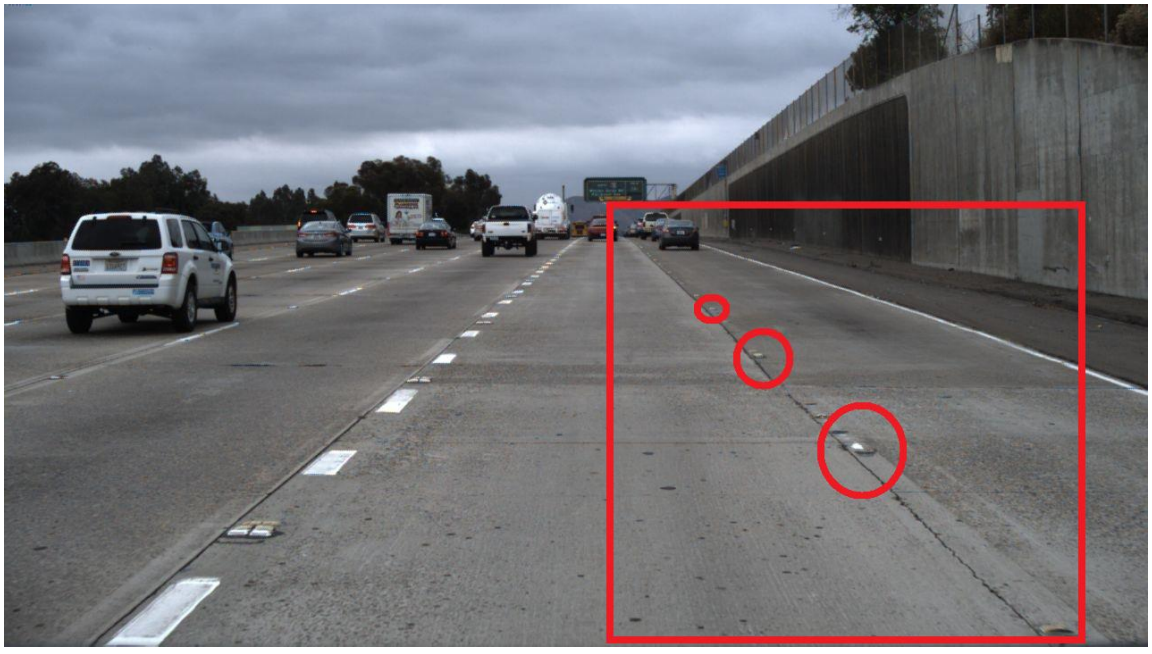
1.23.1 Yol çizgisi bilgisi içermeyen yol çizgileri

Bazı yol görüntülerinde hiçbir yol çizgisi bilgisinin bulunmamasıdır. Örneğin, Şekil 0.11'de herhangi bir yol çizgisi bilgisine ulaşmak imkansızdır. Bu durum, algoritmanın başarısızlıkla sonuçlanmasına ve performansının düşmesine neden olmuştur. Veri setindeki benzer görüntülerin varlığı, bu sorunun etkisini daha da artırmıştır.

Şekil 0.11 Yol çizgisi içermeyen yol görüntüsü

1.23.2 Çizgisel olmayan yol çizgileri

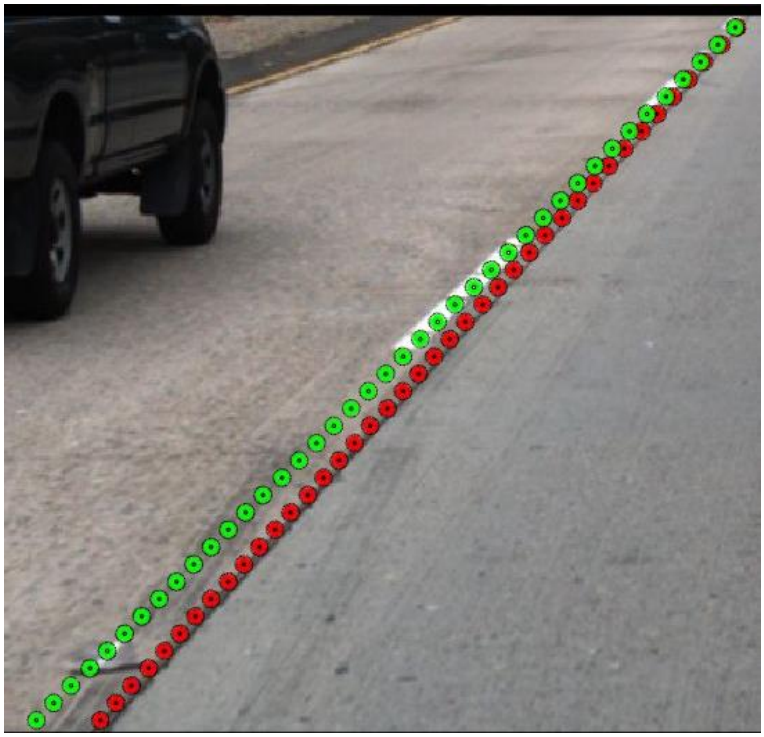
Bazı yol görüntülerinde, standart yol çizgisi yapıları yerine kare şeklinde metal bloklar yer almaktadır. Örnek olarak, Şekil 0.12'de kırmızı alanlarla işaretlenen bu metal bloklar, algoritmanın performansını olumsuz etkileyen unsurlardır. Bu durum, geleneksel yol çizgisi takip algoritmalarının bu yapıları tespit etmesini zorlaştırmış ve algoritmanın başarısızlıkla sonuçlanmasına yol açmıştır.



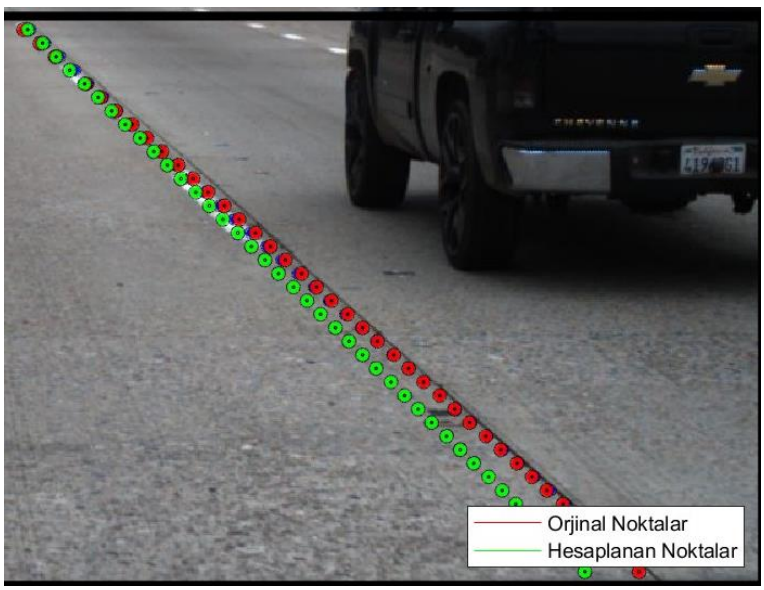
Şekil 0.12 Çizgisel olmayan bir yol çizgisi örneği

1.23.3 Orijinal yol çizgisi koordinatlarındaki yanlış değerler

Bazı görüntülerde, veri setinde sağlanan yol çizgisi koordinat değerlerinde hatalar tespit edilmiştir. Örnek olarak, Şekil 0.13 ve Şekil 0.14'te kırmızı renkler ile gösterilen orijinal yol çizgisi koordinatları hatalıdır. Şekillerde yeşil ile gösterilen koordinatlar ise hesaplanan yol çizgisi bilgileridir. Bu hatalı değerler, algoritmanın başarı oranını olumsuz yönde etkilemiştir.



Şekil 0.13 Orijinal yol çizgisi koordinatlarında yanlış değer içeren görüntü(1)



Şekil 0.14 Orijinal yol çizgisi koordinatlarında yanlış değer içeren görüntü(2)

1.24 Gelecekteki Çalışmalar

Bu algoritma ile ilgili gelecekte planlanan çalışmalar şu şekilde özetlenebilir:

- **Veri Seti Çeşitlendirmesi:** Mevcut veri setindeki sorunlar, algoritmanın gelişim sürecini olumsuz etkilemiştir. Bu nedenle, gelecekte farklı veri setleri üzerinde çalışmalar yapılması ve algoritmanın farklı koşullar altında performansının değerlendirilmesi gerekmektedir.
- **Parametre Optimizasyonu:** Daha stabil bir veri seti kullanıldığında, algoritma parametrelerinin daha tutarlı ve optimal bir şekilde ayarlanması mümkün olacaktır. Bu sayede, algoritmanın genel performansı ve başarımı artırılabilir.
- **τ Değeri ve Parlaklık İlişkisi:** τ değeri ile ortalama parlaklık değeri arasındaki ilişkiyi incelemek için farklı yaklaşımlar denenmelidir:
    - Görüntünün dikey ekseni boyunca her satırın ortalama parlaklık değerini hesaplayarak, her satır için farklı τ değerleri belirlenebilir.
    - Parlaklık ve τ değeri arasındaki ilişkiyi modellemek için temel doğru denkleminin yanı sıra farklı yöntemler de araştırılabilir.
    - Görüntünün her satırındaki lokal maksimum noktalar tespit edilerek, bu noktalardan minimum noktalara kadar olan genişlik τ değeri olarak kullanılabilir.

KAYNAKLAR

[1] Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, Massachusetts: The Mit Press, 2016.

[2] The Editors of Encyclopedia Britannica, “Steam engine,” Encyclopædia Britannica. Oct. 16, 2018.

[3] The Editors of Encyclopedia Britannica, “Karl Benz | German engineer,” *Encyclopædia Britannica*. Mar. 31, 2019.

[4] Wikipedia Contributors, “Benz Patent-Motorwagen,” *Wikipedia*, Oct. 30, 2019.

[5] M. D. o. Transportation, “Edward Hines,” *Michigan.gov*, 2024.

[6] Tesla, “Otopilot,” *Tesla.com*, 2024.

[7] R. C. Gonzalez and R. E. Woods, *Digital image processing*. New Delhi: Dorling Kindersley, 2014.

[8] M. Nieto, J. Arróspide Laborda, and L. Salgado, “Road environment modeling using robust perspective analysis and recursive Bayesian segmentation,” *Machine Vision and Applications*, vol. 22, no. 6, pp. 927–945, Aug. 2010

[9] Papers with Code, “Papers with Code - TuSimple Dataset,”

[10] Mathworks, “Read and Display Image,” Mathworks.com, 2009.

[11] S. Bisht, N. Sukumar, and P. Sumathi, “Integration of Hough Transform and Inter-Frame Clustering for Road Lane Detection and Tracking,” *IEEE Xplore*, May 01, 2022.

[12] Astika Istiningrum, Umi Salamah, and Nurcahya Pradana Taufik Prakisya, “Lane Detection with Conditions of Rain and Night Illumination Using Hough Transform,” 2022 5th International Conference on Information and Communications Technology (ICOIACT), vol. 22, Aug. 2022

[13] H. Zhu, “An Efficient Lane Line Detection Method Based on Computer Vision,” *Journal of Physics: Conference Series*, vol. 1802, no. 3, p. 032006, Mar. 2021

[14] E. Boring, “Visual Perception as Invariance.” 1952.

[15] K. Jain, *Fundamentals of Digital Image Processing*. New Delhi: Prentice-Hall Of India, 2006.

[16] R. Szeliski, COMPUTER VISION : Algorithms and Applications. S.L.: Springer Nature, 2020.

[17] D. Forsyth and J. Ponce, *Computer Vision : a Modern Approach*. Boston: Pearson, 2015.

[18] H. Li *et al.*, “The Effect of Rainfall and Illumination on Automotive Sensors Detection Performance,” *Sustainability*, vol. 15, no. 9, p. 7260, Jan. 2023

[19] Mathworks, “Average or Mean Value of Array - MATLAB Mean,”

[20] H. Jumiawi and A. El-Zaart, “Otsu Thresholding Model Using Heterogeneous Mean Filters for Precise Images Segmentation,” International Conference of Advanced Technology in Electronic and Electrical Engineering, vol. 26, Nov. 2022

[21] W. Niblack, An Introduction to Digital Image Processing. Birkeroed: Strandberg, 1985.

[22] J. Sauvola and M. Pietikäinen, “Adaptive Document Image Binarization,” Pattern Recognition, vol. 33, no. 2, pp. 225–236, Feb. 2000

[23] Mathworks, “Convert Intensity Image to Binary Image Using Level Threshold,”

[24] N. Otsu, “A Threshold Selection Method from Gray-Level Histograms,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan. 1979

[25] Mathworks, “Remove Small Objects from Binary Image - MATLAB Bwareaopen,”

[26] S. Datta, “Comparative Study and Analysis of Various Edge Detection Algorithms in Digital Image Processing,” SAS Journal, vol. 1, pp. 78–90, Jan. 2013.

[27] J. Canny, “A computational approach to edge detection,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, pp. 679–698, Dec. 1986

[28] Mathworks, “Compare Edge Detection Using Canny and Prewitt Methods,”

[29] P. V. C. Hough, “Method and Means for Recognizing Complex Patterns,” Dec. 1962

[30] D. H. Ballard, “Generalizing the Hough transform to detect arbitrary shapes,” *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, Jan. 1981

[31] R. O. Duda and P. E. Hart, “Use of the Hough Transformation to Detect Lines and Curves in Pictures,” *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972

[32] J. Illingworth and J. Kittler, “A Survey of the Hough Transform,” *Computer Vision, Graphics, and Image Processing*, vol. 44, no. 1, pp. 87–116, Oct. 1988

[33] V. F. Leavers, “Which Hough Transform?” *CVGIP: Image Understanding*, vol. 58, no. 2, pp. 250–264, Sep. 1993

[34] Mathworks, “Hough Transform - MATLAB Hough,”

[35] Mathworks, “Identify Peaks in Hough Transform - MATLAB Houghpeaks,”

[36] Mathworks, “Extract Line Segments Based on Hough Transform - MATLAB Houghlines,”

[37] D. S. Moore, G. P. McCabe, and B. A. Craig, *Introduction to the Practice of Statistics*. W H Freeman & Company, 2012.

[38] Mathworks, “Polynomial Curve Fitting - MATLAB Polyfit,”

[39] R. Liu, Z. Yuan, T. Liu, and Z. Xiong, “End-to-end Lane Shape Prediction with Transformers,” *IEEE Xplore*, Jan. 01, 2021.