OTONOM ARAÇLAR İÇİN GERÇEK ZAMANLI NESNE TANIMA VE UZAKLIK TAHMİNİ SİSTEM TASARIM VE UYGULAMASI

YÜKSEK LİSANS TEZİ

SALİH PALAMUT

MERSİN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

MERSİN
OCAK- 2022

OTONOM ARAÇLAR İÇİN GERÇEK ZAMANLI NESNE TANIMA VE UZAKLIK TAHMİNİ SİSTEM TASARIM VE UYGULAMASI

YÜKSEK LİSANS TEZİ

SALİH PALAMUT
ORCID ID: 0000-0003-3431-6268

MERSİN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSYAR MÜHENDİSLİĞİ
ANABİLİM DALI

Danışman
Doç. Dr. Erdinç AVAROĞLU
ORCID ID: 0000-0003-1976-2526

MERSİN
OCAK – 2022

ÖZET

OTONOM ARAÇLAR İÇİN GERÇEK ZAMANLI NESNE TANIMA VE UZAKLIK TAHMİNİ SİSTEM TASARIM VE UYGULAMASI

Günümüzde yaşanan teknolojik gelişmelere birlikte hayatımızın her alanında insanlara kolaylık ve zaman kazandıran ürünler geliştirilmektedir. Otomotiv sanayisinde ise insanların ve çevrede bulunan canlı/cansız nesnelerin korunmasına yönelik olarak nesne tanıyan hatta acil durumlarda karar veren sürücü destek sistemleri geliştirilmektedir. Bu sistemlerin temel amacı olası insan hatalarını iyileştirme, ortadan kaldırma ve bunların yanı sıra insansız araçların alt yapısını oluşturmaktır. Bu sistemler karmaşık sensörlerin birleşiminden oluşabileceği gibi sadece kamera tabanlı sistemlerden de oluşturulabilir.

Günümüzde yaygın olarak kullanılan yöntem mikro dalga radar hesaplama yöntemidir. Mikro dalga radarlar bir yüzeye çarpıp geri yansıma prensibine dayanarak çalıştığı için önde bulunan aracın binek otomobil tarzı olması dışında hataya açık olduğu gözlemlenmiştir. Ayrıca bu sistem engebeli ve eğimli yüzeylerde, aşırı soğuk ve sıcak havalarda hatalı çalışma eğilimi göstermektedir.

Bu çalışmada bazı araçların önünde bulunan mikro dalga radarı ile mesafe ölçümü yerine iki adet kamera kullanılarak derin öğrenme tabanlı gömülü sistem bir mesafe tahmin prototipi hazırlanmıştır. Bu prototip öncelikle kenar bulma algoritması ile test edilmiş, ardından oyuncak arabalar ile farklı uzaklıklarda cm bazında ölçüm tahminleri alınmıştır. Ayrıca kamera gürültüsü ve kalibrasyon hataları işlenmiş en optimum değerler saptanarak üzerine YOLO ile görüntü tanımlama kısmı eklenmiştir. En son alınan değerler YOLO'nun çerçeve hesabıyla gerçek arabalarla yapılmıştır. Alınan sonuçlara göre Jetson Nano platformu gerçek zamanlı nesne tanımlama ve uzaklık tahmini konusunda çok yetersiz olduğu yapılan çalışmalarca ortaya konulmuştur. Gerçek hayatta saniyede 5 kare işlem hızıyla seyir halinde hiçbir araca kullanılamayacağı verilerle sabitlenmiştir. Ayrıca yapılan 150~1500 cm arası ölçüm çalışmasında, hata oranlarının 500 cm de ~%0.15 iken 1100 cm de ~%90.5 olarak hesaplandığından ölçeklendirilebilir olmadığı kanaatine varılmıştır. Bu sonuçlar ışığında öneriler bölümünde başka yöntemler tavsiye edilmiştir.

Anahtar Kelimeler: Derin öğrenme, Görüntü işleme, Çift görü, Mesafe tahmini.

Danışman: Doç. Dr. Erdinç AVAROĞLU, Mersin Üniversitesi, Bilgisayar Mühendisliği Anabilim Dalı, Mersin.

ABSTRACT

REAL-TIME OBJECT IDENTIFICATION AND DISTANCE ESTIMATION SYSTEM DESIGN AND APPLICATION FOR AUTONOMOUS VEHICLES

With the technological developments in today's world, products that provide convenience and time to people are being developed in all areas of our lives. In the automotive industry, on the other hand, driver support systems that recognize objects and even make decisions in emergency situations are being developed for the protection of people and living/inanimate objects in the environment. The main purpose of these systems is to improve and eliminate possible human errors, as well as to create the infrastructure of unmanned vehicles. These systems can consist of a combination of complex sensors, or they can be created from only camera-based systems.

Since microwave radars hit a surface and work based on the back reflection principle, it has been observed that the vehicle in front is prone to error, except that it is a passenger car style. In addition, this system tends to malfunction on uneven and sloping surfaces, and in extremely cold and hot weather.

In this study, a deep learning based embedded system distance estimation prototype was prepared by using two cameras instead of microwave radar in front of some vehicles. This prototype was first tested with the edge detection algorithm, then measurement estimates were taken with toy cars at different distances on a cm basis. In addition, camera noise and calibration errors were processed and the most optimum values were determined and the image definition part with YOLO was added. The most recent values were made with real cars with YOLO's frame calculation. According to the results obtained, studies have shown that the Jetson Nano platform is very inadequate in real-time object identification and distance estimation. It has been fixed by data that it cannot be used in any vehicle while driving at a processing speed of 5 frames per second in real life. In addition, in the measurement study between 150~1500 cm, it was concluded that it is not scalable since the error rates were calculated as ~0.15% at 500 cm and ~90.5% at 1100 cm. In the light of these results, other methods have been recommended in the recommendations section.

Keywords: Deep learning, Image processing, Stereo vision, Distance estimation

Advisor: Assoc. Prof. Erdinç AVAROĞLU, Department of Computer Engineering, University of Mersin, Mersin.

TEŞEKKÜR

Tezin hazırlanmasında emek edenlerden ilk olarak hocalarım Dr. Öğr. Üyesi Abdullah ELEWİ ve Doç. Dr. Erdinç AVAROĞLU' na, mekanik tasarım ve üretim konusunda Toros üniversitesi Öğr. Gör. Serkan PALAMUT ve endüstriyel tasarım mühendisi Serap PALAMUT' a ve bu yolculukta bana maddi manevi yardımlarını kaçınmayan aileme katkılarından dolayı teşekkür ederim.

İÇİNDEKİLER

	Sayfa
İÇ KAPAK	**i**
ETİK BEYAN	**iii**
ÖZET	**iv**
ABSTRACT	**v**
TEŞEKKÜR	**vi**
İÇİNDEKİLER	**vii**
TABLOLAR DİZİNİ	**ix**
ŞEKİLLER DİZİNİ	**x**
SİMGELER VE KISALTMALAR	**xii**
1. GİRİŞ	**1**
1.1. Derin Öğrenme Yönteminin Tanımı ve Önemi	2
1.2. Kullanım Alanları	3
2. KAYNAK ARAŞTIRMALARI	**4**
2.1. Kamera Tabanlı Uzaklık Tespit Sistemleri	4
2.1.1. Zorlu Aydınlatma Koşullarında Güçlü Araç Tespiti ve Mesafe Tahmini	4
2.1.1.1. İşlemler	4
2.1.1.2. Sonuç	6
2.1.2. Görünür Işık İletişimine Dayalı Düşük Çözünürlüklü Bir Kamera Kullanarak Araçtan Araca Mesafe Tahmini	9
2.1.2.1. İşlemler	10
2.1.2.2. Sonuç	11
2.1.3. DisNet: Monoküler kameradan mesafe tahmini için yeni bir	13
2.1.3.1. İşlemler	13
2.1.3.2. Sonuç	14
2.1.4. CNN Tarafından Çıkarılan Derinlik İpuçları ile Monoküler Kamera Görüntülerinden Kör Nokta Engel Algılama	15
2.1.4.1. İşlemler	16
2.1.4.2. Sonuç Analizi	17
2.1.5. DistanceNet: Tekrarlayan Evrişimli Sinir Ağı Kullanarak Monoküler Görüntülerden Katedilen Mesafeyi Tahmin Etme	18
2.1.5.1. İşlemler	19
2.1.5.1.1. Eğitim	19
2.1.5.1.2. KITTI Veri Kümesi	19
2.1.5.2. Sonuç Analizi	19
2.1.6. Kör Alan Tesiti için Derin Öğrenme Tabanlı Nesnelerin Mesafe Ölçüm Yöntemi	21
2.1.6.1. İşlemler	21
2.1.6.2. Sonuç	22
2.2. Kaynak Araştırması Sonucu	24
3. MATERYAL ve YÖNTEM	**25**
3.1. Gerçek Zamanlı Sistemler	25
3.1.1. Gerçek Zamanlı Linux Çekirdeği	26
3.1.2. YARDIMCI ÇEKİRDEK YAKLAŞIMLARI	27
3.1.3. ÖNCELİKLİ ÇEKİRDEK YAKLAŞIMI	29
3.2. Çift Görü	29
3.2.1. Lens Gürültüsü	31
3.2.2. Kalibrasyon	32
3.2.2.1. Kalibrasyon Tahtası	32
3.2.2.2. Çift Kamera Tek Düzlem Oluşturma	33
3.3. Kullanılan Donanım ve Bileşenleri	34
3.3.1. Tek Kart Bilgisayar	34
3.3.2. Kamera	35

	Sayfa
3.3.3. Kamera Yerleşimi	36
3.3.3.1. Montaj Parçası	37
3.3.3.2. Montaj	37
3.4. Kullanılan Yazılım ve Bileşenleri	38
3.4.1. Gnu/Linux	38
3.4.2. Jetson Nano Developer Kit SD Card Image,	38
3.4.2.1. Jetson Nano Sistem Kurulumu	38
3.4.2.2. Cihazı Çalıştırma	39
3.4.2.3. İmajı Yazma	39
3.4.2.4. Cihaz Bağlantılarının Yapılması	39
3.4.2.4.1. SD Kartın Yerleştirilmesi	40
3.4.2.4.2. Ekran Bağlanması	40
3.4.2.4.3. Klavye ve Fare Bağlantısı	41
3.4.2.4.4. USB Güç Bağlantısı	41
3.4.2.4.5. Besleme Tipi Seçimi	41
3.4.2.4.6. Usb Erişim Bağlantısı	42
3.4.2.4.7. AdaptörGüç Bağlantısı	43
3.4.2.5. Cihazı Çalıştırma	43
3.4.2.6. Uzak Erişim Ayarlarını Yapma	51
3.4.3. Jetson Nano Üzerinde Gerçek Zamanlı Lınux Çekirdeği Uygulanması	53
3.4.3.1. Test Yazılımı İşlemleri	55
3.4.3.1.1. Test Yazılının Çalıştırılması	55
3.4.3.1.2. Yamadan Önce Ve Sonrasının Karşılaştırılması	55
3.4.4. C/C++	57
3.4.5. OpenCv	58
3.4.5.1. İşlevselliği	58
3.4.5.2. Kurulumu	58
3.4.6. CUDA	59
3.4.7. DARKNET YOLO Kütüphanesi ve Çalışma Şekli	60
3.4.7.1. Bilgisayar Görüşü ve Konvolüsyonel Yapay Sinir Ağları	60
3.4.7.2. YOLO	61
3.4.7.3. YOLO(You Only Look Once)'nun Çalışma Şekli	62
3.4.7.4. Yolo Kurulumu	63
4. BULGULAR ve TARTIŞMA	**64**
4.1. Kenar Bulma Algoritmasıyla Uzaklık Tespiti	64
4.1.1. Kalibrasyon Öncesi	64
4.1.2. Kamera Kalibrasyonu	66
4.2. YOLO Algoritmasıyla Uzaklık Tespiti	68
4.2.1. OpenCv de YOLO Kullanımı	68
4.2.2. Nesne Tabanlı Uzaklık Tespiti	69
4.2.3. Nesne Tabanlı Uzaklık Tespiti Sonuçları	70
4.3. Gerçek Hayata Uyarlama	70
5. SONUÇLAR ve ÖNERİLER	**73**
KAYNAKLAR	75
ÖZGEÇMİŞ	**81**

TABLOLAR DİZİNİ

	Sayfa
Tablo 2.1. Performans Değerleri [7]	9
Tablo 2.2. Simülasyon Değerleri [26]	11
Tablo 2.3. Tahmini Uzaklık ve Gerçek Uzaklık	14
Tablo 2.4. Test Sonuçları [32]	18
Tablo 2.5. KITTI üzerindeki yaklaşık sonuçlar [39]	20
Tablo 2.6.Seri 09 ve 10 un Ortalama Sonuçları [39]	20
Tablo 2.7. Parametreler	22
Tablo 2.8. Değerler	23
Tablo 3.1. Teknik Özellikler	34
Tablo 3.2. Sensör bilgileri	36
Tablo 3.3. Besleme Tipi	42
Tablo 3.4. Otomatik Giriş	51
Tablo 3.5. Kernel İndirme	53
Tablo 3.6. Kernel Ayar	53
Tablo 3.7. Menü Seçimleri	54
Tablo 3.8. Kernel Derleme	55
Tablo 3.9. Kaynaktan Derleme	55
Tablo 3.10. Grafik Programı	55
Tablo 3.11. Cyclictest Parametreleri	56
Tablo 3.12. İşaretçi Örneği	57
Tablo 3.13. OpenCv Kurulum Adımları	58
Tablo 3.14. Yolo Kurulumu	63
Tablo 4.1. Kamera Tanımlama Fonksiyonu	64
Tablo 4.2. OpenCv Kamera Tanımlama	64
Tablo 4.3. Başlangıç Değerleri	65
Tablo 4.4.FPS Karşılaştırması	70
Tablo 4.5. Uzaklık Değerleri	70
Tablo 4.6.FPS Karşılaştırması	72
Tablo 4.7. Ölçüm Değerleri	72

ŞEKİLLER DİZİNİ

	Sayfa
Şekil 1.1. Hatalı Durumlar	2
Şekil 1.2. Yapay zeka kapsamı [4]	3
Şekil 2.1. Yeni Tip Haar [7]	5
Şekil 2.2. Başarılı araç algılama örnekleri [7].	6
Şekil 2.3. Gün için araç tespiti [7].	7
Şekil 2.4. Yağmurlu gündüz ve yağmurlu gece için performans değerlendirmesi [7].	8
Şekil 2.5. Kar durumunda araç tespiti ve mesafe tahmini [7].	8
Şekil 2.6. Sistem mimarisi [26]	10
Şekil 2.7. LED sinyalinin görüntüden çıkarılması [26]	10
Şekil 2.8. Önerilen çözüm telafisi. [26]	11
Şekil 2.9. Genel Performans [26]	12
Şekil 2.10. Görüntü sensörünün çözünürlük telafisi ile mesafe hatası [26].	12
Şekil 2.11. Nesne mesafesi tahmini için kullanılan DisNet tabanlı sistem.	14
Şekil 2.12. Yüz metre tahmini	15
Şekil 2.13. Çoklu mesafe tahmini	15
Şekil 2.14.Kamera Konumları	16
Şekil 2.15. PCA ve DCT Algoritması [32]	17
Şekil 2.16 Sekans 05 ve 07	20
Şekil 2.17 Kalibrasyon plakası.	21
Şekil 2.18. Kameralar	22
Şekil 2.19. Grafiksel Gösterim	24
Şekil 3.1. Kalibre Verisi	25
Şekil 3.2. Sistem Modeli	26
Şekil 3.3. Üç Farklı Yardımcı Çekirdek Yaklaşımı (a) RTAI, (b) Xenomai ve (c) RTLinux [43]	28
Şekil 3.4. İdeal Çift Kamera Düzeneği [53]	30
Şekil 3.5. Eşitsizlik Uzaklık Eğrisi [53]	31
Şekil 3.6. OpenCv Kordinat Sistemi [53]	31
Şekil 3.7. Radial Bozulma	32
Şekil 3.8. Teğetsel Bozulma	32
Şekil 3.9. Kalibrasyon	33
Şekil 3.10. Görüntü Düzeltme ve Birleştirme [53]	33
Şekil 3.11. Grafiksel Görüntü	35
Şekil 3.12. RaspberryPi V2 Kamera [58]	35
Şekil 3.13. Gözbebekleri Arası Mesafe	36
Şekil 3.14. Metal Parça Ölçüleri	37
Şekil 3.15.Montajlı Görünüm	37
Şekil 3.16.İmaj ve Tarihleri	39
Şekil 3.17. Etcher Yazılımı	39
Şekil 3.18. SD Kart Takma	40
Şekil 3.19. HDMI Ekran Bağlantısı	40
Şekil 3.20.Usb Klavye Ve Fare Bağlantısı	41
Şekil 3.21. Usb Güç Bağlantısı	41
Şekil 3.22.Usb Seri Haberleşme	42
Şekil 3.23. Güç Bağlantısı	43
Şekil 3.24.İlk Ekran	44
Şekil 3.25. Lisans Sözleşmesi	44
Şekil 3.26. Dil Seçimi	44
Şekil 3.27. Ülke Seçimi	45
Şekil 3.28. Ülke Seçimi İkinci.Adım	45
Şekil 3.29. Ülke Seçimi Üçüncü Adım	45
Şekil 3.30. Yerelleştirme Seçimi	46

	Sayfa
Şekil 3.31. Saat Seçimi	46
Şekil 3.32. Tam isim veya nick name.	46
Şekil 3.33. Kullanıcı Adı	47
Şekil 3.34. Şifre	47
Şekil 3.35. Şifre Doğrulama	47
Şekil 3.36. Zayıf Şifre Uyarısı	48
Şekil 3.37. Disk Boyutu	48
Şekil 3.38. Disk Yapılandırması	48
Şekil 3.39. İnternet Arayüzü	49
Şekil 3.40. Konak Adı	49
Şekil 3.41. Nvpmode	49
Şekil 3.42. Kurulum	50
Şekil 3.43. Giriş Ekranı	50
Şekil 3.44. Kullanıcı Girişi	50
Şekil 3.45. Sistem Erişimi	50
Şekil 3.46. Güncelleme	51
Şekil 3.47.Ip Adresini Öğrenme	51
Şekil 3.48. X11VNC Açılışı	52
Şekil 3.49. Uzak Masa Üstü	52
Şekil 3.50. Çekirdek Ayarları Bölümü	54
Şekil 3.51 Çekirdek Özellikleri Bölümü	54
Şekil 3.52. Gerçek Zaman Destek Çeşidi	54
Şekil 3.53. SMP Kernel	56
Şekil 3.54. RT Kernel	56
Şekil 3.55. Cuda Bilgisi	59
Şekil 3.56. Nesne Tanıma için Örnek Görüntü [79]	60
Şekil 3.57. LeNet Mimarisi [80]	61
Şekil 3.58. YOLO'nun Çalışma Şekli [83]	62
Şekil 4.1. Uzaklık Bulma Fonksiyonu	65
Şekil 4.2. Nesne Konumu	66
Şekil 4.3.Uygulama Ekran Görüntüsü	66
Şekil 4.4. Hesaplanan Çıktı	66
Şekil 4.5. Kalibre İşlemi	67
Şekil 4.6. Çift Bord	67
Şekil 4.7. Kalibre Hatası	68
Şekil 4.8.Yolo Verileri	68
Şekil 4.9.Yolo nesne Tanıma	69
Şekil 4.10.FPS Artışı	69
Şekil 4.11.Son Çalışma	70
Şekil 4.12. Görsel Gösterim	71
Şekil 4.13. Önden Görünüm	71
Şekil 4.14. İçeri Görünüm	71
Şekil 4.15. Ölçüm 1500cm	71
Şekil 4.16. Ölçüm 1000cm	72
Şekil 4.17. Ölçüm 150cm	72
Şekil 5.1. Jetson Platformları [86]	73
Şekil 5.2. ZED Kamera [87]	74

SİMGELER VE KISALTMALAR

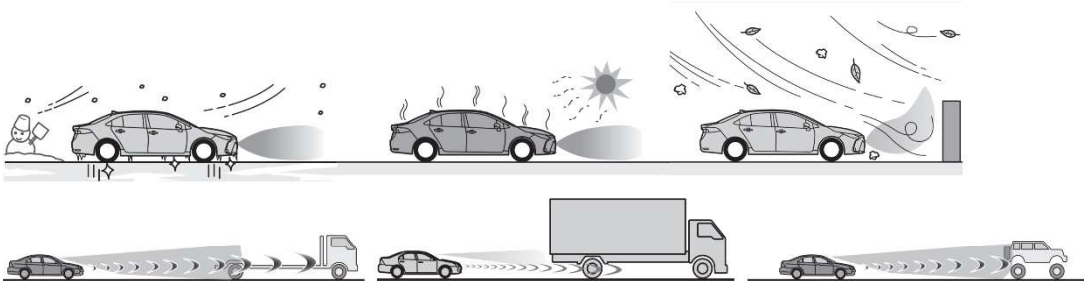
Kısaltma/Simge	Tanım
BSD	Berkeley Software Distribution
CNN	Convolutional Neural Network
GNU	GNU General Public License
GPU	Graphic Processing Unit
HFOV	Horizontal Field of View
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
SBC	Single Board Computer
VOC	Visual Object Classes
vSLAM	visual simultaneous localization and mapping
YOLO	You Only Look Once

1. GİRİŞ

Sürücüsüz ve elektrikli araçlara yapılan yatırımın artması otomotiv alanında yeni araştırma alanları doğurmuştur. Tesla benzeri firmaların ürettiği araçlar ve bu araçların petrol türevi yakıtla çalışan araçlardan daha çevreci oluşları sebebiyle otomotiv sektöründe bulunan köklü firmalar eskiden konsept olarak ürettiği ve tasarım aşamasında bıraktığı ürünlerini piyasaya sunmaya başlamışlardır. Özellikle 2018 yılında Almanya Köln'de [1] başlayan ve günümüzde çoğu Avrupa ülkesinde de benzer olarak uygulamaya konulan dizel yakıt kullanan araçlarının kısmen veya tamamen yasaklanması sektörün elektrikli araçlara geçişini daha da hızlandırmıştır. Elektrikli araçlar temelde tüm kontrollerini ve kararlarını elektronik olarak gerçekleştirdiğinden sürücü destek sistemlerinin araca uygulanma maliyeti azalmıştır. Geçiş döneminde hali hazırda satışı olan araçlar üstünde sürücü destek sistemlerinin ar-ge süreci devam etmektedir. Bu sebeple araçların üzerinde bulunan sürücü destek sistemleri hem özellik hem de etkinlik bakımından iyileştirilmiştir. Eskiden şerit ihlalini sadece görsel veya sesli uyarı olarak veren şerit takip sistemi günümüzde direksiyona müdahale ederek aracı iki şerit arasına güvenli bir şekilde tekrar konumlandırmaktadır.

Eski nesil olarak tabir edilen radar maliyet ve düşük çözünürlükten dolayı düşük çözünürlüklüdür. Oysaki, otomotiv radarı için önemli gereksinimler yüksek çözünürlük, düşük donanım maliyeti ve küçük boyuttur. Çok girişli, çok çıkışlı (MIMO) radar teknolojisi, nispeten az sayıda antenle yüksek açısal çözünürlük elde edebildiği için otomotiv radar üreticilerinden büyük ilgi görmektedir [2]. Şu an yeni ve maliyetli olduğundan kullanılmamakta veya çok lüks araçlarda bulunmaktadır.

Günümüzde en yaygın kullanılan yöntem ses sinyalinin katı bir nesneye gönderilip nesneden yansıyan sinyale göre uzaklık hesaplanmasıdır. Mevcut durumda araçların önlerine çıkan nesneleri katı varlık olarak algılaması ve radar sistemleri kullanılarak çözülmesine odaklanılmıştır. Bu sistemler Şekil 1.1. de görüldüğü gibi rüzgâr ve sıcaklık değişiminden etkilendiği gibi öndeki aracın küçük kişisel taşıma araçları gibi dar bir araç olması, motosiklet veya bisiklet olması, arka ucu küçük olması (yüksüz bir kamyonet gibi) veya aracın arka ucu alçaktaysa (alçak kasalı tırlar gibi) hatalı çalışmaktadır [3].



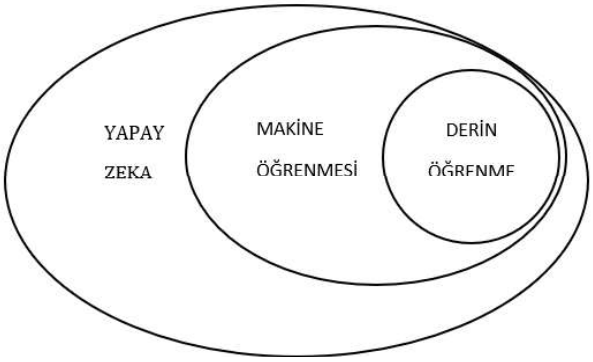
Şekil 1.1. Hatalı Durumlar

İlgili tez çalışması beş ana bölümden oluşmaktadır. Bu kapsamda, mikro dalga radarlar incelenmiş ve problem tanımı "Giriş" bölümünde yapılmıştır. Tezin literatür taraması "Kaynak Araştırmaları" bölümünde ayrıntılı olarak işlenmiştir. Bu çalışmada gerçekleştirilen ve kullanılan sensör, teknoloji ve uygulama yapısı "Materyal ve Yöntem" bölümünde yer almıştır. "Bulgular ve Tartışma" bölümünde gerçekleştirilen prototip ve elde edilen veriler sunuluştur. Tezin son bölümü olan "Sonuç ve Öneriler" bölümünde çalışma sonuçlarının genel değerlendirmesi ve bu değerlendirmeler sonucunda öneriler verilmiştir.

1.1. Derin Öğrenme Yönteminin Tanımı ve Önemi

Derin öğrenim yöntemi, tek hücreli sinir sistemi olan canlıların taklidinden başlayıp en gelişmiş canlı olan insanın beyin organın çalışma kuramı referans alınarak geliştirilmiş ve sürekli bir geliştirilme aşamasında olan, bilgisayar destekli sistemlerin mevcut/toplanan girdilerden yararlanarak bir sorunu algılaması ve sorunun çözüm yollarına yönelik istenilen doğrultuda sonuçlar türetmesiyle ilgilenen bir çalışma alanıdır. Son on yıllık tarihte yaşanan veri işleme gücü artmış donanımların üretiminin hızlanması, kolay erişilebilme olanağı sağlanması, yazılım alanında yapılan açık kaynak desteklemeleri ve çalışan örnek kodların varlığı sayesinde popülerliği giderek artmıştır. Derin öğrenme kapsamı makine öğrenimi ve yapay zekâ alanlarının kapsamı olarak değerlendirilmektedir.

Makine öğrenmesi; bilgisayarlı sistemlerin, canlıların yapabildiği bilinen süreçleri öğrenmesi, bu süreçleri matematiksel olarak uygulaması ve gerçek hayatta aktarması konularıyla ilgilenmektedir. Makine öğrenmesi, veri madenciliğiyle elde edilen veri setlerinin artması ve bu verileri. İçinde makine öğrenmesi ve derin öğrenmeyi barındıran yapay zekâ alanı canlılar tarafından gerçekleştirilen zekâ temelli eylemlerin, değerlendirmelerin ve düşüncelerin bilgisayarlı sistemler tarafından uygulanabilmesiyle ilgilenmektedir. Derin öğrenme alanı yapay zeka ve makine öğrenmesi alanlarının sağladığı geniş bir çalışma alanına sahiptir. Bu ilişki Şekil 1.2.'de gösterilmiştir.



Şekil 1.2. Yapay zeka kapsamı [4]

1.2. Kullanım Alanları

Yıllarca ve sürekli gelişim halinde olan derin öğrenme alanı, makine öğrenmesi alanıyla çözülemeyen, çözümde istenilen başarıya ulaşılamayan problemlerin çözümünde başarıya ulaşıp önemli sonuçlar ortaya koymuştur.

Derin öğrenme yöntemleriyle gelişen bazı alanlar ve yapılan faydalı çalışmalar aşağıda örnek olarak verilebilir [5,6]:

- El yazısı tanımlana, tanıma ve buna bağlı dokunmatik kalemli yazma sistemlerinin geliştirilmesi.
- Sesi tanımlama, algılama ve konuşma analizi yapan sistemlerinin geliştirilmesi.
- Görüntü işleme, tanıma ve sınıflandırma sistemlerinin geliştirilmesi.
- Otomatik yazı, dil bilgisi denetimi ve tercüme araçlarının geliştirilmesi.
- Yazılı metinden veya görüntüden konuşma sesi üretimine.
- Dijital reklam, pazarlama ve kişiye yönelik satış yöntemlerinde daha tutatrlı şekilde hedef kitle tespit edilmesi.
- Satranç ve go gibi oyunlar da insan rakiplerini yenecek seviyeye ulaşılması.
- İdeal şartlarda insanlardan daha iyi, normal şartlarda ise deneyimli şoförlere yakın yakın otonom sürüş destekli taşıtların geliştirilmesi.
- Sorulan soruları insana yakın cümle ve cümle öbekleriyle yanıtlandırabilen akıllı asistan uygulamalarının geliştirilmesi.

2. KAYNAK ARAŞTIRMALARI

Bu bölümde, literatürde yer alan kamera kullanılarak uzaklık tahmini için geliştirilmiş nesne tanıma destekli sistemlerin tasarım, uygulama ve kullanım şekilleri incelenmiştir. Başkalarının eserlerinden yararlanılırken ilgili eserlere bilimsel kurallar çerçevesinde atıfta bulunulmuş ve içerikleri tanım, yöntem ve sonuç bilgileri toplanarak bu çalışmada yer verilmiştir.

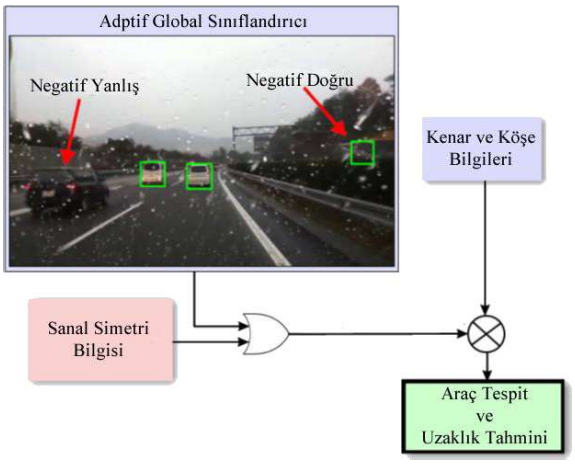
2.1. Kamera Tabanlı Uzaklık Tespit Sistemleri

2.1.1. Zorlu Aydınlatma Koşullarında Güçlü Araç Tespiti ve Mesafe Tahmini

Dr. Mahdi Rezaei ve arkadaşlarının yaptığı stereo görüş tabanlı algoritmalarda yer alan yüksek hesaplama maliyetleri ve kalibrasyon sorunlarından kaçınan bu çalışmada, eş zamanlı araç algılama ve araçlar arası mesafe tahmini için gerçek zamanlı monoküler görüş tabanlı teknikler önerilmiştir [7]. Son derece zorlu kıyaslama veri kümeleri için öndeki araçları algılayarak ve olası bir çarpışma meydana gelmeden önce dikkati dağılmış bir sürücüye yardımcı olmak için güvenli durma mesafelerini belirleyerek bir çarpışma uyarı sistemi geliştirilmiştir. Algoritmalarının doğruluğunu ve sağlamlığını geliştirmek için araç algılama, arka ışık segmentasyonu, sanal simetri algılama, araçlar arası mesafe tahmini için uyarlanabilir küresel Haar benzeri [8,9,10] özelliklerin yanı sıra verimli bir tek sensörlü çok özellikli füzyon tekniği sunulmuştur [11]. Önerilmiş olan algoritma hem gündüz hem de gece öndeki araçları ve ayrıca kısa ve uzun menzilli mesafeleri tespit edebilmektedir. Çeşitli hava ve aydınlatma koşullarında (güneşli, yağmurlu, sisli veya karlı) deneysel sonuçlar, önerilen algoritmanın en gelişmiş algoritmalardan daha iyi performans gösterdiğini göstermektedir [12].

2.1.1.1. İşlemler

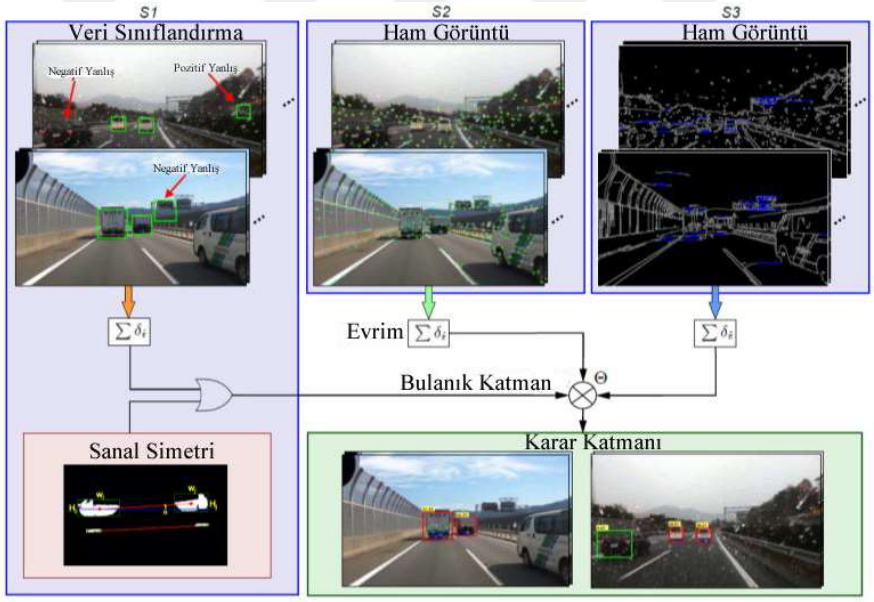
Bu çalışmada yoldaki araçları tespit etmek ve yerini belirlemek için bilgisayarla görü tabanlı yöntemler, ya çok karmaşık yol görüntülerini işleyerek doğrulamayı arttırmaya çalışır ya da çok pahalı hesaplama maliyetlerine gereksinim duyar. Bu durum yapılan çalışmalarda, tanıtılan sürücü yardım sisteminin birçoğunun gerçekçi ve pratik olmamasına neden olmaktadır. Yaptıkları bu çalışmada kullandıkları yeni tip Haar özellikleri kullanarak aday bölgelerin tespitini, bu bölgelerin müteakip analizi ile birleştiren önerilen tespit algoritmasının kısa taslağı.Şekil 2.1. de yer almaktadır.



Şekil 2.1. Yeni Tip Haar [7]

AGHaar sınıflandırması, yatay çizgiler, köşe özellikleri ve sanal simetri algılama ile elde edilen sonuçların nasıl birleştirileceğini açıklanmıştır. Şekil 2.2.ye göre,nihai hedef, doğru araç tespitidir. Elde edilen veriler tek bir sensörden türetildiğinden, açıkça zaman senkronize olduğundan ve farklı işlemler için aynı pozda olduğundan, çoklu veri füzyon işlemi için bundan faydalanılmıştır. Genellikle çoklu sensör füzyon tekniklerinde yer alan zaman hizalama, veri kaydı, sensör doğrulama ve benzeri diğer aşamalar bu işlemde gereksizdir. Yeni AGHaar yönteminin tek başına yol senaryolarının çoğunda yeterince doğruluk sağladığı gösterilmiş ve daha da güvenilir bir algılama sağlamak, bir aracı tespit etmek, mevcut tüm kanıtlar için, tıpkı bir sürücünün sürüş sırasında yaptığı gibi, veri birleştirme uygulanmıştır. Örneğin, bir aracın tüm gövdesi sisli havalarda görünmüyorsa, uzman bir sürücü, konumunu ve boyutunu tahmin etmek için bir aracın plakasını, arka lambalarını veya diğer özelliklerini aramayı düşünebilir. Füzyon yaklaşımında, hesaplama maliyetini yalnızca önemsiz ölçüde artırırken daha doğru sonuçlara yol açar. Bu, tüm sürecin gerçek zamanlı performansını engellemez. Veri füzyonu için iki olası yaklaşımı, yani Bayesian ve Dempster-Shafer [13] teorisini düşünülmüştür. Bayes yöntemi, girdi varlıklarının ağırlıklarını olasılıklar olarak yorumlar. Dempster-Shafer teorisi (inanç teorisi veya kısaca D-S teorisi olarak da adlandırılır), yalnızca olasılıklar kavramına yaklaşan insan uzmanlığına dayalı “kitleler” atar. Bayes yaklaşımı "saf" istatistiksel analize dayandığından, her bilgi kaynağı için tüm istatistiksel verileri sağlama konusunda da "saf" (yani çok doğru) olunması gerekir. Bu, sonuç olarak, farklı yol sahnelerinden çok çeşitli kaydedilmiş videolar arasında kapsamlı bir ilk veri-tabanı analizi gereksinimini beraberinde getirir. Aksi takdirde, sonuçta ortaya çıkan hatalı ağırlık atamaları, veri füzyonunun tamamen yanlış sonuçlarına neden olabilir [14]. Bayes yönteminin aksine, D-S teorisi, Bayesçi olasılık yaklaşımına kıyasla kanıt ve belirsizliği modellemek için alternatif bir yöntem olarak hizmet ederek uzmanların belirsiz yargılarını ifade etmedeki etkinliği ile iyi bilinmektedir. D-S teorisi iki fikre dayanmaktadır: (1) İlgili bir soru için

"öznel olasılıkları" belirlemek için bir inanç derecesinin tanımlanması ve (2) Dempster'ın bağımsız kanıt öğelerinden inanç derecelerini birleştirme kuralı. D-S teorisini araç tespiti için bir veri birleştirme çözümü olarak kullanarak, sadece "araç" ve "araçsız" olmak üzere iki kategoriyi dikkate almakla kalmıyor, aynı zamanda bir "bilinmeyen" durum için bir inanç derecesi de atanmış olur. "Bilinmeyen" durumu için bir kitle göz önüne alındığında, olası yanlış tespitleri önlemek için bir güvenlik payı eklenmiştir. Bu, adım otomatik olarak bilgi konsensüsü ve insan uzmanlığının bir kombinasyonuna dayanan daha rasyonel kararlara götürür; oysa Bayes tekniğinde, yalnızca iki olasılık değerine ("mevcut" veya "var olmayan" için) sahip olunur, ancak her ikisinin bir kombinasyonuna sahip olunmaz. Dikkate alınmış bu bağlamda, özellikle bilgi eksikliğimiz varsa ve her bir bilgi kaynağının doğruluğunun bireysel olarak garanti edilemediği bir durumda, D-S teorisine dayalı bir füzyon yaklaşımının daha kabul edilebilir sonuçlara yol açtığı gösterildi [7].

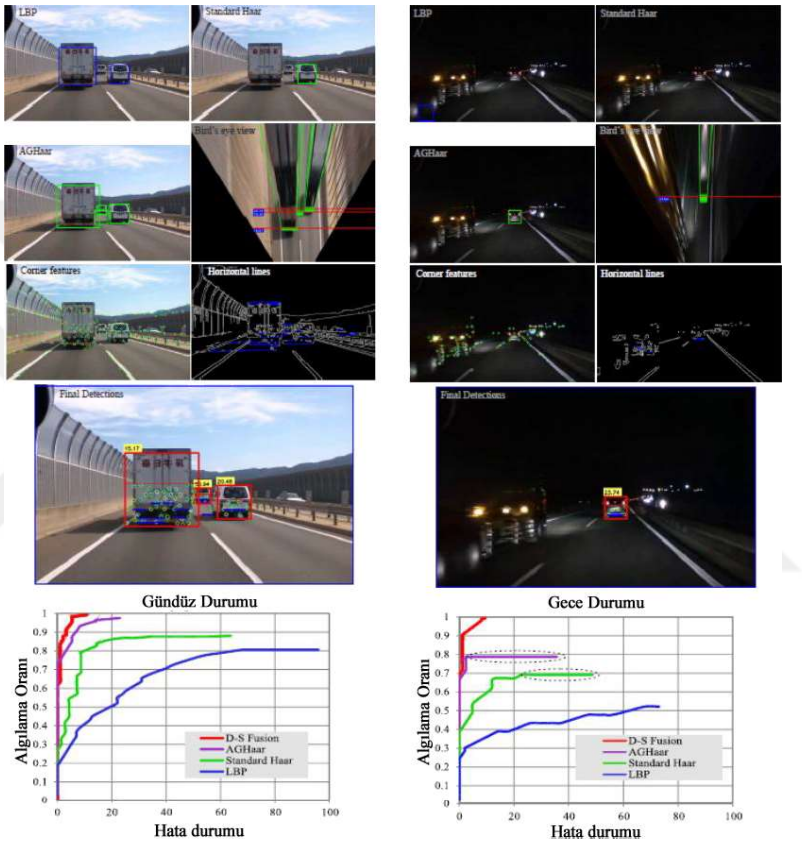


Şekil 2.2. Başarılı araç algılama örnekleri [7].

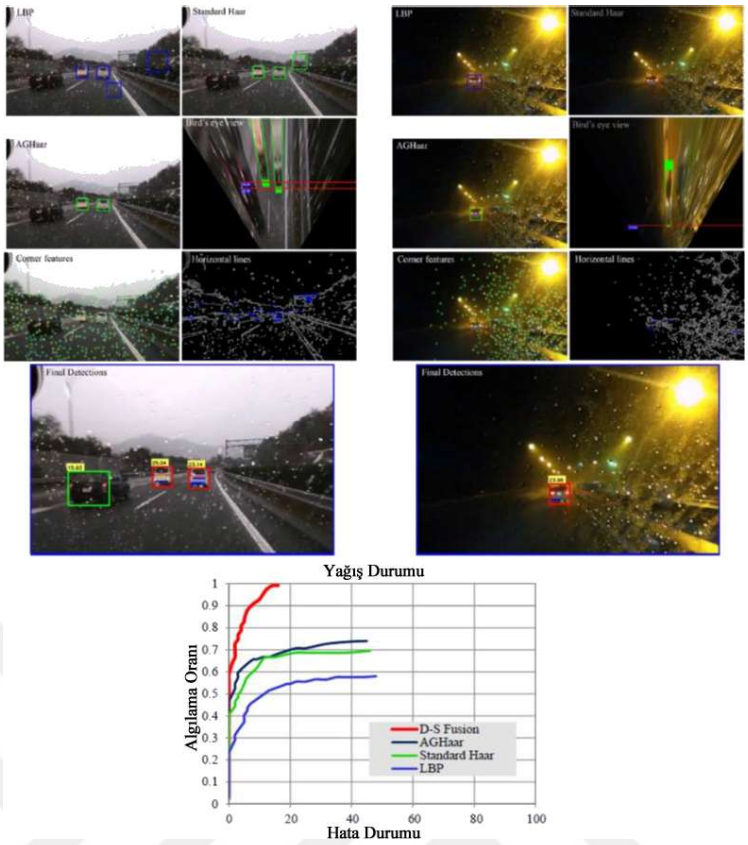
2.1.1.2. Sonuç

Önerdikleri yöntemlerin ve uygulanan D-S füzyon tekniğinin zaman etkinliği, yalnızca bir monoküler kamera kullanarak tüm çoklu veri füzyon sistemi için 25 ila 28 fps'lik gerçek zamanlı bir işlem yapılmasını sağlar. 8 GB RAM ve Windows 7 ile Corei5 2.7 GHz PC platformunda elde ettikleri sonuçlar sırasıyla Soldan sağa, yukarıdan aşağıya: İlk üç görüntü, tartışılan LBP, HAAR ve AGHaar yaklaşımları için algılama sonuçlarını gösterir, dördüncü görüntü AGHaar tabanlı görüntü için kuşbakışı mesafe tahmini sağlar. Beşinci görüntü köşe özellikleri sağlar, altıncı görüntü yatay kenar algılamanın sonucunu gösterir ve yedinci, önerilen veri birleştirme

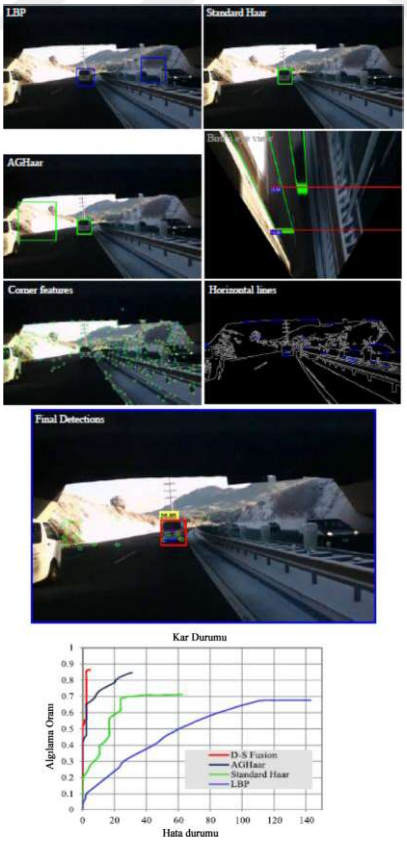
tekniğinden sonra araç algılama ve mesafe tahmininin nihai sonuçlarını gösterir. Tahmini mesafeler (m cinsinden), kırmızı sınırlayıcı kutuların sol üst tarafındaki sarı dikdörtgenlerde verilmiştir. Alttaki görüntü, gündüz ve gece için ROC eğrisini ve performans değerlendirmesini temsil eder [7]. Bu temsiller Şekil 2.3.de gün için araç tespiti, mesafe tahmini ve performans değerlendirmelerini, Şekil 2.4. Yağmurlu gündüz ve yağmurlu gece durumları için performans değerlendirmelerini ve. Şekil 2.5. Kar durumunda için performans değerlendirmelerini göstermektedir.



Şekil 2.3. Gün için araç tespiti [7].



Şekil 2.4. Yağmurlu gündüz ve yağmurlu gece için performans değerlendirmesi [7].



Şekil 2.5. Kar durumunda araç tespiti ve mesafe tahmini [7].

Araştırmalarında, belirli bir arka görüş aracı algılaması için bile, yüksek boyutlarla uğraşılması gerektiğini kanıtladı. Standart Haar, Kuşbakışı AGHaar ve LBP Son Tespitler Köşe özellikleri Yatay çizgiler Algılama Parametre oranı ve sağlam bir sonuç elde etmek için zorluklar. Araştırma, çok yönlü araç tespitleri gibi genel bir araç tespitine yönelseydi, sonuçlar henüz kabul edilebilir olmayabilirdi. Örneğin, KITTI kıyaslama web sitesi sonuçlarında listelenen en son başarılar ve son teknoloji çalışmalar, (ideal) gün ışığı koşullarında bile çoklu görüntülü araç algılama için %18,4 ila %74,9 arasında değişen çok düşük algılama oranları göstermektedir. [15]. Tablo 2.1'de yer alan sonuçlar, gerçek dünya uygulamalarının ihtiyaçlarından veya endüstri beklentilerinden hala geridedir. Felzenszwalb ve diğerleri tarafından önerilen son teknoloji nesne algılama yöntemi olan DPM'nin sınırlayıcı kutu yanlışlığını ve son derece yüksek hesaplama maliyetini de verilmiştir [16,17].

Tablo 2.1. Performans Değerleri [7]

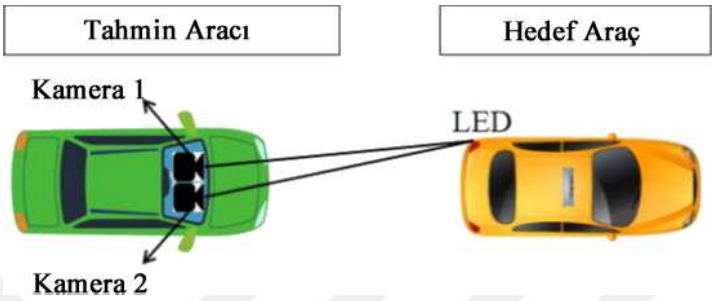
	LBP		Standart Haar		AGHaar		VSD & D-S Fusion	
	Kesinlik	Çağrı	Kesinlik	Çağrı	Kesinlik	Çağrı	Kesinlik	Çağrı
Gün verisi	62.8	81.0	73.4	88.2	89.0	97.5	95.2	99.6
Gece verisi	63.2	52.6	73.9	69.5	81.5	79.2	95.7	99.2
Yağış verisi	70.6	57.7	75.6	69.8	78.7	73.8	91.6	99.3
Kar verisi	48.2	67.0	69.4	71.4	84.1	84.8	97.2	87.5
Mesafe verisi	0	0	1.9	3.0	2.1	6.1	96.1	98.8
Karışık hava verisi	54.4	57.4	65.2	66.8	74.3	75.1	95.1	96.8

2.1.2. Görünür Işık İletişimine Dayalı Düşük Çözünürlüklü Bir Kamera Kullanarak Araçtan Araca Mesafe Tahmini

Görünür ışık iletişimi (VLC), kablosuz spektrum doygunluğunu azaltan gelişmekte olan bir teknolojidir ve bu yeni ve gelecek vaat eden kablosuz iletişim teknolojisi, önümüzdeki yıllarda önemli miktarda ilgi görecektir. VLC, düşük maliyeti, çevre dostu olması, bakım kolaylığı, yüksek güvenliği, kontrol edilebilirliği vb. nedeniyle popüler hale gelmiştir [18,19,20]. Işık Yayan Diyot (LED) tabanlı VLC sistemleri, araç kullanımı için uygundur ve bu tür kullanım için çok sayıda yol güvenliği uygulaması ve hizmeti bulunmaktadır [21]. Bu sistemler, sürücülerin güvende kalmasına ve trafik düzenlemelerine uymasına yardımcı olur. VLC sistemleri karayolu trafik güvenliği bilgileri sağlayabilir, olası kazaları en aza indirebilir ve yoldaki trafik akışını artırabilir [22]. VLC teknolojisi, dünya çapında birçok araştırma enstitüsünde ticarileştirilmek üzere geliştirilmektedir ve ayrıca MRC'nin sualtı VLC ekipmanı, Casio'nun Picapicamera, [23,24] gibi bazı ürünler piyasaya sürülmüştür.

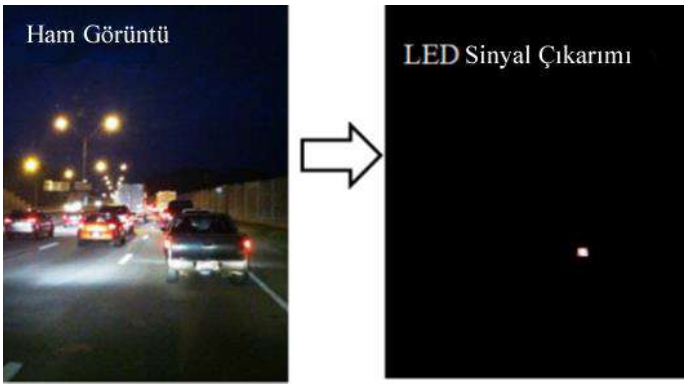
2.1.2.1. İşlemler

Sistem mimarisi Şekil 2.6'de anlatılmaktadır. Hedef aracın LED'lerinin resimlerini çekmek için tahmin yapan araca iki kamera bağlanmıştır. LED'leri içeren görüntüleri yakaladıktan sonra, görüntünün LED koordinatlarını belirlemek için veri çıkarma işlemi gerçekleştirilir [25]. Görüntülerin koordinatları daha sonra iki araç arasındaki mesafeyi belirlemek için kullanılacaktır.



Şekil 2.6. Sistem mimarisi [26]

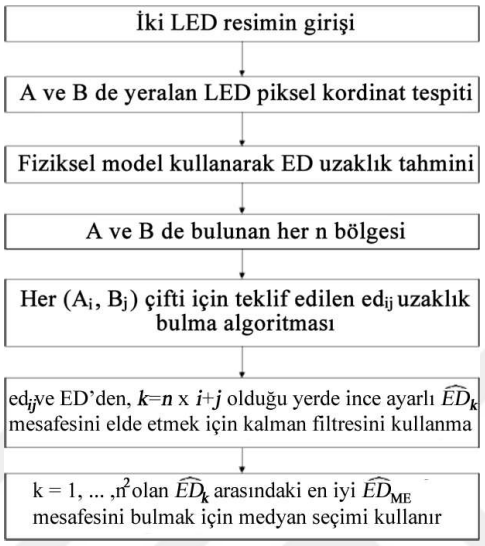
Görüntüden veri çıkarma işlemi Şekil 2.7'de gösterilmektedir. LED'lerin parlaklığının, arka planda ışık vermeyen nesnelerinkinden çok daha yüksek olduğu varsayılmaktadır. Bu nedenle, yakalanan görüntüdeki LED'lere karşılık gelen piksel değerleri, arka plana karşılık gelen değerden çok daha yüksektir. Sonuç olarak, görüntüdeki LED'lerin piksel koordinatları kolayca tespit edilebilir. Sokak lambaları gibi aydınlatma nesneleri için, araç LED'lerinin modülasyonlu yanıp sönme modeli sayesinde araçların LED'leri yine de kolayca ayırt edilebilir [26].



Şekil 2.7. LED sinyalinin görüntüden çıkarılması [26]

Önerilmiş çözünürlük telafisine genel bakış Şekil 2.8'de gösterilmiştir. Hedef araçtaki bir LED, araca monte edilmiş iki kamera tarafından yakalanmıştır. İki kameradan alınan görüntüdeki LED piksel koordinatları A ve B'yi elde etmek için işlenmiştir. Sistemin fiziksel modeli kullanılarak iki araç ED arasındaki mesafe tahmin edilmiştir. Bundan sonra, bu iki pikselin her biri n olası

bölgeye bölünmüştür. A_i ve B_j olası bölgelerinin her bir çifti için, LED'in iki görüntüdeki bu iki bölgede göründüğü varsayılmış ve araç mesafesi edij önerilen algoritma kullanılarak tahmin edilmiştir. Daha sonra ed_{ij} ve ED'den daha doğru mesafe ED_k elde etmek için Kalman filtre kullanılmıştır. Son olarak, n^2 mesafesi ED_k arasından en iyi ED_{Me} mesafesini bulmak için medyan seçim yöntemi kullanılarak sonuçlandırılmıştır [26].



Şekil 2.8. Önerilen çözüm telafisi. [26]

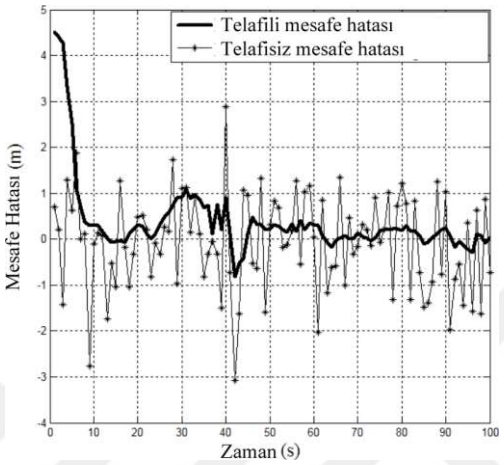
2.1.2.2. Sonuç

Sistem performansı, Matlab kullanılarak değerlendirilmiştir. Simülasyonda, LED görüntüleri, sensör boyutu, sensör çözünürlüğü, lens odak uzaklığı, kamera yönü, 2 kamera arasındaki gerçek mesafe, LED boyutu, LED parlaklığı ve Tablo 2.2'de verilen 2 araç arasındaki mesafe gibi sistem parametreleri ile simüle edilmiştir. Simüle edilmiş LED görüntüleri, LED piksel koordinatlarını elde etmek için işlenmiştir. Bu LED piksel koordinatları daha sonra iki araç arasındaki mesafeyi tahmin etmek için önerilen algoritmaya girdi olarak verilmiş, daha sonra önerilen algoritmanın performansını değerlendirmek için mesafe hataları hesaplanmıştır. Mesafe hataları, gerçek mesafe ile önerilen algoritma kullanılarak tahmin edilen mesafe çıkarılarak hesaplanmıştır [26].

Tablo 2.2. Simülasyon Değerleri [26]

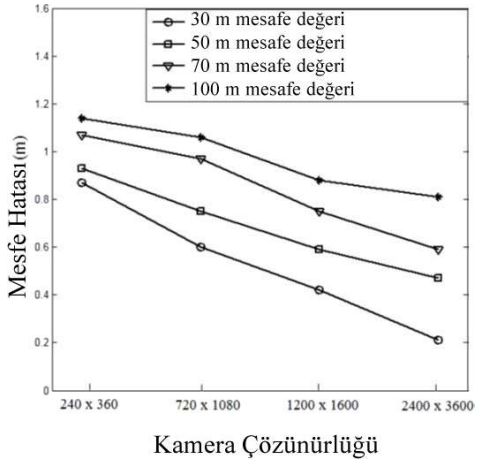
Parametre	Değer
Algılayıcı Boyu	2.4x3.6 mm^2
Çözünürlük	240 x 360 piksel
Odak Uzaklığı	35 mm
İki Kamera Arası Uzaklık	10 cm
LED Boyutu	10 x 20 cm^2
LED Parlaklığı	16000 cd/cm^2
İki Araç Arası Uzaklık	10 m~100 m

Önerilen sistemin genel performansı Şekil 2.9'de gösterilmektedir. Bu simülasyonda iki araç arasındaki başlangıç mesafesi 50 m'dir. Tahmini ve hedeflenen aracın ortalama hızları sırasıyla 45 ve 50 km=s'dir. İki aracın hızları zaman içinde %10 aralığında rastgele artabilir veya azalabilir. Ayrıca, hedef aracın, tahmin eden araç tarafından görülebilen yalnızca 1 LED'e sahip olduğu varsayılmıştır. İki aracın 100 saniye seyahat ettiği varsayılmaktadır. Bu süre boyunca, tahmin eden araç iki araç arasındaki mesafeyi tahmin etmeye devam eder.



Şekil 2.9. Genel Performans [26]

Önerilen çözünürlük telafisi yöntemi, daha yüksek çözünürlüğe sahip kameralarla kullanılabilir. Kameranın farklı çözünürlüğüne karşılık gelen önerilen telafinin performansı Şekil 2.10'de gösterilmektedir. Önerilen telafi yönteminin, yüksek çözünürlüklü kameralarla elde edilen tahmin doğruluğunu daha da iyileştirebileceği görülmüştür.



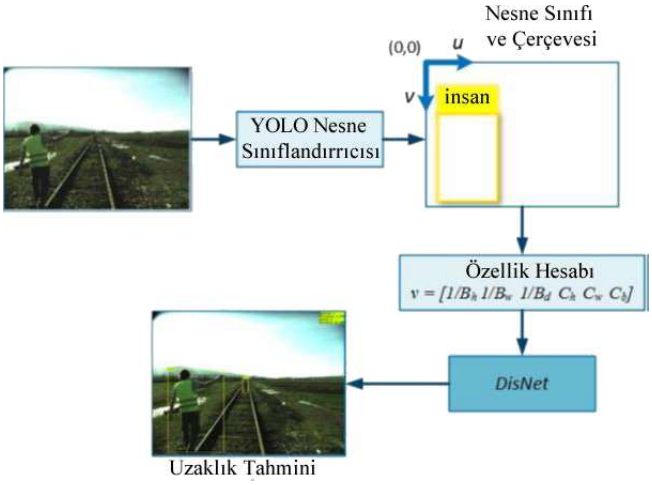
Şekil 2.10. Görüntü sensörünün çözünürlük telafisi ile mesafe hatası [26].

2.1.3. DisNet: Monoküler kameradan mesafe tahmini için yeni bir

Muhammad Abdul Haseeb ve arkadaşlarının yaptıkları bu çalışmada [27], engel tespit sistemine monoküler kameradan kamera ile görüntülenen nesneye olan mesafeyi tahmin etme yöntemi sağlayan bir makine öğrenmesi kurulumu sunulmuştur. Özellikle, H2020 Shift2Rail projesi SMART [28] kapsamında geliştirilmekte olan yerleşik çoklu sensör sisteminin nesnelere olan mesafeleri, lokomotifin önündeki demiryolu raylarındaki olası engellerin otonom olarak öğrenebilmesi için devam eden bir araştırmanın ön sonuçları verilmiştir. Sunulan mesafe tahmin sistemi, nesne ile kamera sensörü arasındaki mesafeyi öğrenmek ve tahmin etmek için kullanılan DisNet adlı Çok Gizli Katmanlı Sinir Ağı'na dayanmaktadır [27].

2.1.3.1. İşlemler

DisNet, giriş özelliklerinin YOLO [29] nesne sınıflandırıcıdan elde edilen nesne sınırlama kutularının parametrelerinin manuel olarak hesaplandığı ve çıktıların, kaydedilen sahnedeki nesnelere olan mesafelerin doğru 3D lazer tarayıcı ölçümleri olduğu, denetimli bir öğrenme tekniği kullanılarak eğitilmiştir. Sunulan DisNet tabanlı mesafe tahmin sistemi, demiryolu sahnelerinin görüntülerinin yanı sıra bir yol sahnesinin görüntüleri üzerinde de değerlendirilmiştir. Elde edilen sonuçlar, farklı tipte monoküler kameralarla görüntülenen nesnelere olan mesafelerin tahmini için kullanımını sağlayan önerilen DisNet sisteminin genel bir doğasını göstermektedir. DisNet tabanlı mesafe tahmin sisteminin mimarisi Şekil 2.11'de gösterilmektedir. Kamera görüntüsü, son teknoloji bir bilgisayar görüşlü nesne dedektörü YOLO'ya dayalı Nesne Sınıflandırıcıya girilir [29] COCO veri seti [30] ile eğitilmiştir. YOLO, CNN e dayalı hızlı ve doğru bir nesne detektörüdür. Çıktıları, görüntüdeki algılanan nesnelerin sınırlayıcı kutuları ve algılanan nesnelerin ait olduğu sınıfların etiketleridir. YOLO nesne sınıflandırmasından elde edilen kutuları sınırlayan nesneler daha sonra özellikleri, sınırlayıcı kutu parametrelerini hesaplamak için işlenir. Giriş özelliklerine bağlı olarak, eğitilmiş DisNet, nesnenin kamera sensörüne tahmini mesafesini çıkış olarak verir. Şekil 2.11'te gösterilen sistem mimarisinde, iki kişinin raylar üzerindeki mesafelerinin tahminine bir örnek gösterilmektedir.



Şekil 2.11. Nesne mesafesi tahmini için kullanılan DisNet tabanlı sistem.

2.1.3.2. Sonuç

RGB görüntülerinde DisNet nesne uzaklığı tahmini sonuçlarından bazıları Şekil 2.12 ve 2.13'te verilmiştir. Görüntülerde algılanan nesnelere (kişilere) olan tahmini uzaklıklar Tablo 2.3'te verilmiştir.

Tablo 2.3. Tahmini Uzaklık ve Gerçek Uzaklık

Şekil	Nesne	Raylı Sahne	
		Gerçek Uzaklık	DisNet Uzaklık tahmini
2.12	Kişi 1	100 m	101,89 m
	Kişi 2		99,44 m
2.13	Kişi 1	50 m	54,26 m
	Kişi 2	150 m	167,59 m
	Kişi 3	100 m	132,26 m
	Kişi 4	300 m	338,51 m

RGB kamera görüntüsünden bir tren yolu sahnesindeki nesnelere olan mesafelerin DisNet tahmini. Şekil 2.12'de 100 m'de tespit edilen kişilerin mesafe tahmini ve Şekil 2.13'te Sınırlayıcı kutularla kaplanmış büyütülmüş RGB görüntüsü ve sırasıyla 50, 100, 150 ve 300 m'de tespit edilen kişilerin mesafe tahmini yer almaktadır.

Şekil 2.12. Yüz metre tahmini

Şekil 2.13. Çoklu mesafe tahmini

2.1.4. CNN Tarafından Çıkarılan Derinlik İpuçları ile Monoküler Kamera Görüntülerinden Kör Nokta Engel Algılama

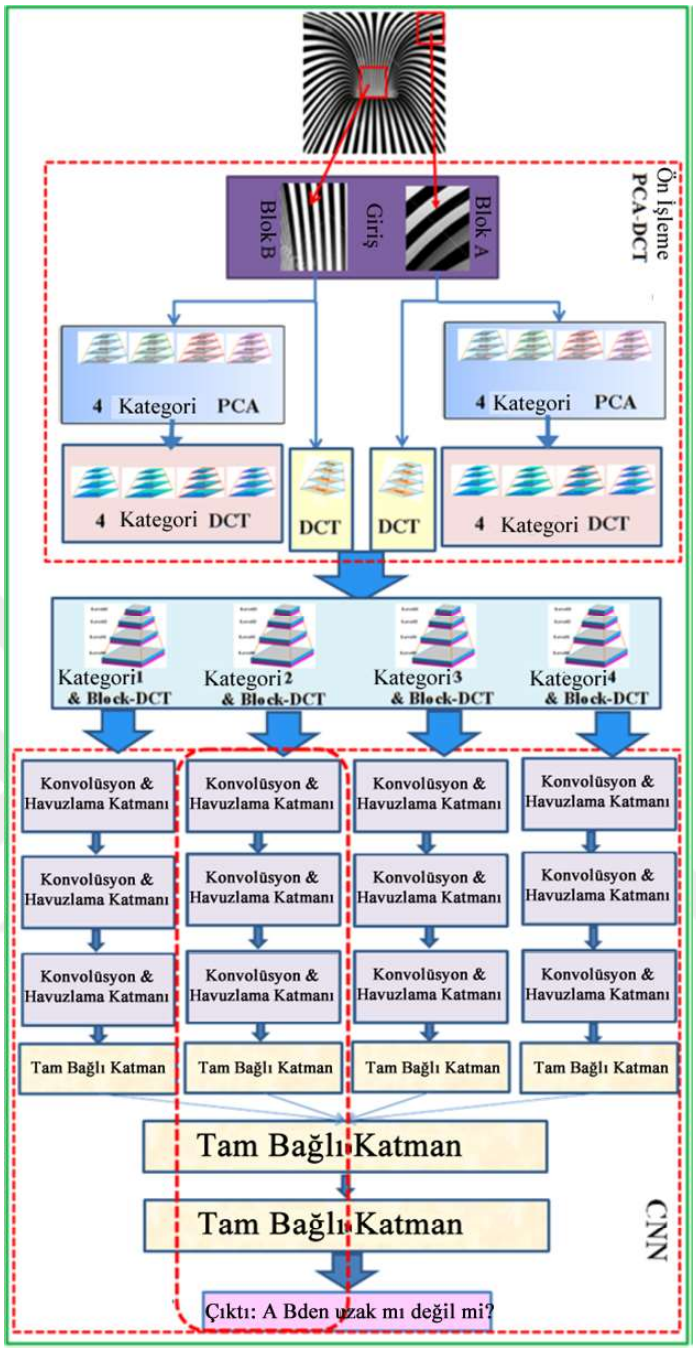
Yuxiang Guo ve arkadaşlarının yaptıkları çalışmada [31], bir monoküler kameradan alınan görüntüler, bir aracın yan görüş kamerası tarafından yakalanan kör nokta alanındaki engellerle ilgili derinlik bilgilerini tespit etmek için işlenmiştir. Görüntüdeki iki blok uzaklıklarına göre karşılaştırıldığında derinlik bilgisi "yakın" veya "uzak" sınıflandırma sonucu olarak verilmiş ve derinlik bilgisi kör nokta alanı tespiti amacıyla kullanılmıştır. Yapılan çalışmada, önerilen derinlik bilgisi, bulanıklık ipuçları ve doku ipuçlarının bir kombinasyonundan çıkarılmıştır. Derinlik bilgisi, tek bir görüntü içinde seçilen iki görüntü bloğunun özellikleri karşılaştırılarak

tahmin edilir. Yapılan ön deneye göre, ideal görüntüye yakın bir dizi ile derin öğrenmeyle eğitilmiş bir evrişimsel sinir ağı (CNN) modelinin iyi bir doğruluk elde ettiğini göstermiştir. Araç kör nokta alanında belirli bir eşiğe göre yakın ve uzak engelleri ayırt etmek için aynı CNN modeli uygulanmış ve umut verici sonuçlar elde edilmiştir. Önerilen yöntem, standart bir kör nokta kamerası kullanır ve diğer ek algılama cihazları olmadan güvenliği artırabilir. Bu nedenle, önerilen yaklaşım, sürücünün kör noktasındaki nesnelerin tespiti için araç uygulamalarında uygulanma potansiyeline sahiptir [32]. Yapılan çalışmada kameralar Şekil 2.14'da görüldüğü gibi montajlanmıştır.

Şekil 2.14.Kamera Konumları

2.1.4.1. İşlemler

Monoküler kamera uzak mesafede bulunan bir nesneye odak noktası yaptığında, kamera konumuna yakın olan görüntü içeriği bulanıklaşırken, daha uzaktaki alan belirgin doku ipuçlarına sahip olur. Bulanıklaştırma ipuçları ve doku ipuçları arasındaki bu korelasyon, derinlik ipuçları olarak yorumlanabilir. Monoküler derinlik ipuçları, doku ipuçları ve bulanıklık ipuçları ile karakterize edilen bir görüntüden elde edilir. Doku ipuçları, blok içindeki kenarların yoğunluğunu ifade ederken, bulanıklık ipuçları, bulanıklık derecesi ve kenarların keskinliğidir. Bu çalışmada, hesaplamaları basitleştirmek için temel olarak yerel ipuçlarına odaklanılmıştır. Bu noktadan başlayarak, doku ve bulanıklık ipuçlarının çıkarılmasının etkinliğini artırmaya çalışılmıştır. İlgili kavramlardan, ilk önce kaba-ince analizin çoklu çözünürlüklü görüntü temsili tartışılmıştır. PCA, görüntü verilerinden öznitelikleri çıkarmanın popüler bir yöntemi olduğu için ayrıntılı olarak açıklanmamıştır. PCA, önerilen yöntemde, görüntü kovaryans matrisinin öz değer ve vektör analizi yoluyla çıkarılan kenar çizgileri ile görüntü verilerinin hazırlanması için kullanılır. DCT, nesne görüntüsü için kenar çizgilerinin özellik dağılım yoğunluğunu belirlemek için uygulanır. Önerilen yaklaşım, PCA ve DCT'nin birleşik uygulamasını kullanır [32]. Önerilen yöntemdeki CNN modeli, önceki deneylerde kullanılan eğitilmiş modelle aynıdır [33]. Bu adımlar Şekil 2.15'de gösterilmiştir.



Şekil 2.15. PCA ve DCT Algoritması [32]

2.1.4.2. Sonuç Analizi

Monoküler derinlik ipuçlarını elde etmek için önerilen yöntem tanımlanmış, test edilmiş ve deneysel sonuçlar, etkinliğini doğrulamıştır. Derin öğrenme CNN modeli, ideal test koşulları altında tatmin edici bir doğruluk elde etmiştir. Araç BSD'si için aynı CNN modelini kullanan sonuçlara makul doğruluk göstermiştir. Bu nedenle, önerilen yöntemin daha karmaşık sürüş durumlarında uygulanma potansiyeline sahip olduğu çıkarımı yapılmıştır. Çeşitli avantaj ve dezavantajları Tablo 2.4'te yer alan değerlerle verişmiş ve aşağıdaki gibi açıklanmıştır:

(1) Dört kategorili PCA proses konfigürasyonu ile kaba-ince analizi birleştiren önerilen yaklaşım, derinlik ipucu çıkarmanın etkinliğini arttırır.

(2) Tespit, araç şekli profilleriyle sınırlı olmadığı için önerilen yöntem sağlamdır. Yöntem, göreceli derinlik bilgisini elde etmek için yerel ipuçlarını kullandığından, ego ana aracına yaklaşan herhangi bir engel tespit edilebilir.

(3) Önerilen yöntem görüntü rengiyle ilgili değildir. Ancak, algılama performansı gece çekilen görüntüler için yeterince iyi değil.

(4) Yerel ipuçları için tek bir görüntü hesaplamasında değişen pencere bloğu nedeniyle, mevcut mikroişlemci kapasitesi nedeniyle toplam hesaplama maliyeti hala ilgili. Bunun üstesinden 7 nm çipler ve özel yapay zeka çipleri kullanılarak gelinebilir.

Tablo 2.4. Test Sonuçları [32]

Test Resim Numarası	Doğruluk
1	% 74.13
2	% 78.13
3	% 80.12
4	% 81.38
Ortalama	% 78.44

2.1.5. DistanceNet: Tekrarlayan Evrişimli Sinir Ağı Kullanarak Monoküler Görüntülerden Katedilen Mesafeyi Tahmin Etme

Robin Kreuzig ve arkadaşlarının yaptıkları çalışmada, klasik monoküler vSLAM/VO [34] yöntemleri, ölçek belirsizliği probleminden, bu problemin. hibrit yaklaşımlarla, örneğin bir CNN tarafından tahmin edilen derinlik haritalarını kullanarak, derin öğrenme yöntemleri ekleyerek bu sorunun çözümünden bahsedip, ölçek tahminini, bir dizi sonraki görüntü için kat edilen mesafeyi tahmin etmeye dayandırmanın daha iyi bir yöntem olduğunu önerir. Yeni bir uçtan uca çoktan bire seyahat edilen mesafe tahmincisi önerisi yapılmıştır. Derin yinelenen evrişimli sinir ağı (RCNN) kullanılarak, bir dizi ardışık çerçevenin ilk ve son görüntüsü arasındaki kat edilen mesafe, DistanceNet'leri tarafından tahmin edilir [35 - 38]. Modelin CNN bölümünde geometrik özellikler öğrenilmiş ve bunlar daha sonra RNN tarafından dinamikleri ve zamansal bilgileri öğrenmek için kullanılmıştır. Ayrıca, mesafeyi tahmin etmek için sıralı regresyon kullanarak mesafelerin doğal düzeninden yararlanılır. KITTI [39] veri kümesi üzerindeki değerlendirme, yaklaşımları, mesafe tahmini açısından mevcut son teknoloji derin öğrenme poz tahmin edicilerinden ve klasik mono vSLAM/VO yöntemlerinden daha iyi performans gösterdiğini öne sürmüşlerdir. Böylece, DistanceNet'leri ölçek problemini çözmek ve mevcut ve gelecekteki klasik mono vSLAM/VO yöntemlerini geliştirmeye yardımcı olmak için bir bileşen olarak kullanılması önerilmiştir [40].

2.1.5.1. İşlemler

2.1.5.1.1. Eğitim

İlgili çalışmada ağın eğitimi aşağıdaki yazılım ve donanım tarafından yapılmıştır [40]:

- Python 2.7.15 ile PyTorch 1.0
- 12 GB RAM'li 3 NVIDIA Titan Xp
- 64 GB RAM'li AMD Ryzen Threadripper 1900X

2.1.5.1.2. KITTI Veri Kümesi

Geiger et al. [39], bir aracın 22 farklı sürüş dizisini içeren KITTI kilometre sayacı kıyaslamasını yayınlandı. Algoritmaların kıyaslamadaki performansını test etmek için, kesinlik tespiti verileri yalnızca ilk 11 dizi için kullanılmıştır. Ağlarının eğitiminde otoyolda kaydedilen tek dizi olan 01 dizisi yaklaşık 1100 görüntü içerdiği için eğitim seti dışında tutulmuş, onun yerine kırsal ve kentsel sekansların tamamı kullanılmıştır. Aracın kat ettiği mesafeyi art arda 10 karede toplayarak, ayrı sınıflara ayrılabilecek maksimum mesafeyi 15 metre olarak belirlenmiştir. KITTI dizileri 10 fps ile kaydedilmiştir ve bu nedenle seçtikleri zaman adımı uzunluğu ile tam olarak eşleşmiştir. Yaptıkları, ağın çıktısının yalnızca bir mesafe sınıfı değil, aynı zamanda metre/saniye cinsinden bir hız ölçümü olduğu anlamına gelmektedir [40].

2.1.5.2. Sonuç Analizi

Kullanılan sekanslar 03, 04, 05, 06, 07 ve 10 olmak üzere, klasik yöntemler için üst kısım ve derin öğrenme yöntemleri için alt kısım olmak üzere Tablo 2.5 iki bölüme ayrılarak verilmiştir. Her iki kısım için de en iyi sonuçlar vurgulanmıştır. DSO-stereo her şeyden daha iyi bir performans sergiliyor ancak bu bir sürpriz değil çünkü ölçek belirsizliğinin olmadığı bir stereo kamera kurulumu kullanıyorlar. Klasikler arasında en iyisi Bizim değerlendirmemizde mono-tabanlı yöntemler PMO tarafından sağlanmıştır. [40]

Yaptıkları DistanceNet kümesi ile karşılaştırıldığında, RMSE için daha kötü bir sonuca sahiptir, ancak doğrulukta beklenmedik şekilde yüksek bir değere sahiptir. Bu, bazı iyi ve bazı daha kötü sonuçları olan tutarsız bir yöntemin göstergesidir. Bu değerler Tablo 2.6'de yer almıştır [40].

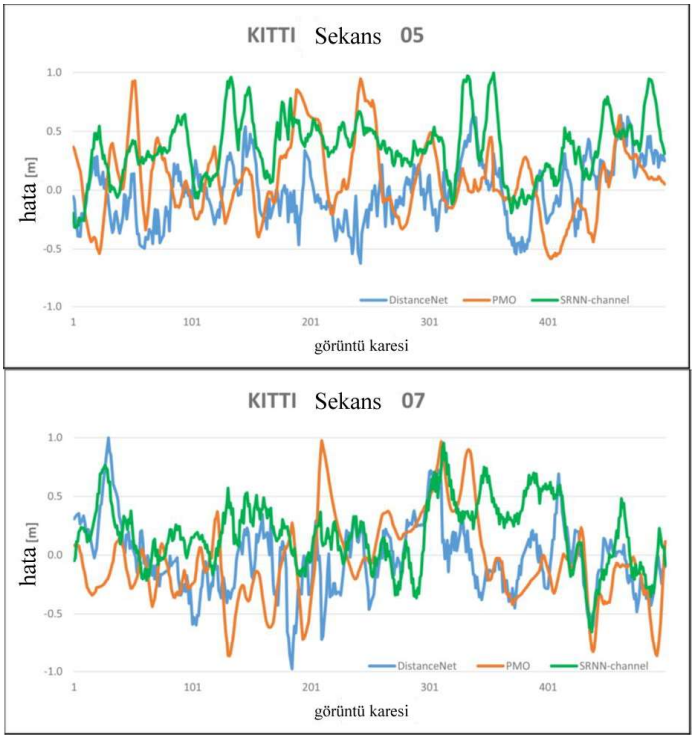
Tablo 2.5. KITTI üzerindeki yaklaşık sonuçlar [39]

Method	RMSE	Acc	AccDev
ORB-SLAM-mono	7.4623	0.0221	0.0368
DSO-mono	7.3854	0.0241	0.0452
PMO	0.7463	0.7183	0.9633
DSO-stereo	0.0756	0.9387	1.0
SRNN	0.6754	0.6121	0.9667
SRNN-se	0.6526	0.5801	0.9727
SRNN-point	0.5234	0.6267	0.9822
SRNN-channel	0.5033	0.6487	0.9873
DistanceNet-FlowNetS	0.5544	0.6292	0.9752
DistanceNet-Reg	0.5315	0.6848	0.9855
DistanceNet-LSTM	0.4167	0.6871	0.9896
DistanceNet-BCE	0.3925	0.7158	0.9930
DistanceNet	0.3901	0.6984	0.9916

Tablo 2.6. Seri 09 ve 10 un Ortalama Sonuçları [39]

Method	RMSE	Acc	AccDev
GeoNet	6.2302	0.0306	0.0544
SfMLearner	7.5671	0.0216	0.0505
DistanceNet	0.4624	0.6669	0.9841

Şekil 2.16. 05 ve 07 dizilerinin ilk 500 karesi için DistanceNet, SRNN-channel ve PMO hataları**.** Tahmini mesafenin gerçek mesafeden çıkarılmasıyla bir hata hesaplanır**.** "Pozitif hatalar gerçek mesafeyi aştı ve negatif olanlar ulaşmadı" [39].



Şekil 2.16 Sekans 05 ve 07

2.1.6. Kör Alan Tesiti için Derin Öğrenme Tabanlı Nesnelerin Mesafe Ölçüm Yöntemi

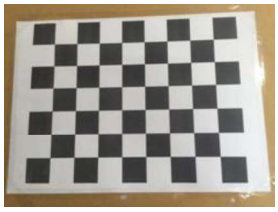
Jiaxu Zhang ve arkadaşlarının yaptıkları çalışmada, geri dönüş sürecinde şehir içi trafik kazalarının nedenlerinden biri görme alanı tıkanıklığıdır. Araç güvenliği ve zekası gereksinimlerini karşılamak için, derin öğrenme ve binoküler görüşe dayalı bir hedef mesafe ölçümü yöntemi önerilmiştir. Yöntem önce binoküler stereo görüş modeli kurar ve içsel dışsal ve dışsal parametreleri kalibre eder, görüntüdeki engel nesnelerini belirlemek ve konumlandırmak için Daha Hızlı R-CNN algoritmasını kullanır, ardından elde edilen eşleştirme noktalarını hedef nesnenin uzamsal koordinatları için kalibre edilmiş bir binoküler stereo modelinde değiştirir. Son olarak, engel mesafesi formül ile hesaplanır. Fiziksel testler yapmak için farklı konumlarda engellerin farklı açılardan fotoğraflarını çekin. Deneysel sonuçlar, bu yöntemin engel nesnesi tanımlama ve konumlandırmayı etkili bir şekilde başarabileceğini ve görme alanı körlüğünün sürüş güvenliği üzerindeki olumsuz etkisini iyileştirebileceğini göstermektedir [41].

2.1.6.1. İşlemler

P nesne noktaları, x eksen koordinatları ve M merkez noktası olmak üzere, iki boyutlu görüntü koordinatları aracılığıyla üç boyutlu konumun hesaplanmasını sağlamak için. Merkez eşleştirme noktaları x_1 ve x_2, üç boyutlu koordinatları hesaplamak için binoküler modelde (1) değiştirilir.

$$\begin{bmatrix} P_1 & -x_1 & 0 \\ P_2 & 0 & -x_2 \end{bmatrix} \begin{bmatrix} M \\ 1 \\ 1 \end{bmatrix} = 0 \qquad (1)$$

Formüldeki kameranın P_1 ve P_2 parametrelerinin kalibre edilmesi gerekir ve kalibrasyon sonucunun ölçüm doğruluğu üzerinde çok önemli bir etkisi vardır. Mesafe ölçüm sistemi, kameranın dahili ve harici parametre matrisini çözmek için binoküler görüş kalibrasyonu için referans olarak kalibrasyon plakasını kullanır. Kalibrasyon işleminde dürbün kameraları arasındaki konumlar ve açılar ayarlanır ve Şekil 2.17'de gösterilen dama kalibrasyon tahtası üzerinde farklı açılardan 28 fotoğraf çekilmiştir.



Şekil 2.17 Kalibrasyon plakası.

Köşe noktaları elde edildikten sonra, iki kameranın içsel parametreleri, Tablo 2.7'de gösterildiği gibi kalibrasyon yöntemi [42] kullanılarak çözülmüştür. Daha sonra sol ve sağ görüntüler arasındaki eşleştirme ilişkisi, stereo eşleştirme, yani dış parametre ile hesaplanmıştır. Döndürme matrisi R (2) ve kaydırma vektörü T (3). sonuçları denklemlerde verilmiştir.

Tablo 2.7. Parametreler

Parametreler	$\mathbf{f_x}$	$\mathbf{f_y}$	$\mathbf{c_x}$	$\mathbf{x_y}$
Sol Kamera	1083.08	1086.58	416.18	324.09
Sağ Kamera	1092.60	1096.39	421.79	257.48

$$R = \begin{bmatrix} 0.9976 & -0.0122 & 0.0677 \\ 0.0109 & 0.9997 & 0.0198 \\ -0.0679 & -0.0191 & 0.9975 \end{bmatrix} \quad (2)$$

$$t = [2.1858x10^2 \quad -2.2566 \quad 5.7440] \quad (3)$$

2.1.6.2. Sonuç

Test işlemi sırasında veri toplama sistemi, GY0814-3MP model iki CCD kamera, aynı parametrelere sahip bir USB 2.0 arayüzlü CCD endüstriyel dijital kamera ve 3 milyon piksele sahip bir endüstriyel lensten oluşmaktadır. Hedef mesafe ölçümünü elde etmek için bir 210mm*297mm kalibrasyon plakası, bilgisayar ve ilgili yazılımı seçin. Veri toplama platformu Şekil 2.18'de gösterilmiştir. İki kamera, farklı mesafelerin ve etkin aralıkların ihtiyaçlarını karşılamak için ayarlanabilir aralık, açı ve yükseklik ile platform üzerine sabitlenmiştir. Deneysel bölüm iki bölümden oluşmaktadır: İlk bölüm, derin öğrenme yoluyla sol ve sağ kameraların görüntüleme düzlemlerinde hedefin karşılık gelen görüntü noktalarını tespit etmektir; ikinci kısım, kalibre edilmiş kamera modeline görüntü noktalarını koyarak üç boyutlu koordinatları ve mesafeleri elde etmektir.

Şekil 2.18. Kameralar

Bu yazıda önerilen yöntemi doğrulamak için fiziksel ölçüm deneyleri yapılmıştır. Dürbün kameraları, farklı mesafelerde ve açılarda engellerin fotoğraflarını çeker. Elde edilen görüntü verileri, eğitilmiş olan hedef tespit ağına girilir. Model, engel türlerini ve x1 ve x2 merkez koordinatlarını otomatik olarak ayırt edebilir. Eşleşen x1, x2 noktaları ve kalibre edilmiş kamera projeksiyon matrisi P_1, P_2'yi denklem (1) ile değiştirerek, uzaydaki engelin üç boyutlu X koordinatı elde edilebilir. Mesafe formülü (4) (5) kullanılarak engeller ve araçlar arasındaki mesafe hesaplanabilir.

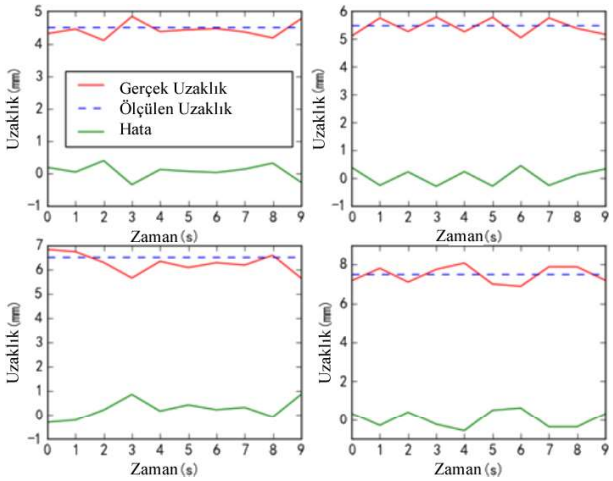
$$\cos\theta = \frac{v_1 v_2}{\|v_1\|\|v_2\|} \tag{4}$$

$$D = \sin\theta \,.\, v_1 \tag{5}$$

Mesafe ölçüm deneyi, dürbün kameradan 5-7,5 metre mesafedeki hareketli bir engel üzerinde gerçekleştirilmiştir. Tablo 2.8'de gösterildiği gibi, doğruluk, gerçek mesafe ile hata hesaplanarak analiz edilir. Deneysel sonuçlar metinsel olarak Tablo 2.8'de ve grafiksel olarak Şekil 2.19'da verilmiştir.

Tablo 2.8. Değerler

Gerçek Uzaklık/m	Ölçülen Uzaklık/m	Hata/m	Hata Oranı %
4.5	4.311	0.189	4.20
4.5	4.452	0.047	1.04
4.5	4.101	0.398	8.84
4.5	4.456	0.044	0.90
4.5	4.374	0.126	2.80
5.5	5.11	0.390	7.09
5.5	5.756	-0.256	4.65
5.5	5.271	0.229	4.16
5.5	5.382	0.118	2.15
5.5	5.264	0.236	4.29
6.5	6.814	-0.314	4.83
6.5	6.577	-0.077	1.71
6.5	6.283	0.217	3.34
6.5	6.329	0.171	2.63
6.5	6.073	0.426	6.55
7.5	7.171	0.329	4.39
7.5	7.804	-0.304	4.05
7.5	7.098	0.402	5.36
7.5	7.747	-0.247	3.29
7.5	7.56	0.06	0.80



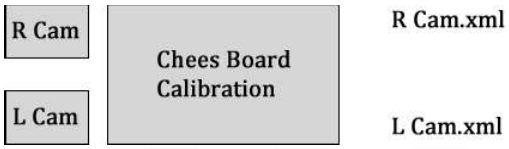
Şekil 2.19. Grafiksel Gösterim

2.2. Kaynak Araştırması Sonucu

Yapılan araştırmalarda görüldüğü üzere, bu tip sistemlerin yüksek işlem gücüne ihtiyaç duyduğu, bu nedenle yüksek maliyetli ve gerçek hayata tam olarak uygulanamaz olduğu anlaşıldığından yaptığımız çalışmada başarı oranının düşük olması beklenmektedir.

3. MATERYAL VE YÖNTEM

Sistem Gerçek zamanlı olabilmesi için Linux çekirdeği gerçek zamanlı olarak derlenecektir. Kameralardan alınan görüntü önce gürültüden arındırılacak ardından derin öğrenme ile hazırlanmış bir algoritma tarafından uzaklık değeri tahmin edilecektir. İlk aşamada satranç tahtası yöntemi kullanılarak kameralar kalibrasyon işlemine tabi tutulacaktır. Şekil 3.1'de ki gibi her bir kameranın lens gürültüsü ortadan kaldırılacak ve odak noktası xml dosyalarında kayıtlı olacak şekilde hesaplanacaktır.



Şekil 3.1. Kalibre Verisi

İkinci aşamada ilk aşamadan elde edilen kalibrasyon değerleri kameradan alınan her bir kareye uygulanacaktır. Bu uygulamanın matematiksel işlemi yoğun olduğundan GPU tarafından işlenmesi uygun olacaktır. Kalibre edilmiş görüntüler derinlik haritası oluşturmak için kullanılacak ve üçüncü aşama bilgisayarlı görü ile konvolüsyonel yapay sinir ağları yardımıyla geliştirilen nesne tanımlama algoritmasıyla tespit edilen nesnelerin yanında uzaklık tahminleri yer alacaktır.

3.1. Gerçek Zamanlı Sistemler

Gerçek zamanlı sistemlerde, bu tip bir programın doğru yürütülmesi mantıksal sonucun doğrululuğu ve bu sonucun belirli bir zaman çerçevesinde gerçekleştirilmesi gerekmektedir [43].

Bir sistemin gerçek zamanlı olabilmesi için görev zamanlama algoritmalarını destekleyebilmesi ve belirli bir zaman çerçevesinde işlemesi gerekmektedir. Literatürde birden fazla algoritma mevcuttur, sık kullanılan algoritmalardan (RM, EDF, PIP, PCP, SRP) en az birini tam anlamıyla desteklemesi gerekmektedir [44].

Gerçek zamanlı sistemlerin uygulama tipleri üç maddede halinde incelenebilir. Bu maddeler zaman limitlerine göre oluşturulmuştur [45].

Esnek uygulama: Bu tip uygulamaların amacı hizmet kalitesini (QoS) korumaktır. Buna en iyi örnek video karelerinin bir saniyede geçmesi gereken kare sayısını korumaktır. Bazı karelerin teslim sürelerinin kaçırılması insan veya nesneler üzerinde güvenlik açısından kritik sonuçlara yol açmadığından sadece kalitenin düşmesi anlamına gelir, fakat bu durumun kabul edilebilir bir kullanıcı deneyimi sağlaması esastır.

•Sıkı uygulama: esnek uygulamaya benzer, ancak herhangi bir durumda gerçekleşen işlem kaçırma çıktıyı geçersiz kılar. Bu tarihten sonra gerçekleşen sonuçların hiçbir değeri yoktur ve kullanılmamalıdır.

•Katı uygulama: genellikle gerçek zamanlı bir sitenden bahsedildiğinde bu tip sistemler akla gelir. Bu sınıfta son teslim tarihinin kaçırılması istenmeyen sonuçlara yol açacağından zaman kısıtlamalarını garanti etmek için tüm sistemin ve sistem kodlarının detaylı analizlerini gerektirir.

Bu tarz uygulamalar güvenlik açısından kritik uygulamalarda kullanılırlar. Havacılıkta kullanılan otomatik pilot uygulamaları bu sınıfa girer.

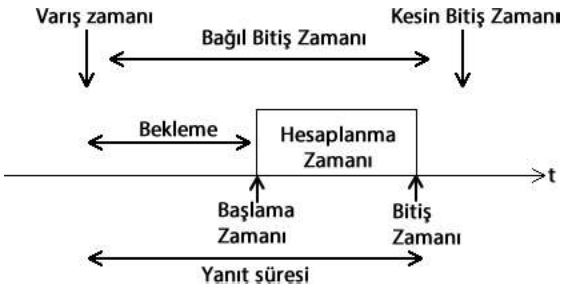
Gerçek zamanlı sistemleri daha iyi anlayabilmek için yapısına bakalım. Bu yapı için önemli olan bazı terimler şunlardır:

İş Cevap Süresi (Job Response Time): İşin sonlandığı zamandır. Serbest bırakma anında ölçülür.

Görev Cevap Süresi (Task Response Time): Tüm işler arasındaki maksimum yanıt süresidir.

Kritik An (Critical Instant): Bir görev için; en büyük görev yanıt süresini üreten varış zamanıdır.

Sistem modeli Şekil 3.2'de olduğu gibi, serbest bırakılma | varış zamanı (release | arrival time) ile işin kesin bitirilme zamanı (deadline) arasındaki süre, hesaplanma zamanı (execution time) olarak tanımlanmaktadır [46].



Şekil 3.2. Sistem Modeli

Bu bölümde sistem tanımlanması yapılmış olup, ikinci bölümde gerçek zamanlı Linux çekirdeğinden bahsedilmiş ve diğer bölümlerde çekirdek yaklaşımları gösterilmiştir.

3.1.1. Gerçek Zamanlı Linux Çekirdeği

RTLinux, robotları, veri toplama sistemlerini, üretim tesislerini ve diğer zamana duyarlı aletleri ve makineleri kontrol etmeyi mümkün kılan katı gerçek zamanlı Linux sürümüdür [5].

Yaklaşık 43 yıl önce araştırmacılar Bell Laboratuvarlarında Multi-Environment Real-Time (MERT) adında deneysel bir işletim sistemi gerçekleştirdi [48]. Bu işletim sisteminin hem gerçek zamanlı hem de genel amaçlı uygulamaları çalıştırması amaçlanmıştır. Ancak MERT tasarımcıları, her ikisini de destekleyebilecek tek bir işletim sistemi yapmaya çalışmak yerine, gerçek zamanlı bir işletim sistemi ile genel amaçlı (zaman paylaşımlı) bir işletim sisteminin birlikte çalıştığı bir sistem yapmaya çalıştılar.

Tasarımcılar bunu şu şekilde ifade etti:

"… Gerçek zamanlı işletim sistemi ile aynı makinede sofistike bir zaman paylaşım sisteminin mevcudiyeti, insan-makine ara yüzünün gerçek zamanlı süreçlere tasarlanmasında yararlanılabilecek güçlü araçlar sağlar."

Yani MERT'in tasarımcıları, gerçek zamanlı işletim sistemini gerçek zamanlı olmayan işletim sisteminden ayırarak, uygulama geliştiricilerin gerçek zamanlı olmayan işletim sisteminin hizmetlerini kullanmasına izin verebildiklerini iddia ettiler [47].

Dolayısıyla RTLinux, MERT'de bulunan temel fikrin bir varyasyonu, benim mütevazı görüşüme göre daha iyi bir varyasyonudur [47].

Bu çalışmada PREEMPT_RT yaması kullanılacağından diğer çekirdek yaklaşım tarzlarından bahsetmemiz gerekmektedir.

3.1.2. Yardımcı Çekirdek Yaklaşımları

PREEMPT_RT yamasına yönelik en yaygın alternatif yaklaşımlar, genellikle gerçek zamanlı iş parçacıklarının yönetiminden sorumlu ek bir işletim sistemi "pico-kernel" in varlığına dayanır. Eşdeğer olarak kokernel, pico-kernel, nano-kernel veya dual kernel (dk) tabanlı yaklaşımlar olarak adlandırılırlar. Buradaki temel fikir, piko-çekirdeğin donanım ve genel amaçlı Linux çekirdeği arasında bir katman olarak çalışmasını sağlamaktır [47].

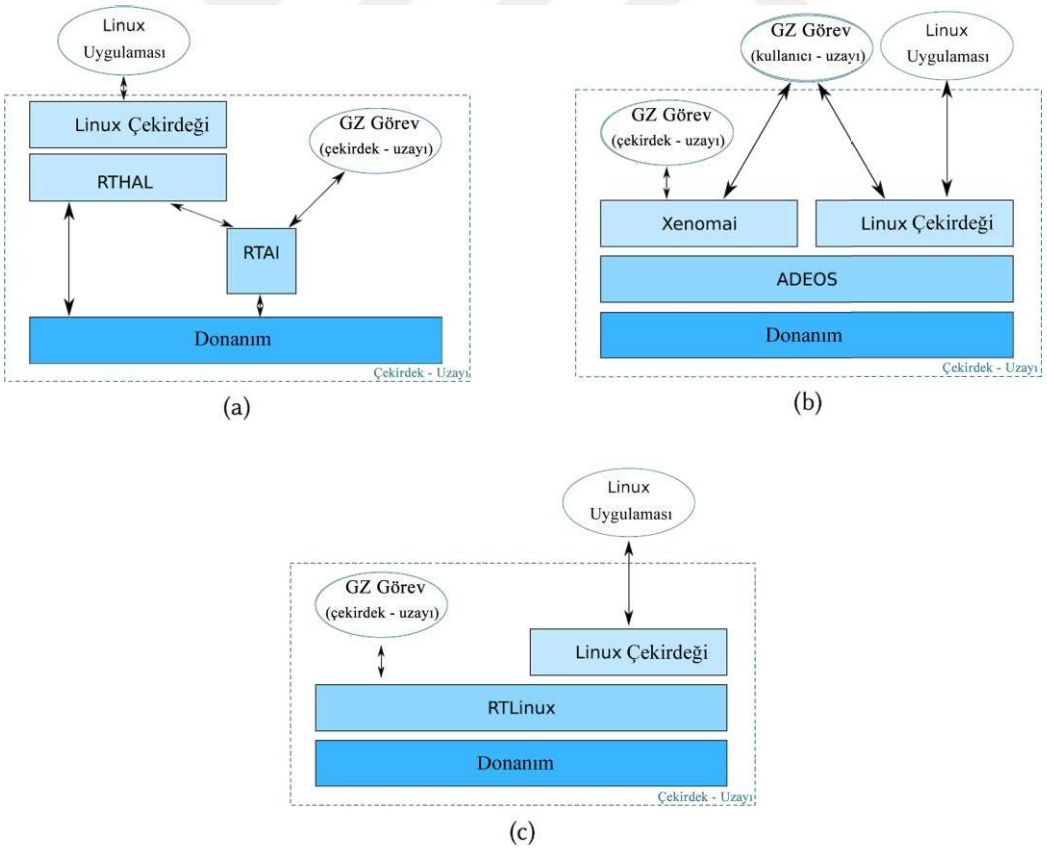
En yaygın kullanılan açık kaynaklı yaklaşımlar RTLinux, Xenomai ve RTAI dir. Bu yaklaşımlardan ikisi, Xenomai veya RTAI oluşan donanım kesintilerini (interrupt) yönetmek için Adaptive Domain Environment for Operating System (ADEOS) katmanını kullanır.Aslında ADEOS, her biri özel bir adres alanına ve kaynaklara sahip ayrı işletim sistemi alanları yaratır. Etki alanları arasında kaynak paylaşımı (ör. Cihazlar) yalnızca ADEOS aracılığıyla mümkündür [49].

•RTAI: Paolo Mantegazza tarafından Politecnico di Milano'da 2000 yılında geliştirilen RTAI [50], çekirdek alanı düzeyinde Linux gerçek zamanlı yeteneklerini sağlamayı amaçlayan bir ortak çekirdek yaklaşımıdır. Kullanıcı alanında LXRT arabirimi aracılığıyla sınırlı sayıda bu tür

yetenekler de sağlanır. Daha sonra RTAI, ADEOS alt sistemine taşındı. Bu durumda, Xenomai'den farklı olarak, kesintilerin işlenmesi esas olarak RTAI tarafından yönetilir.

•XENOMAI: Philippe Gerum tarafından 2002 yılında piyasaya sürülen Xenomai [51] kullanıcı alanında gerçek zaman sağlayan bir katmandır. Xenomai başlangıçta ADEOS katmanına dayanıyordu, ancak son sürümlerde yalnızca basitleştirilmiş bir kısmı, özellikle de I-pipe (interrupt pipeline) adı verilen kesme dağıtım alt sistemi kullanmaktadır.

•RTLINUX: Victor Yodaiken tarafından 1997 yılında geliştirildi, ilk kararlı sürüm 2000 civarında gerçekleşti ve ardından patent 2007'de Wind River Systems'a satıldı. RTLinux [47], Linux çekirdeğini gerçek zamanlı uygulamalarla birlikte tamamen öncelikli bir işlem olarak çalıştırır. Temel olarak, RTLinux, genel amaçlı çekirdeğin ne zaman çalıştırılacağına ve gerçek zamanlı görevlerin zor teslim tarihlerine göre ne zaman çalıştırılacağına karar veren bir zamanlayıcı uygular. Açıkçası, kesintileri RT görevine veya Linux çekirdeğine doğru şekilde iletmek için kesintiler için bir filtre görevi görür. Tüm RT görevleri, Linux çekirdeğinin aynı adres alanını paylaşır ve sonuç olarak yalnızca çekirdek-uzay evreleri olabilirler. Bu yaklaşımların farkı şekil 3.3'te grafiksel olarak yer almıştır [43].



Şekil 3.3. Üç Farklı Yardımcı Çekirdek Yaklaşımı (a) RTAI, (b) Xenomai ve (c) RTLinux [43]

.

3.1.3. Öncelikli Çekirdek Yaklaşımı

Gerçek zamanlı bir Linux çekirdeğinin ilk izleri, 1990'ların sonlarında literatürde bulunabilir. En başından beri çözülmesi en zor sorun, Linux çekirdeğinin doğrudan kesmeleri destekleyecek şekilde öncelikli kullanmanın imkansızlığından ibaretti. Aslında, bazı çekirdek rutinleri, gecikme açısından dramatik bir etkiye sahip olan uzun yürütme süreleriyle karakterize edildi.

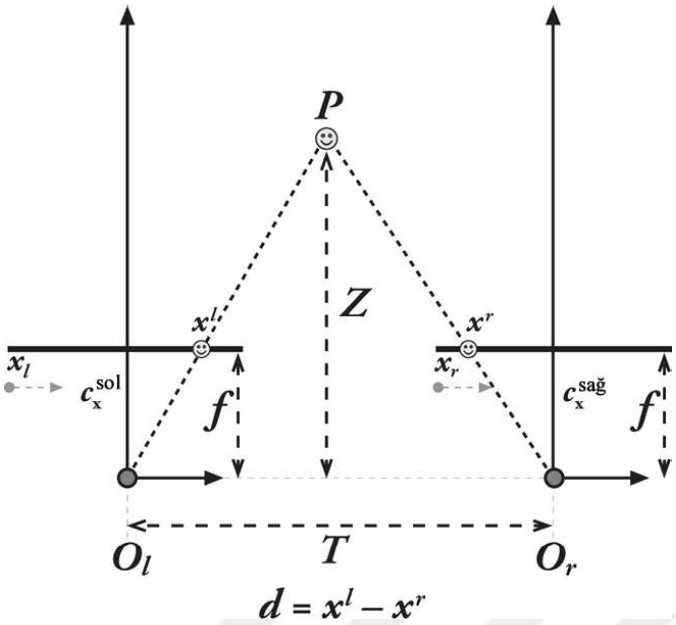
Finney tarafından 2001 yılında yayınlanan bir makaleye göre [52], eğer işlerin sonlanma zamanı milisaniyeler düzeyinde gerçekleşecek şekilde kalırsa, tek çekirdekli bir sistemde gerçek zamanlı görevler için saf bir Linux çekirdeği kullanmayı önerdi.

Linux çekirdeğinin PREEMPT_RT yaması aslında bir grup çekirdek geliştiricisi tarafından geliştirilen bir dizi yamadır. Proje Ingo Molnár tarafından başlatıldı ve ilk sürüm 2.6.11 çekirdek sürümüne dayanıyordu (Mart 2005).

Böylelikle bir yama ile hali hazırda kullanılan Linux çekirdeği öncelikli çekirdek modeline çevrilebiliyordu.

3.2. Çift Görü

İlk olarak Şekil 3.4'te gösterildiği gibi "ideal" bir stereo donanım kabul edilmiştir. Bu durumda, mükemmel bir şekilde bozulmamış, hizalanmış ve ölçülmüş bir sisteme sahip olduğumuzu varsayıyoruz. Görüntü düzlemleri birbiriyle tam olarak paralel optik eksenlere sahip iki kamera, birbirinden ayrı ve eşit odak uzunluklarına sahiptir. ($f_l = f_r$). Ayrıca, ana noktaların c_x^{sol} ve $c_x^{sağ}$ ilgili sol ve sağ görüntülerinde aynı piksel koordinatlarına sahip olacak şekilde kalibre edildiğini varsayalım [53].

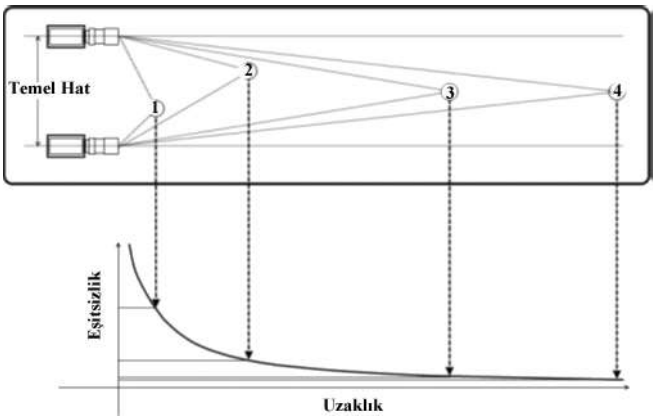


Şekil 3.4. İdeal Çift Kamera Düzeneği [53]

P nesne konumu, Z nesne uzaklığı, T iki kamera arası bilinen uzaklık, f odak uzaklığı, x^l sol görüntü x eksen iz düşümü ve x^r sağ görüntü izdüşümü olmak üzere. Bu basitleştirilmiş durumda, derinliğin (Z) eşitsizlikle ters orantılı olduğunu görebiliriz. T ve f değişkenlerinin bilindiğinden yola çıkarak Z (4) denklemi elde edilir [53].

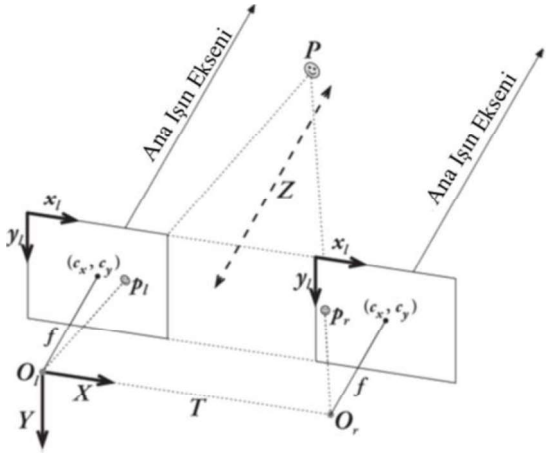
$$\frac{T-(x_l-x_r)}{Z-f}=\frac{T}{Z} \quad \Rightarrow \quad Z=\frac{f.T}{x_l-x_r} \tag{4}$$

Derinlik eşitsizlikle ters orantılı olduğundan, bu iki terim arasında açıkça doğrusal olmayan bir ilişki vardır. Eşitsizlik sıfıra yakın olduğunda, küçük eşitsizlik farkları büyük derinlik farkları yaratır. Eşitsizlik büyük olduğunda, küçük eşitsizlik farklılıkları derinliği fazla değiştirmez. Sonuç olarak Şekil 3.5'in açıkça gösterdiği gibi, stereo görüş sistemlerinin yalnızca kameraya daha yakın nesneler için yüksek derinlik çözünürlüğüne sahiptir [53].



Şekil 3.5. Eşitsizlik Uzaklık Eğrisi [53]

Stereo görüş için OpenCV'de kullanılan iki ve üç boyutlu koordinat sistemlerini Şekil 3.6'da gösterir. Sağ elini kullanan bir koordinat sistemi olduğuna dikkat edin: sağ işaret parmağınızı x ekseni yönünde çevirir ve sağ orta parmağınızı y ekseni yönünde bükerseniz, başparmağınız yönünü gösterecektir. ana ışın. Sol ve sağ görüntüleyici pikselleri, görüntüde sol üstte görüntü orijinlerine sahiptir ve pikseller sırasıyla (xl , yl) ve (xr , yr) koordinatlarıyla gösterilir. İzdüşüm merkezleri Ol ve Or'dadır ve ana ışınlar görüntü düzlemini ana noktada (merkezde değil) (cx, cy) kesişir. Matematiksel düzeltmeden sonra, kameralar sıraya göre hizalanır (eş düzlemli ve yatay olarak hizalanır), birbirlerinden T ile yer değiştirir ve aynı odak uzaklığına f sahiptir. Bu düzenleme ile mesafe için çözmek nispeten kolaydır.

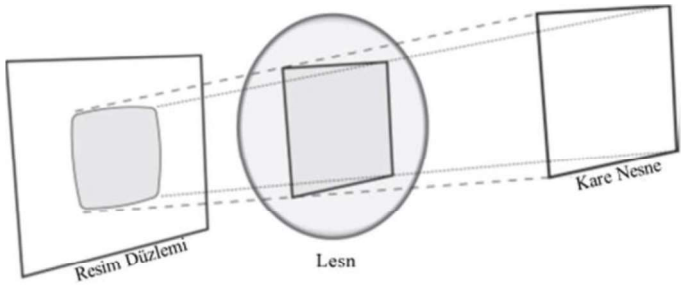


Şekil 3.6. OpenCv Kordinat Sistemi [53]

3.2.1. Lens Gürültüsü

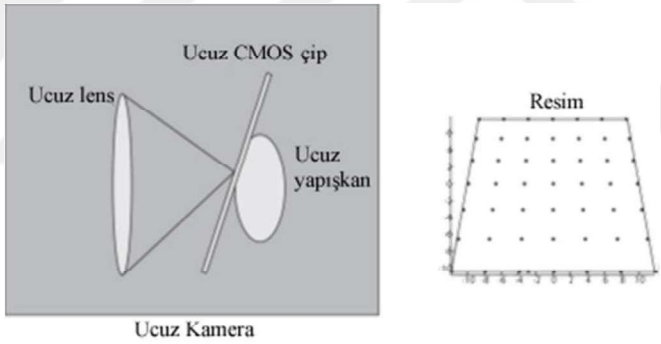
Teoride Lensler gürültüsüz olabilir fakat gerçek hayatta lensler kusursuz olamadığından bu olay pekte mümkün değildir. Gürültünün temel nedeni üretim kaynaklıdır. Gerçek kameraların lensleri, genellikle görüntüleyicinin kenarlarına yakın piksellerin konumunu fark edilir şekilde

bozar. Bu şişkinlik olayı "fıçı" veya "balıkgözü" etkisinin kaynağıdır. Radyal gürültünün neden oluştuğuna dair örnek Şekil 3.7'de vermektedir [53].



Şekil 3.7. Radial Bozulma

İkinci en büyük yaygın bozulma, teğetsel bozulmadır. Bu bozulma, merceğin görüntüleme düzlemine tam olarak paralel olmamasından kaynaklanan üretim hatalarından kaynaklanır. Genelde ucuz kameralar bu gürültüye sahiptir. Teğetsel bozulma örneği Şekil 3.8'de görülmektedir [53].



Şekil 3.8. Teğetsel Bozulma

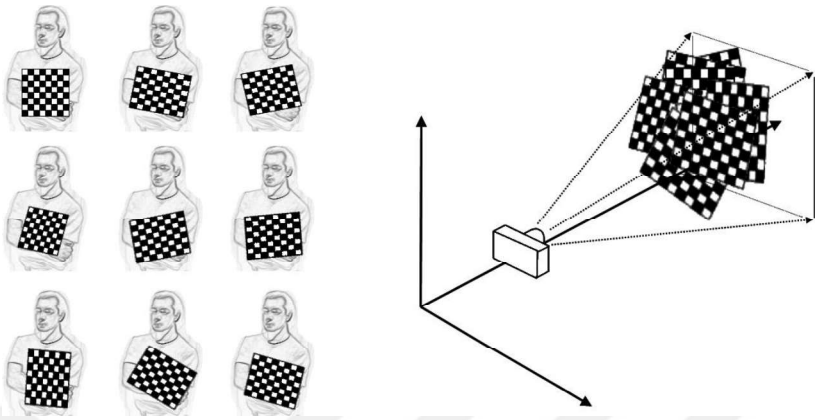
3.2.2. Kalibrasyon

Kameranın ve kamera bozulmalarının matematiksel tanımlarını yaptığımızdan bu sorunları matematiksel bir modelle çözebiliriz [54]. OpenCv bu işlem için cv::calibrateCamera() isminde bir fonksiyon ile gelmektedir [55].

3.2.2.1. Kalibrasyon Tahtası

Temelde satranç tahtasıdır. Literatürdeki bazı kalibrasyon yöntemleri, üç boyutlu nesnelere dayanır (örneğin, işaretleyicilerle kaplı bir kutu), ancak düz satranç tahtası desenleriyle uğraşmak çok daha kolaydır; diğer şeylerin yanı sıra, hassas üç boyutlu kalibrasyon

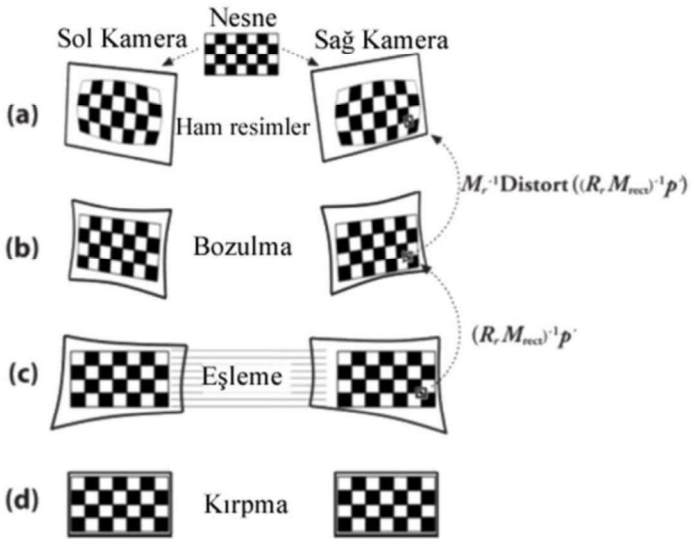
nesneleri yapmak (ve depolamak ve dağıtmak) oldukça zordur. Çeşitli yönlerde tutulan bir satranç tahtasının görüntüleri Şekil 3.9'da (solda), bu görüntülerin küresel koordinatlardaki konumlarını (kameraya göre) ve kameranın özünü tamamen çözmek için yeterli bilgi sağlar [53].



Şekil 3.9. Kalibrasyon

3.2.2.2. Çift Kamera Tek Düzlem Oluşturma

İki kamera arasında oluşturulan görüntü düzleminin tek görüntü düzlemi gibi hesaplanarak ele alınması işlemidir. Bu işlem Şekil 3.10'da adım adım incelenirse, ilk adımda ham görüntüler alınır (a), görüntünün lens gürültüsü temizlenir (b), görüntülerde örtüşen alanlar hizalanır (c) ve son olarak odaklanma sağlamak için görüntüler kırpılır(d).



Şekil 3.10. Görüntü Düzeltme ve Birleştirme [53]

3.3. Kullanılan Donanım ve Bileşenleri

Bu bölümde kullandığımız donanımların tanımları, özellikleri ve ürettiğimiz parçaların detayları yer almaktadır.

3.3.1. Tek Kart Bilgisayar

Single-board computer (SBC) (tek kart bilgisayar), giriş seviyesinde kullanılan ev herhangi bir bilgisayarına kıyasla daha az güç tüketen ve daha az bir çalışma kapasitesi olan bilgisayarlardır. ARM işlemcili ve x86 işlemcili olmak üzere ikiye ayrılır [56].

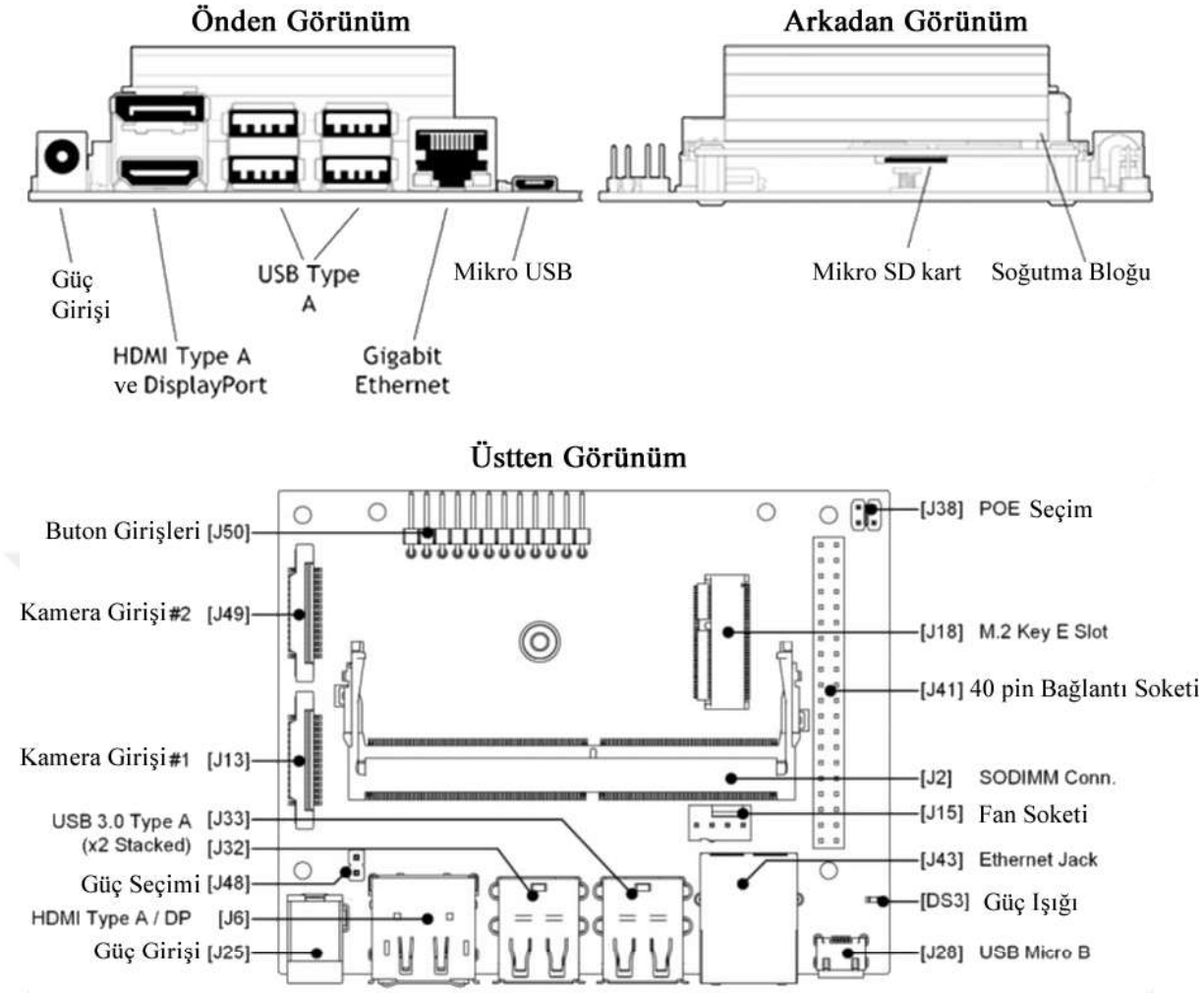
Nvdia Jetson Nano tek kart üzerinde birleştirilmiş Nvdia ekran kartı destekli mini bilgisayar sistemidir. Bu uygulamada 4Gb ram ve iki adet kamera desteği sağlayan modeli kullanılmıştır. Normal bilgisayarda yer alan CPU, GPU ve RAM modülleri tümleşik şekilde gelmektedir. Üzerinde dahili depolama alanı ve depolama için sata vb. bağlantı ara yüzlerine sahip olmadığından dolayı üzerinde çalışılan mikro Sd depolama biriminin yüksek okuma/yazma hızına sahip olmalıdır [57].

Bu cihaz çoğu ulusal/uluslararası yarışmalarda otonom araç vb. kategorilerinde kullanılmaktadır. Cihazın teknik özellikleri Tablo 3.1'de verilmiştir [57].

Tablo 3.1. Teknik Özellikler

Özellik	Açıklama
Grafik İşlemci	Maxwell 128 çekirdek
Merkezi İşlemci	ARM A57 Dört çekirdek @ 1.43 GHz
Bellek	64-bit LPDDR4 4 GB 25.6 GB/s
Depolama	microSD (not included)
Vidyo Kodlama	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Vidyo Onatma	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)
Kamera	2x MIPI CSI-2 DPHY lanes
Bağlantı	Gigabit Ethernet, M.2 Key E
Ekran	HDMI and display port
USB	4 port USB 3.0, USB 2.0 ve Micro-B
Diğer	GPIO, I2C, I2S, SPI, UART
Mekanik	69 mm genişlik , 45 mm derinlik, 260-pin edge bağlantı

Aşağıdaki Şekil 3.11'de x ve y de bordun iki boyutlu grafiksel görüntüleri verilmiştir.

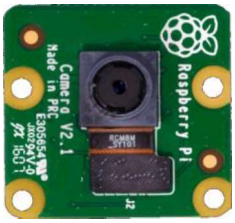


Şekil 3.11. Grafiksel Görüntü

3.3.2. Kamera

Yeni kamera aparatı Şekil 3.12'de yer alan Raspberry Pi Kamera Modülü v2,orijinal kameranın yerini 2016 yıldan itibaren devraldı. Bu yeni modül Sony firmasının IMX219 kodlu 8Mp destekli bir sensörle gelmektedir. Kamera Modülü, yüksek çözünürlüklü videoların yanı sıra sabit fotoğraflar yakalamak için kullanılabilir. Amatörler için kolay kullanım sunar [58].

Bu kamera J13 ve J49 bağlantı noktalarına tam uyumludur. Bu çalışma kapsamında şekilde bulunan kameradan 2 adet kullanılmıştır.



Şekil 3.12. RaspberryPi V2 Kamera [58]

Kameranın üretici tarafından verilen matematiksel değerleri Tablo 3.2'de yer almaktadır. Bu değerler programlama aşamasında kullanılmıştır [58].

Tablo 3.2. Sensör bilgileri

Tanım	Değer
Sensör çözünürlüğü	3280 × 2464 piksel
Sensör görüntüleme alanı	3.68 x 2.76 mm (4.6 mm köşeden köşeye)
Piksel boyutu	1.12 µm x 1.12 µm
Optik senör boyutu	1/4"
Odak Uzaklığı	3.04 mm
Yatay görüş alanı (Hfov)	62.2 derece
Dikey görüş alanı (Vfov)	48.8 derece

OpenCv de bu formülleri kullanabilmemiz için focal_lenght değerini piksel cinsine çevirmemiz gerekmektedir. Aşağıda bu işlem için iki adet denklem (5,6) verilmiştir [59].

$$focal_{pixel} = \frac{focal_{mm}}{sensor\_width_{mm}} image\_width_{pixel} \quad (5)$$

$$focal_{pixel} = \frac{image\_width_{pixel} * 0.5}{\tan(H_{fov} * 0.5 * {}^{\pi}/_{180})} \quad (6)$$

3.3.3. Kamera Yerleşimi

Sistem tasarımında önemli ölçüt kameraların yerleşimidir. Bu çalışmamızda insanların iki göz arasındaki uzaklığın ortalamasına yakın bir değer baz alınmıştır. Pupillary distance (PD) Şekil 3.13'te görüldüğü gibi gözbebeklerinin merkezleri arasındaki uzaklıktır. Ortalama bir yetişkinin PD'si 54-74 mm, çocukların ise 43-58 mm arasındadır [60].

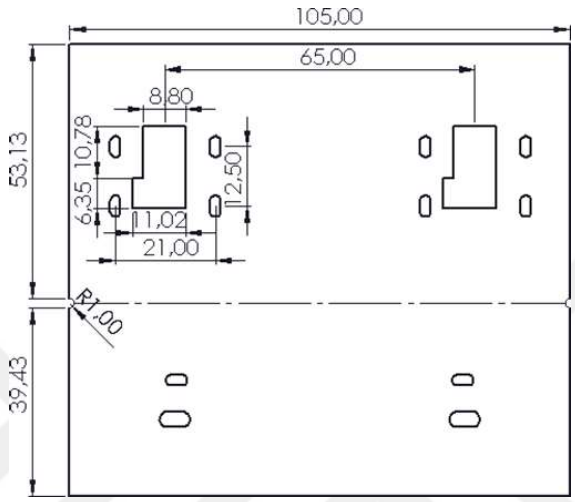
Şekil 3.13. Gözbebekleri Arası Mesafe

Bu aralıklardan yola çıkarak iki kamera merkezi arasının uzaklığı 65 mm olarak seçilmiş ve stant 2 mm saç metalden imal edilmiştir.

3.3.3.1. Montaj Parçası

Kamera yerleşim modelimize göre aşağıda ki parçanın kesim, büküm ve boyanma işlemlerine müteakip çalışmamızda kullanılmıştır.

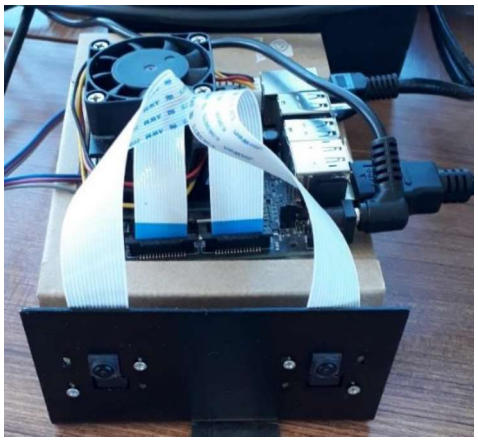
İki kamera arası uzaklık olarak 65mm kabul edilen parçanın çizimi Şekil 3.14'te görüldüğü gibidir.



Şekil 3.14. Metal Parça Ölçüleri

3.3.3.2. Montaj

Rasbpery pi kameraları metrik 2 vidava ve civatalar ile metal parçaya sabitlenmiştir. Kamera kabloları kamera soketlerine takıldıktan sonra Jetson Nano kutusu çift taraflı bant yardımıyla metal plakaya yapıştırılmıştır. Montaj görünümü Şekil 3.15'te görüldüğü gibidir.



Şekil 3.15. Montajlı Görünüm

3.4. Kullanılan Yazılım ve Bileşenleri

3.4.1. Gnu/Linux

Linux işletim sistemi, kendinden önceki işletim sistemi olan UNIX mimarisine ve POSIX standartlarına uygun olarak sıfırdan C dili kullanılarak 1991 yılında Finlandiyalı Linus Torvalds tarafından üniversite öğrencisi olduğu zamanda yazılmaya başlanmıştır. UNIX mimarisi sadece geliştirme aşamasında esinlenme amaçlı kullanılmış olup Linux içinde Unix'ten alınmış herhangi bir kod içermemektedir. Geliştirilmiş yazılım, kullanıcı araçlarından mahrum sadece bir çekirdek yazılımıydı.

Linux'tan önce 1984 yıllında Richard Stallman MIT'deki görevinden ayrılarak UNIX mimarisiyle uyumlu GNU Tasarısı adını verdiği yeni bir işletim sistemini özgür yazılım projesi çatısında geliştirmeye başlamıştır [61]. Bu süreç 1991 yılına geldiğinde kullanıcı araçları hazır olan ancak çekirdek yazılımı eksik kalmış bir haldeydi.

Linus Torvalds tarafından geliştirilen çekirdek yazılımı, daha çok geliştiricinin ulaştırması ve katılımcı desteği alabilmesi adına 1992 yılında özgür yazılım biçiminde GNU Genel Kamu Lisansıyla yaymaya karar verdi. Bu sayede bağımsız başlatılan Linux çekirdeği ve GNU Tasarısı projeleri yarım kalan kısımlarını birleştirerek tamamlamış ve kararlı bir işletim sistemi sürümü olarak kullanılabilir hale evirilmiştir [62]. Yapılan işletim sistemi "Özgür UNIX Benzeri" başlığı altında 1994 yılında GNU bülteninde duyurulmuştur [62].

3.4.2. Jetson Nano Developer Kit SD Card Image,

Ubuntu 18.04 LTS Linux kernel 4.9 sürüm numarası ile hazır bir şekilde gelmektedir. Linux dağıtımı olmasının çok sayıda avantajı vardır. Bunlardan bazıları, C dili ile donanıma doğrudan erişilebilmeye imkan tanıması, açık kaynak olmasından dolayı çekirdeğe müdahale edilerek yeni özellikler kazandırılması, sürekli güncelleme desteği, çoklu sürücü desteği gibi başlıca özellikleri sağlamaktır [63].

3.4.2.1. Jetson Nano Sistem Kurulumu

Kurulumlara geçmeden önce en az 32GB UHS-1 veya daha hızlı bir mikro SD kart temin ediniz. Kurulacak sistem tamamen Sd kart üzerinden çalışacağından Sd kartınızın okuma/yazma hızı performansı doğrudan etkileyecektir [64].

3.4.2.2. Cihazı Çalıştırma

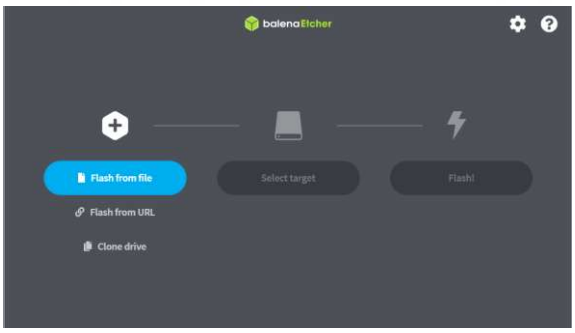
Cihazı çalıştırmadan önce, cihazımıza uygun Linux işletim sistemini ilgili linkten indirelim [64]. Bu dosya yaklaşık 6Gb büyüklüğündedir. Şekil 3.16'da görüldüğü üzere ilgili imajı seçilmelidir.

Title	Version	Release Date
› Jetson Nano Developer Kit SD Card Image	4.5	2021/01/21
› Jetson Nano 2GB Developer Kit SD Card Image	4.5	2021/01/21

Şekil 3.16.İmaj ve Tarihleri

3.4.2.3. İmajı Yazma

Yazma işlemi [65] için bu belgede Şekil 3.17'de yer alan "Etcher" uygulamasını kullanılmıştır. İşletim sistemine göre olan güncel sürümünü ilgili linkten [66] temin edip çalıştırdık. Açılan uygulamada İndirdiğimiz imaj dosyasını "Flash from file" seçeneğinden seçerek yazma işlemine geçtik. "Select target" seçeneğinden SD kart yolunu seçip "Flash" seçeneğinden yazma işlemini başlattık Yazım işlemi bellek hızıma göre 15 dk sürdü.



Şekil 3.17. Etcher Yazılımı

3.4.2.4. Cihaz Bağlantılarının Yapılması

Cihazımızı iki farklı şekilde bağlantısını yapabiliriz.

İlki için klavye, fare,5V/2A usb adaptör, micro usb kablo ve HDMI monitöre ihtiyacımız vardır. İşlem adımları 3.4.2.4.2'den başlayıp 3.4.2.4.4'de kadar sürmektedir.

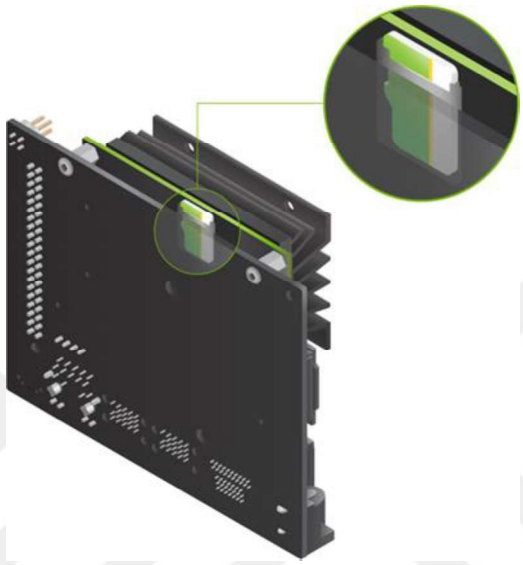
İkincisi için bir bilgisayar, micro usb kablo ve 5V/4A güç adaptörüne ihtiyacımız vardır. İşlem adımları 3.4.2.4.5'ten başlayıp 3.4.2.4.7'ye kadar sürmektedir.

*Tüm işlemlerden önce SD kartı doğru yerleştirdiğinize emin olun.

*Tüm bağlantıların takıldıktan sonra güç bağlantısını yaptığınıza emin olun.

3.4.2.4.1. SD Kartın Yerleştirilmesi

Şekil 3.18'de görüldüğü gibi hazırlanmış Sd kartı Jetson Nano nun işlemci kısmına yerleştirdik.



Şekil 3.18. SD Kart Takma

3.4.2.4.2. Ekran Bağlanması

Cihazın grafiksel arayüzüne ulaşılabilmesi için HDMI destekli bir monitör Şekil 3.19'da ki gibi bağlanmıştır.

Şekil 3.19. HDMI Ekran Bağlantısı

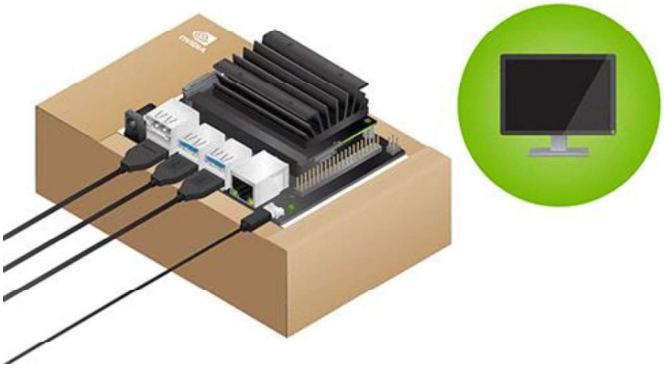
3.4.2.4.3. Klavye ve Fare Bağlantısı

Cihaz girdilerinin sağlanması amacıyla usb destekli klavye ve fare bağlantısı Şekil 3.20' de yer almaktadır.

Şekil 3.20.Usb Klavye Ve Fare Bağlantısı

3.4.2.4.4. USB Güç Bağlantısı

Sistem düşük güçte çalıştırılmak istendiğinde beslemesi usb portu üzerinden Şekil 3.21'de görüldüğü gibi sağlanır.

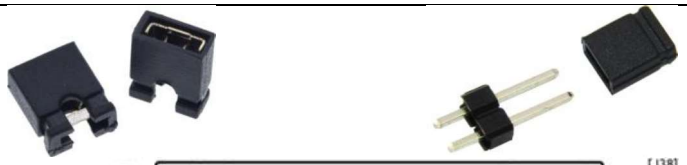
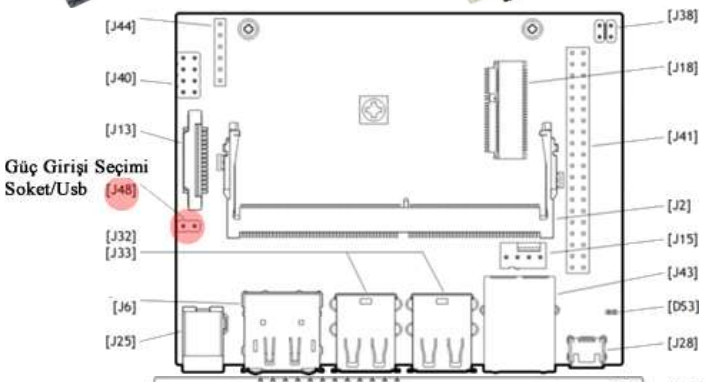
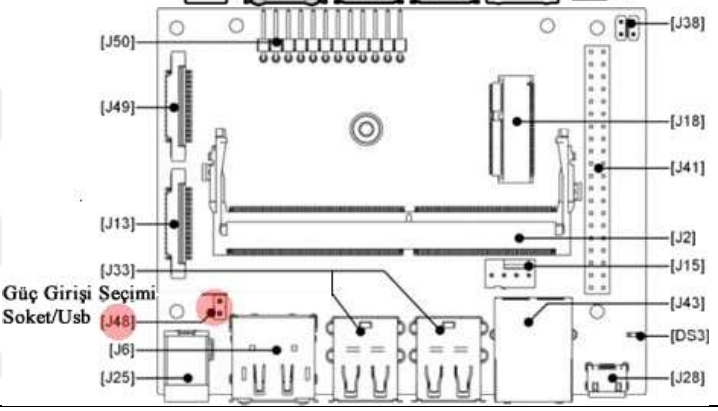


Şekil 3.21. Usb Güç Bağlantısı

3.4.2.4.5. Besleme Tipi Seçimi

Besleme tipi seçimi yapmak için Tablo 3.3'te yeralan Madde 1 de bulunan 2.54 jumper parçası ana kart üzerinde bulunan J48 pinlerine takılır. Bu sayede ana kart tam kapasite de kullanılmak üzere adaptör girişinden gelen beslemeleri kabul eder.

Tablo 3.3. Besleme Tipi

Madde No	Görsel
1	
2	
3	

3.4.2.4.6. Usb Erişim Bağlantısı

Ayarların konsol üzerinden yapılabilmesi için micro usb kablo bilgisayara bağlanmalıdır. Bu bağlantı Şekil 3.22'de gösterilmiş olup, sanal terminal arayüzü açma işlemi gerçekleştirir.

Şekil 3.22.Usb Seri Haberleşme

3.4.2.4.7. Adaptör Güç Bağlantısı

Gün bağlantısı adaptör yardımıyla sağlandığı zaman kullanılan bağlantısı Şekil 3.23'te gösterilmiştir.

Şekil 3.23. Güç Bağlantısı

3.4.2.5. Cihazı Çalıştırma

Bu bölümde günümüz kullanıcılarının yan ekipmanları (usb klavye, fare, hdmi monitör vb.) olmayıp sadece bir adet diz üstü bilgisayarına sahip olduğu göz önüne alınmış ve cihazın uzaktan bağımsız bir yere montajlanma ihtimali düşünülmüş, buna istinaden adaptör ile çalıştırma detaylı olarak anlatılmıştır. Bu bölüm ilgili linkin [67] "Initial Setup Headless Mode" bölümünde açıklanmıştır.

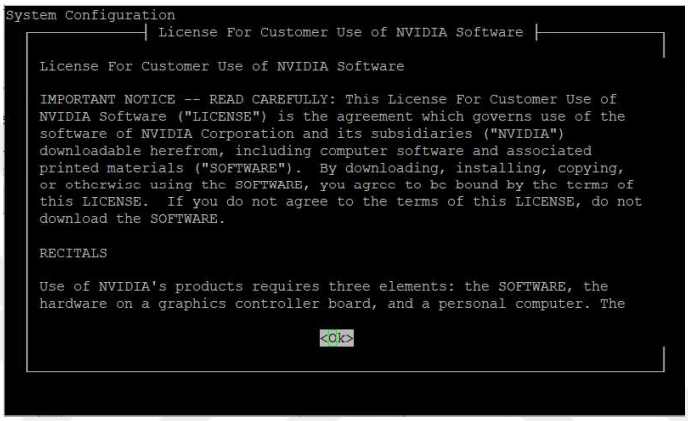
•Güç gediğinde cihaz otomatik olarak açılmıştır.

•1 dakika sonra cihazın sanal seri haberleşme ara yüzü usb bağlantısı üzerinde aktif hale gelmiştir.

•İlgili bilgisayardan işletim sisteminize göre seri bağlantı yapılmış ve haberleşme hızı Baud 115200 olacak şekilde ayarlanmıştır.

•Aşağıda ki işlem adımları Şekil 3.24'ten başlayıp Şekil 3.43'e kadar sırasıyla takip edilmiştir.

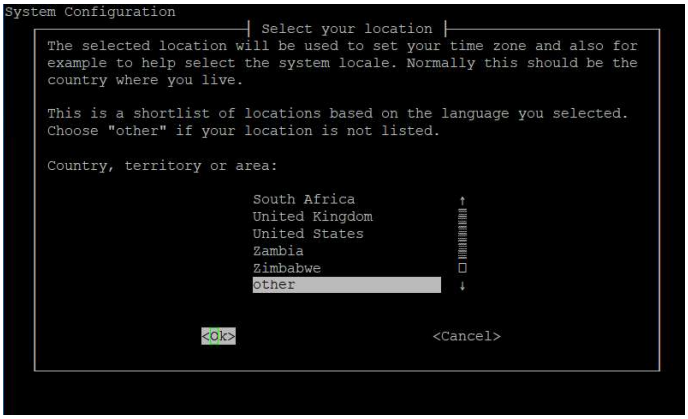
Şekil 3.24.İlk Ekran



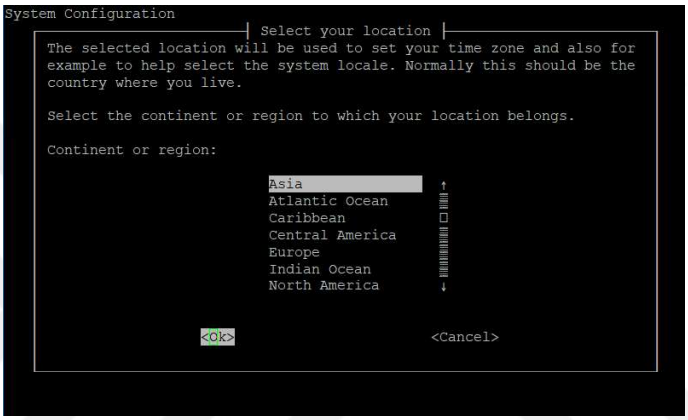
Şekil 3.25. Lisans Sözleşmesi

Dil seçimini yön tuşlarıyla gerçekleştirdikten sonra, bölümler arası geçişi <tab> tuşuna basarak seçimi onaylanmıştır.

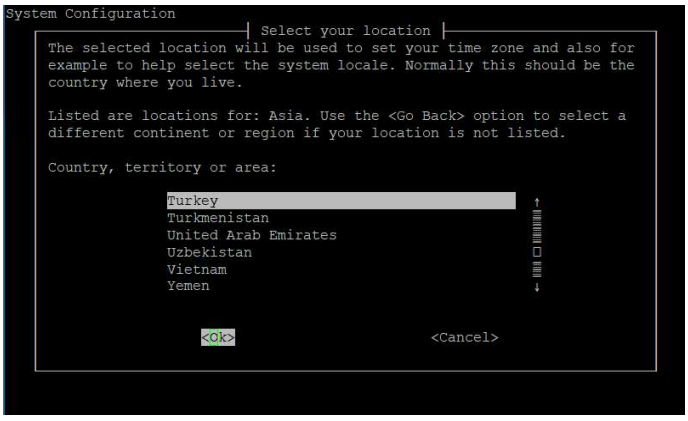
Şekil 3.26. Dil Seçimi



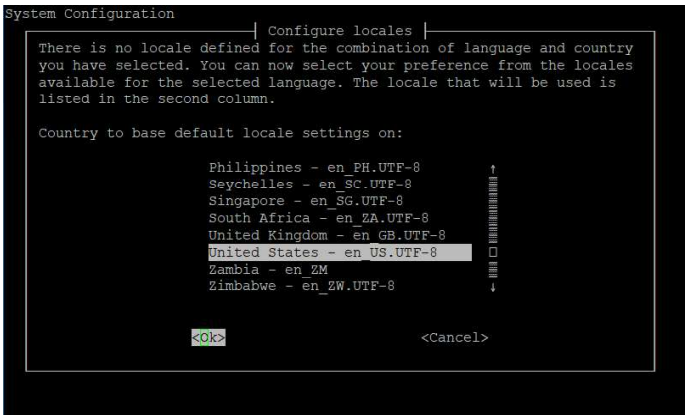
Şekil 3.27. Ülke Seçimi



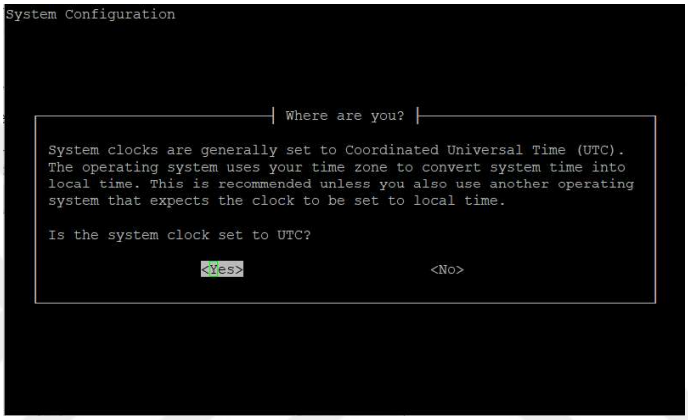
Şekil 3.28. Ülke Seçimi İkinci.Adım



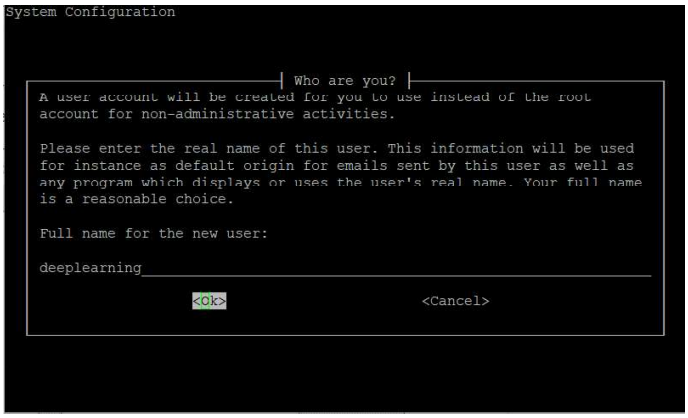
Şekil 3.29. Ülke Seçimi Üçüncü Adım



Şekil 3.30. Yerelleştirme Seçimi



Şekil 3.31. Saat Seçimi



Şekil 3.32. Tam isim veya nick name.

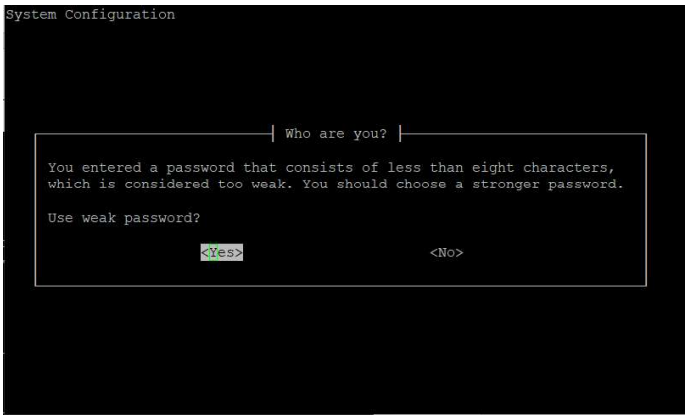
Cihazın kullanıcı adımını ve şifresini oluşturuyorduk ve bu bilgilerle sisteme giriş yaptık.

Şekil 3.33. Kullanıcı Adı

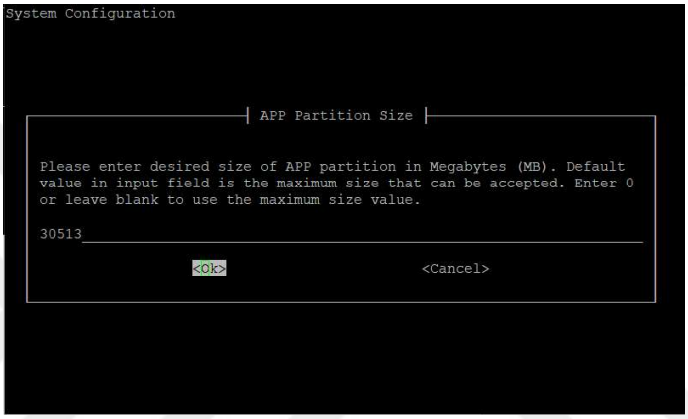
Şekil 3.34. Şifre

Şekil 3.35. Şifre Doğrulama

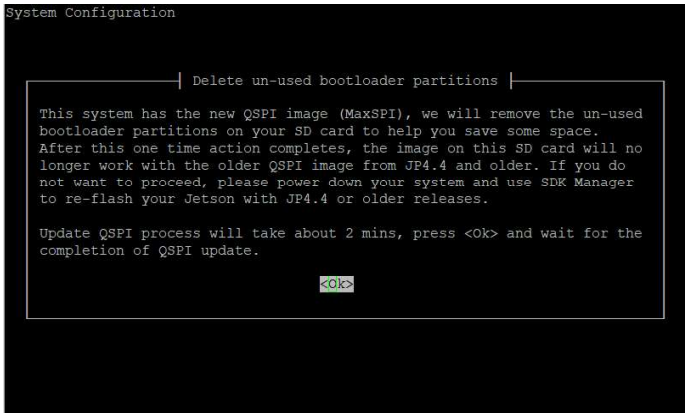
Zayıf şifre uyarısı Şekil 3.36 'da ki gibi gelirse evet deyip geçelim.



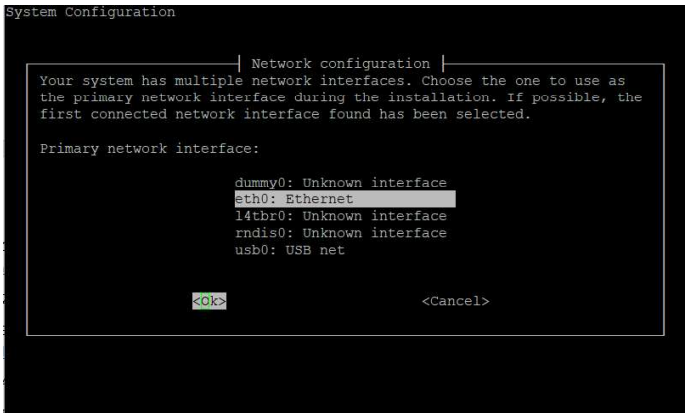
Şekil 3.36. Zayıf Şifre Uyarısı



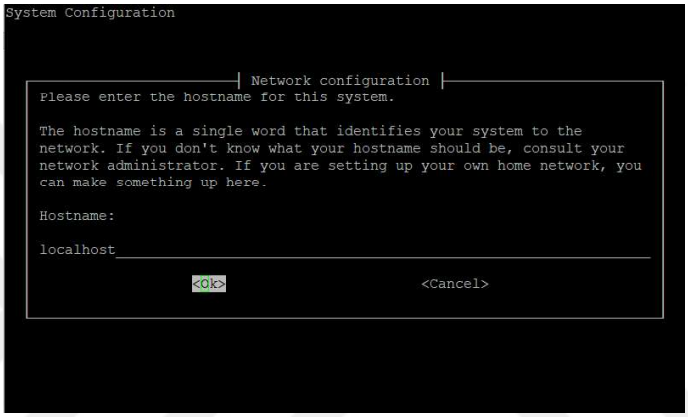
Şekil 3.37. Disk Boyutu



Şekil 3.38. Disk Yapılandırması

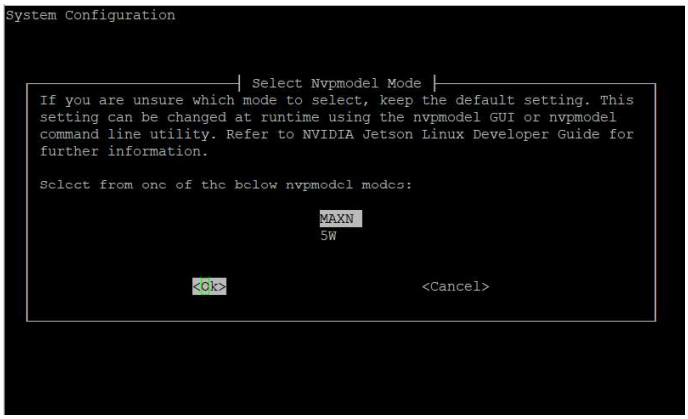


Şekil 3.39. İnternet Arayüzü



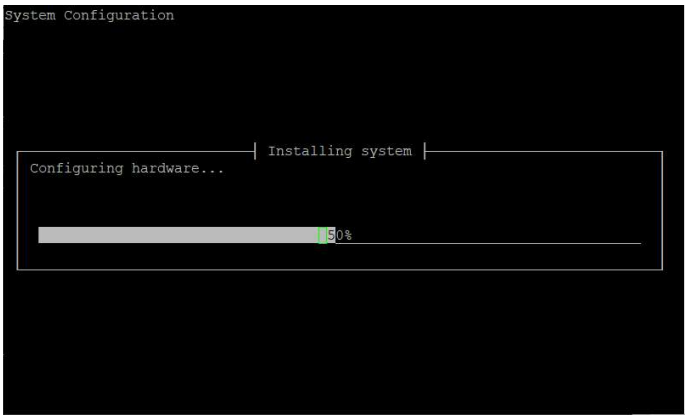
Şekil 3.40. Konak Adı

Cihazı adaptörle kullanım için ayarladık. Bu yüzden seçimi Şekil 3.41'de görüldüğü gibi MAXN olarak seçtik.



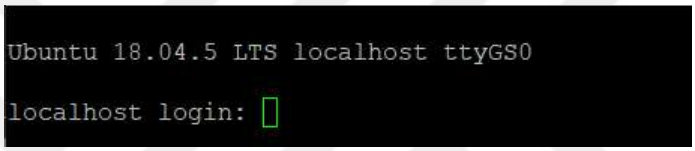
Şekil 3.41. Nvpmode

Bu adımdan sonra Şekil 3.42'te görülen kurulum adımına geçer.



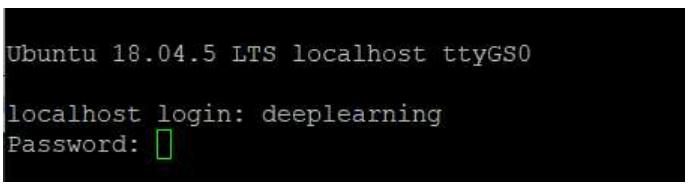
Şekil 3.42. Kurulum

Kurulum işleminden sonra cihaz yeniden başladığında Şekil 3.43'te görüldüğü gibi kullanıcı adını sormuştur.

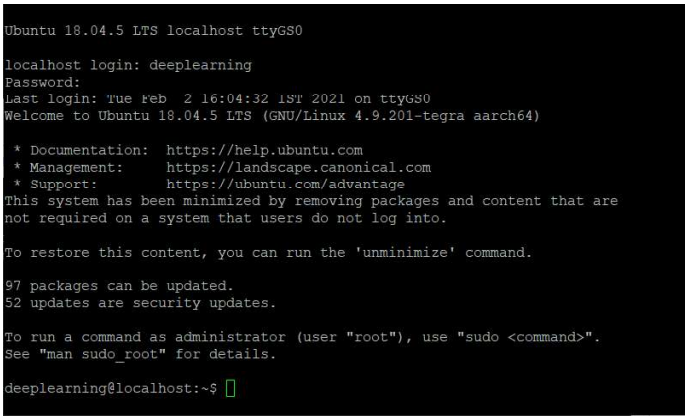


Şekil 3.43. Giriş Ekranı

Önceden belirlenmiş olan kullanıcı adı ve şifre ile Şekil 3.44'de ki gibi giriş yapılmış adından Şekil 3.45'te görülen erişim ekranı gelmiştir.

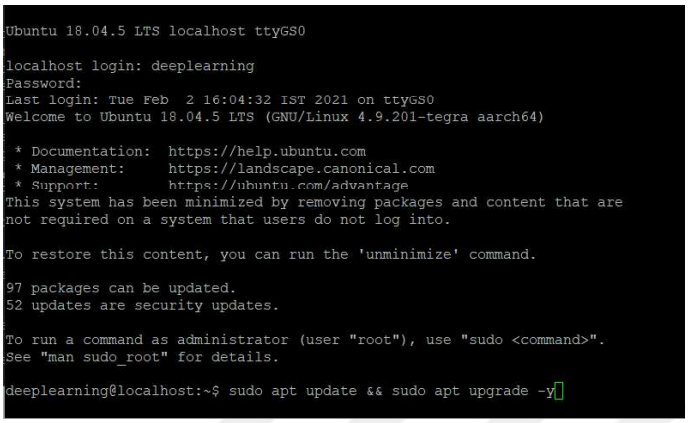


Şekil 3.44. Kullanıcı Girişi



Şekil 3.45. Sistem Erişimi

Güncellemeler “sudo apt update && sudo apt upgrade -y” komutuyla Şekil 3.46’da görüldüğü gibi yapılmış ve sistemimiz kapatılmıştır.



Şekil 3.46. Güncelleme

3.4.2.6. Uzak Erişim Ayarlarını Yapma

Bu ayarlar için uzak masaüstü uygulaması ve ek uygulamalar “sudo apt update && sudo apt install x11vnc ssh nano -y” komutuylu kurulmuş ve uzaktan vnc uzak masaüstü başlatılması için Ubuntu’nun otomatik bir şekilde giriş yapması sağlanmıştır. Bu yüzden ilgili kaynaktaki [68] ayarları kendimize uyarlayarak dosyayı sudo nano /etc/gdm3/custom.conf komutuyla açıp Tablo 3.4’te ki gibi düzenliyoruz.

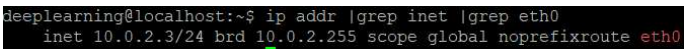
Tablo 3.4. Otomatik Giriş

Eski Ayarlar	Yeni Ayarlar

#Enabling automatic login
#AutomaticLoginEnable = true
#AutomaticLogin = user1
	#Enabling automatic login
	AutomaticLoginEnable = true
	AutomaticLogin = deeplearning

Yeniden başladıktan (sudo reboot) sonra. Cihaza ssh ve vnc üzerinden erişilebilir.

Usb sanal seri bağlantı üzerinden ip addr |grep inet |grep eth0 komutuyla Şekil 3.47’de görüldüğü gibi cihazın ip adresini öğrenelim.

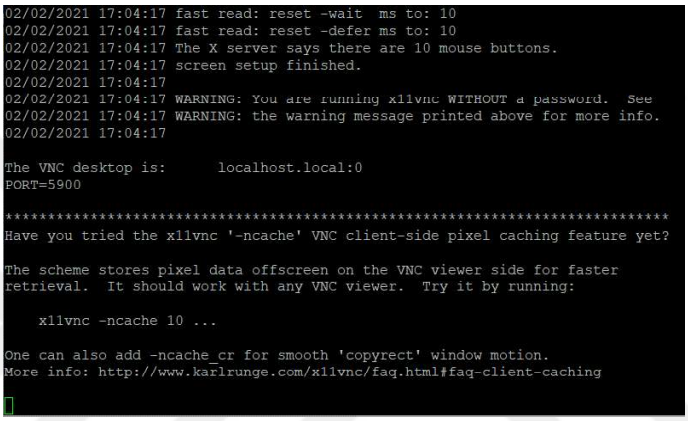


Şekil 3.47.Ip Adresini Öğrenme

Cihazı sürekli aynı ip de bulmak istiyorsak nmtui [69] yardımıyla ipsini sabit olarak ayarlayın.

Şimdi uzak konsol bağlantısı için işletim sistemimize göre bir ssh ara yüzü seçelir. Bu arayüzden ssh deeplearning@10.0.2.3 komutuyla sisteme erişim sağlanır. Linux , Mac ve Windows 10 (powershell) makineler ssh bağlantıyı terminalden desteklemektedir, diğer Windows sürümeleri için putty vb. uygulamalar kullanabilirsiniz.

Vnc uygulaması için Jetson Nano üzerinde x11vnc yazarak uygulamayı çalıştırdıktan sonra işletim sisteminize göre bir adet vnc görüntüleme uygulamasına ihtiyacınız vardır. Sürekli vnc sunucusu açık kalmalı ise lütfen şifre belirleyin. X11vnc hata verirse x11vnc -noshm -display :0 şeklinde çalıştırılır. X11VNC Açılışı Şekil 3.48'de yer almaktadır.



Şekil 3.48. X11VNC Açılışı

İp adresi ve port numarası ile giriş yaptığımızda karşımıza gelen ekran Şekil 3.49 gibi olacaktır.

Şekil 3.49. Uzak Masa Üstü

3.4.3. Jetson Nano Üzerinde Gerçek Zamanlı Lınux Çekirdeği Uygulanması

Bu bölümde Jetson nano için gerçek zamanlı çekirdek (kernel) derleyeceğiz. Linux terminal kullanımı ve temel Linux bilgisi gerekmektedir. Daha detaylı teknik bilgi için ilgili belge kullanılabilir [70].

İşlem yapılacak sistemin çekirdek sürüm numarası öğrenilmeli ve bu sürüm numarasına uygun yama paketleri sistem üzerine indirilmiştir [71]. İndirme adımları Tablo 3.5'te yer almaktadır.

Tablo 3.5. Kernel İndirme

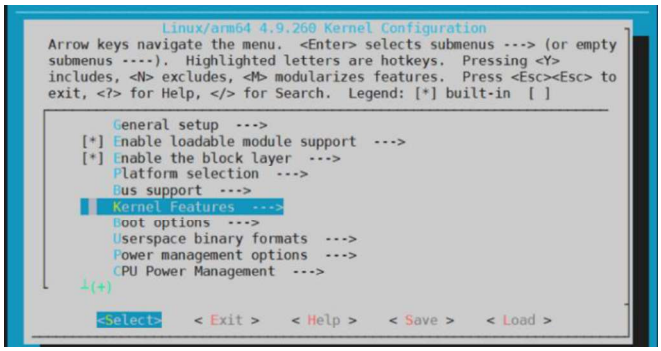
Adım	Komut
1	wget https://developer.nvidia.com/embedded/l4t/r32_release_v5.1/r32_release_v5.1/sources/t210/public_sources.tbz2
2	tar xvf public_sources.tbz2 Linux_for_Tegra/source/public/kernel_src.tbz2
3	cp Linux_for_Tegra/source/public/kernel_src.tbz2 kernel_src.tbz2
4	rm -rf Linux_for_Tegra
5	tar -xvf kernel_src.tbz2
6	cd kernel/kernel-4.9

Artık derleme aşaması için gerekli olan bileşenlere sahip olduğumuzdan derleme aşamasına geçebiliriz. İlk adım olarak çekirdeğimizin ayarlarını Jetson üzerinden temin edip ardından gerçek zaman yapasını yapalım ve sistem çekirdeği ayarını Tablo 3.6'yı kullanarak tam destekli kesme tanımı olan Fully Preemptible Kernel (RT) olarak gerçekleştirilelim.

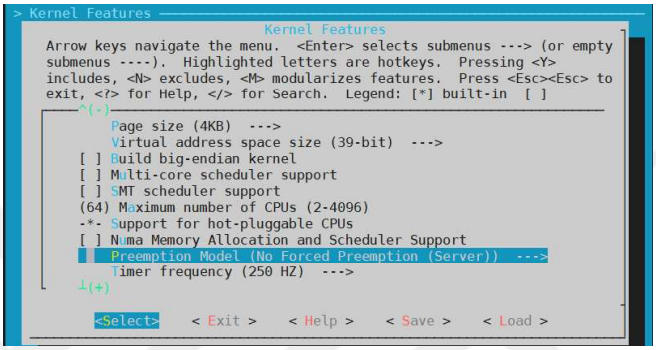
Tablo 3.6. Kernel Ayar

Adım	Komut
1	zcat /proc/config.gz > .config
2	./scripts/rt-patch.sh apply-patches
3	make menuconfig

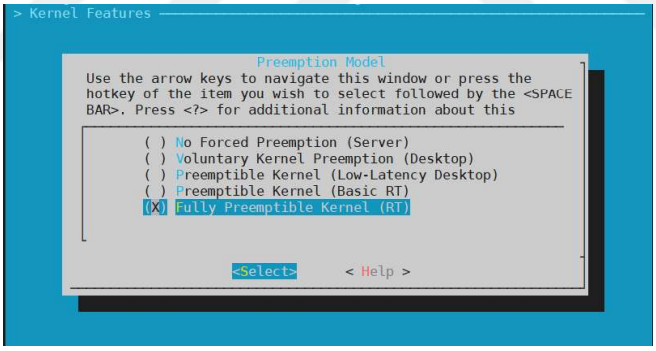
Bu aşamada Şekil 3.50 ile 3.52'den gelen menülerden Kernel Features ->Preemption Model-> Fully Preemptible Kernel (RT) seçimi yapılarak çekirdek gerçek zamanlı olarak ayarlanır. İstenirse xconfig ile Qt ile oluşturulmuş bir ara yüzden de işlem yapılabilir.



Şekil 3.50. Çekirdek Ayarları Bölümü



Şekil 3.51 Çekirdek Özellikleri Bölümü



Şekil 3.52. Gerçek Zaman Destek Çeşidi

Diğer seçim işlemleri Tablo 3.7'de yazanlara göre bulunup gerçekleştirilmiştir.

Tablo 3.7. Menü Seçimleri

Adım	Konum	Değer
1	Kernel Features -> Timer frequency.	1000Hz
2	General setup -> Timers subsystem	Full dynticks system (tickless)

İlgili işlem adımları tamamlanmış ve sayfadan çıkılırken dosya kaydedilmiştir. Bu işlem yaklaşık 60 dakika sürmüştür. Başarılı adımlardan sonra Tablo 3.8'de yer alan adımlara göre boot değişimi yapılmıştır.

Tablo 3.8. Kernel Derleme

Adım	Komut
1	cd ../..
2	./nvbuild.sh
3	sudo make modules_install
4	sudo cp arch/arm64/boot/Image /boot/Image

Bu aşamada işlemler başarılı tamamlanmış ve sistemi gerçek zamanlı çekirdeği seçerek başlattığımızda gerçek zamanlı çekirdeğimiz çalışmaya başlamıştır.

Bu işlemi doğrulamak için Linux komut penceresinden uname -a yazarak çekirdek isminde #1 SMP PREEMPT RT ifadesi olup olmadığı kontrol edilmiş. İşlemler doğru olduğundan #1 SMP PREEMPT RT ifadesi yer almıştır.

3.4.3.1. Test Yazılımı İşlemleri

3.4.3.1.1. Test Yazılının Çalıştırılması

Bu işlemler için cyclictest [72] isimli uygulama kullanılacaktır. Bu uygulamanın kullanılabilmesi için Tablo 3.9'da işlenen adımlarla kaynak koddan derlenmesi gerekmektedir.

Tablo 3.9. Kaynaktan Derleme

Adım	Komut
1	sudo apt install libnuma-dev -y
2	wget https://mirrors.edge.kernel.org/pub/linux/utils/rt-tests/rt-tests-1.10.tar.gz
3	tar xvzf rt-tests-1.10.tar.gz
4	cd rt-tests-1.10/
5	make && sudo make install

3.4.3.1.2. Yamadan Önce Ve Sonrasının Karşılaştırılması

Karşılaştırma için uygulamanın yönetici izinleriyle çalıştırılması gerekmektedir. Bu bilgileri grafik çıktı almak için OSADL nin sunduğu grafik işlemlerini Tablo 3.10'da bulunan adımları uygulayarak kullandım [73].

Tablo 3.10. Grafik Programı

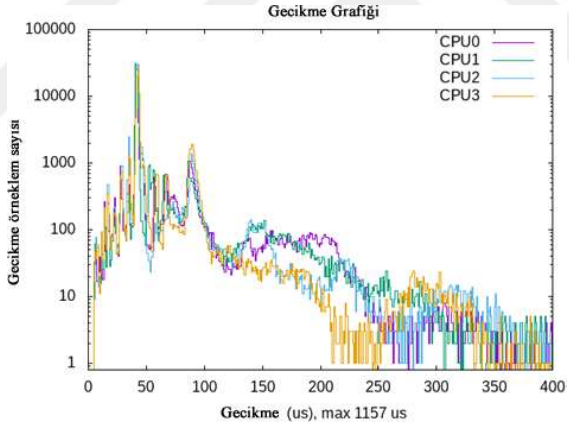
Adım	Komut
1	sudo apt install gnuplot -y
2	mkdir logs && cd logs
3	wget https://www.osadl.org/uploads/media/mklatencyplot.bash
4	chmod u+x mklatencyplot.bash

Bu test yaklaşık 6 saat sürmektedir. Daha hızlı sonuç almak için mklatencyplot.bash dosyasının ilk satırında bulunan döngü sayısını 100000 ile değiştirelim ve cyclictest ifadesi başına sudo ifadesini ekleyelim. İlgili satırın değeri sudo cyclictest -l100000 -m -Sp90 -i200 -h400 -q >output olacaktır. İlk satır bilgilerinin açıklaması Tablo 3.11'de verilmiştir.

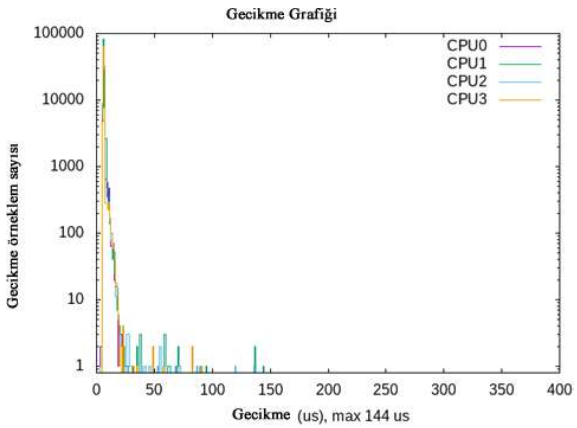
Tablo 3.11. Cyclictest Parametreleri

Simge	Açıklaması
-l	Döngü sayısı
-m	Bellek erişimini kilitle
-Sp90	Tüm mevcut çekirdekleri kullan ve önceliği 90 olarak belirle
-i200	İş parçaları arasında geçen süre
-h400	400 mikro saniye aralıklı histogram çıktısı üret
-q	İşlem bitince ekranda çıktıyı göster
>	Linux konsol bilgilerini dosyaya aktarma işlemi

İlgili komut dosyası olan mklatencyplot.bash içinden çeşitli denemeler yapılabilir. Yapılan denemelerde alınan çıktılar aşağıda verilmiştir. İmajın üzerinde hazır gelen çekirdek değerleri Şekil 3.53 üzerinde, yeni hazırladığımız çekirdeğin çalışma değerleri ise Şekil 3.54 de verilmiştir.



Şekil 3.53. SMP Kernel



Şekil 3.54. RT Kernel

3.4.4. C/C++

Ken Thompson ve Dennis Ritchie 'nin AT&T Bell laboratuvarlarında birlikte yaptıkları çalışmalarda işletim sistemi olan UNIX 'i geliştirmek için B dilinin özelliklerinden üretilmiş yapısal bir programlama dilidir. C dili 1972 yılında geliştirilmesine rağmen, Brian Kernighan ve Dennis M. Ritchie tarafından yayımlanan "C Programlama Dili" kitabından sonra yayılıp yaygınlaşması hızlanmıştır. Günümüzde kullanılan Microsoft Windows, GNU/Linux, *BSD, Minix gibi neredeyse tüm işletim sistemlerinin yapımında %95' lere varan oranlarda kullanılmış ve günümüzde hâlen sistem, alt sistem, sürücü yazılımı, işletim sistemi modülleri, gömülü sistem ve hız gereken her yerde oldukça yaygın kullanılan ve sınırları belirsiz ama kuralları oldukça keskin bir dildir. Bu kural keskinliği, program yazarına sonsuz özgürlüğün yanısıra çok büyük hatalar yapabilme olanağı sağlar. Programlamanın ve donanımlarının gelişim süreciyle beraber programlamanın karmaşıklaşması ve gereksinimlerin sürekli artması ile uygulama programlarında nesne yönelimlimi yaklaşımının öne sürülmesinden sonra C programcıları büyük ölçüde nesne yönelimlini destekleyen C++ diline geçmişlerdir [74]. Fakat günümüzde C dili gömülü sistem ve mikroişlemci programlamasında oldukça yaygın olarak kullanılmaktadır. C dilinin en büyük gücü işaretçiler (pointer) yapısıdır. Bu yapı sayesinde sistemde var olan herhangi bir donanıma kaydedici adresi bilinmek şartıyla Tablo 3.12'te verilen örnekteki gibi erişilebilir. Ram, Hdd, Cpu ve ya Gpu nun istenen bölgesine erişim sağlayabilir. Aşağıda ram değişken örneği yer almaktadır.

Tablo 3.12. İşaretçi Örneği

Kod	**Çıktı**

int i=0x04030201;
int *p=0x61FE0C;
char *c=&i;
 	
printf("i nin ram adresi\t%p\nSabit girilen adres\t%p\n",&i,p);
 	
printf("\ni nin degeri\t0x%x\nisaretci degeri\t0x%x\n",i,*p);
 	
*(c+2)=0xFF;
 	
printf("\n3. deger degisimi 0x%x\n",i);
	i nin ram adresi 61FE0C
	Sabit girilen adres 61FE0C
	
	i nin değeri 0x4030201
	isaretci degeri 0x4030201
	
	3. deger degisimi 0x4ff0201

3.4.5. OpenCv

OpenCV (Open Source Computer Vision Library) [75], bilgisayarlı görü ve makine öğrenimi araçlarını ve ortamını sağlayan açık kaynaklı bir yazılım kütüphanesidir. OpenCV, bilgisayarlı görü uygulamaları için ortak bir altyapı desteği sunmak ve ticari ürünlerde dahil makine görüsünün kullanımını hızlandırmak amacıyla oluşturulmuştur. BSD lisanslı bir ürün olan OpenCV, ticari olarak kodun kullanımını ve değiştirmesini kolaylaştırır.

3.4.5.1. İşlevselliği

Kitaplık, ilkel ve en yeni bilgisayarlı görü ve makine öğrenmesi algoritmalarından oluşan çok kapsamlı fonksiyonlar barındıran 2500'den fazla optimize edilmiş algoritmaya sahiptir. İçinde bulunan yetenekli fonksiyonlar, yüz algılaması ve tanıması, nesne tanıması, videolardaki canlı hareketlerini sınıflandırmak, kamera değişimlerini izlemek, hareket halinde olan nesneleri izlemek, nesne ve ortamın üç boyutlu modellemesini yapmak, çoklu kamera görüntülerinden üç boyutlu nokta kümeleri türetmek, yüksek çözünürlük elde etmek için görüntüleri bir araya getirmek, tüm bir sahnenin görüntüsünü kullanarak, var olan görüntü veri tabanından benzer görüntüleri bulmak, flaş kullanılarak çekilen görüntülerden kırmızı gözleri kaldırmak, göz hareketlerini takibini sağlamak, manzarayı tanımak ve artırılmış gerçeklikle kaplamak için işaretler oluşturmak gibi işlemler için kullanılabilir. OpenCV'nin 47 binden fazla kullanıcısı olan bir topluluk ve 18 milyonu aşan tahmini indirme sayısısın ulaşmış bir kitaplıktır. OpenCv, şirketlerde, araştırma gruplarında ve devlet kurumlarınca yoğun olarak kullanılmaktadır [76].

3.4.5.2. Kurulumu

Aşağıda ki Tablo 3.13'te bulunan kurulum adımları yaklaşık 24 saat sürmektedir.

Tablo 3.13. OpenCv Kurulum Adımları

Adım	Komut
1	cd Desktop
2	mkdir opencv
3	cd opencv
4	git clone https://github.com/opencv/opencv.git
5	git clone https://github.com/opencv/opencv_contrib.git
6	cd opencv_contrib
7	git checkout 4.4.0
8	cd ../opencv
9	git checkout 4.4.0
10	mkdir build

Adım	Komut
11	cd build
12	cmake -D CMAKE_BUILD_TYPE=RELEASE \ -D CMAKE_INSTALL_PREFIX=/usr/local \ -D BUILD_opencv_python3=ON \ -D INSTALL_C_EXAMPLES=OFF \ -D INSTALL_PYTHON_EXAMPLES=OFF \ -D WITH_TBB=ON \ -D WITH_V4L=ON \ -D WITH_QT=OFF \ -D WITH_CUDA=ON \ -D WITH_CUDNN=ON \ -D OPENCV_DNN_CUDA=ON \ -D WITH_OPENGL=ON \ -D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib/modules \ -D BUILD_EXAMPLES=OFF \ -D OPENCV_GENERATE_PKGCONFIG=ON
13	make -j4
14	sudo make install
15	sudo make install
16	sudo sh -c 'echo "/usr/local/lib" >> /etc/ld.so.conf.d/opencv.conf'
17	sudo ldconfig

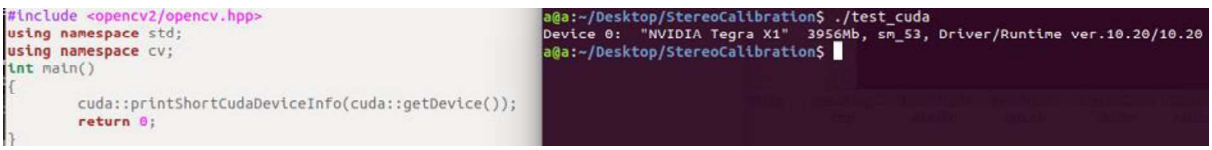
3.4.6. CUDA

"GPU Hızlandırmalı Uygulamalar Geliştirin, Optimize Edin ve Dağıtın"

NVIDIA® CUDA® Toolkit, yüksek performanslı GPU hızlandırmalı uygulamalar oluşturmak için bir geliştirme ortamı sağlar. CUDA Toolkit ile GPU hızlandırmalı gömülü sistemler, masaüstü iş istasyonları, kurumsal veri merkezleri, bulut tabanlı platformlar ve HPC süper bilgisayarlarında uygulamalarınızı geliştirebilir, optimize edebilir ve dağıtabilirsiniz. Araç seti, x86, Arm ve POWER dahil olmak üzere ana mimarilerde uygulamanızı oluşturmak ve dağıtmak için GPU ile hızlandırılmış kitaplıklar, hata ayıklama ve optimizasyon araçları, bir C/C++ derleyicisi ve bir çalışma zamanı kitaplığı içerir.

Bilim adamları ve araştırmacılar, hesaplamaları çoklu GPU yapılandırmaları arasında dağıtmak için yerleşik yetenekleri kullanarak, tek GPU iş istasyonlarından binlerce GPU'lu bulut kurulumlarına kadar ölçeklenen uygulamalar geliştirebilir [77].

Jetson Nano Sd kart imajında dahili olarak gelmektedir. Cuda Test Yazılımı Şekil 3.55'de gösterilmektedir.



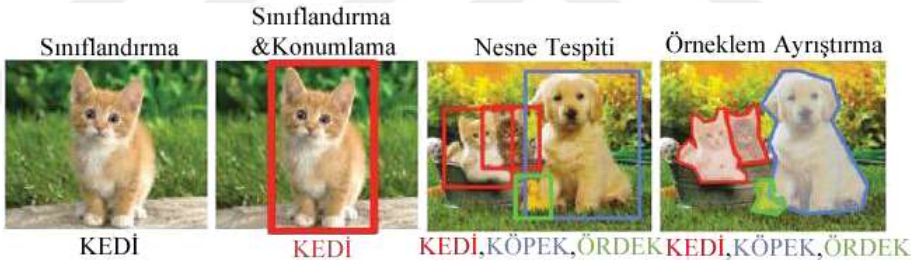
Şekil 3.55. Cuda Bilgisi

3.4.7. Darknet Yolo Kütüphanesi ve Çalışma Şekli

Jetson Nano sisteminin nesne tanıma modülü için kullanılan konvolüsyonel yapay sinir ağları temelinde geliştirilmiş olan GPU destekli gerçek zamanlı nesne tanıma sistemi olan YOLO(You Only Look Once) [78] hakkında bilgiler verilmiştir.

3.4.7.1. Bilgisayar Görüşü ve Konvolüsyonel Yapay Sinir Ağları

İnsanlar etraflarındaki 3 boyutlu dünyayı görmek ve anlamak için gözlerini ve beyinlerini kullanırlar. Örneğin, Şekil 3.56.'da gösterildiği gibi bir görüntü verildiğinde, insanlar görüntüdeki bir "kedi"yi kolaylıkla görebilirler. Böylece görüntüyü sınıflandırır (sınıflandırma görevi); görüntüdeki kediyi konumlandırır (sınıflandırma artı lokalizasyon görevi); resimde mevcut olan tüm nesneleri konumlandırır ve etiketler (nesne algılama görevi); ve mevcut olan bireysel nesneleri bölümlere ayırır (örnek segmentasyon görevi). Bilgisayarlı görme bilimi, bilgisayarlara daha iyi olmasa da benzer bir yetenek vermeyi amaçlamaktadır. Daha doğrusu bilgisayarlı görme, en şaşırtıcı yeteneklerden birini çoğaltabilen yöntemleri geliştirmeyi amaçlamaktadır [79].



Şekil 3.56. Nesne Tanıma için Örnek Görüntü [79]

Bilgisayarların görüntü verileriyle;

• Görüntüye bakmak ve sınıflandırma yapması

• Sınıflandırma artı lokalizasyonu gerçekleştirmesi (örneğin, etrafında bir sınırlayıcı kutu bulmak için)

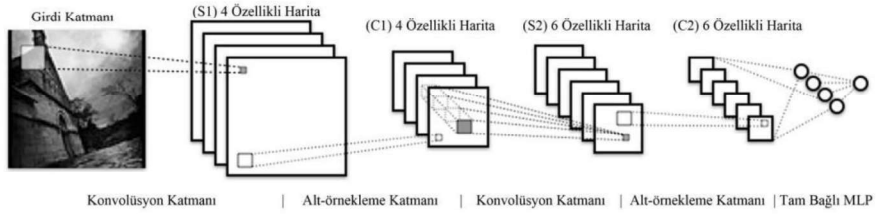
• Görüntüde mevcut olan tüm nesneleri konumlandırması

• Nesneleri etiketlemek veya semantik örnek segmentasyonunu gerçekleştirmesi, yani aynı türden olsalar bile, bir sahnede tek tek nesnelerin segmentasyonunu yapması mümkündür.

Ancak, dünyanın üç boyutlu yapısını iki boyutludan geri kazanmak ve anlamak kameralar tarafından çekilen görüntüler için zorlu bir iştir. Bilgisayar ortamında araştırmacılar, görüntülerden üç boyutlu Şekil ve görünüşü geri kazanmak için matematiksel teknikler geliştirmektedir. Örneğin, yeterince geniş bir resim grubu ve çeşitli görüşlerden yakalanan nesne

bilgisayar görüş algoritmaları ile yeniden yapılandırabilmektedir. Birden fazla görünüş kullanarak nesnenin doğru olan 3 boyutlu yüzey modeli görünümü ortaya çıkarılabilmektedir. Ancak, tüm bu ilerlemelere rağmen, görüntüleri insanlarla aynı seviyede anlaması ve anlamlandırması mümkün değildir.

Konvolüsyonel yapay sinir ağları (Convolutional Neural Network-CNN) ileri yönlü bir yapay sinir ağı türüdür. Algoritmasının temeli, hayvanların görme merkezinden esinlenerek ortaya çıkmıştır. CNN ağı, ilk kez 1988 yılında Yann LeCun tarafından, 1998 yılına kadar geliştirme ve optimizasyon çalışmaları devam eden LeNet (Şekil 3.57.) isimli mimari olarak ortaya çıkmıştır [80].



Şekil 3.57. LeNet Mimarisi [80]

CNN algoritmaları ses ve görüntü işleme alanlarında oldukça yaygın halde kullanılmaya başlanmış ve özellikle görüntü işleme üzerinde en iyi sonuçlar(state-of-art) elde edilmiştir. Günümüzde, CNN tabanlı algoritmalar arasında en popüler olanlar R-CNN(Region Based CNN), Fast R-CNN, Faster R-CNN ve YOLO(You Only Look Once)'dur.

3.4.7.2. Yolo

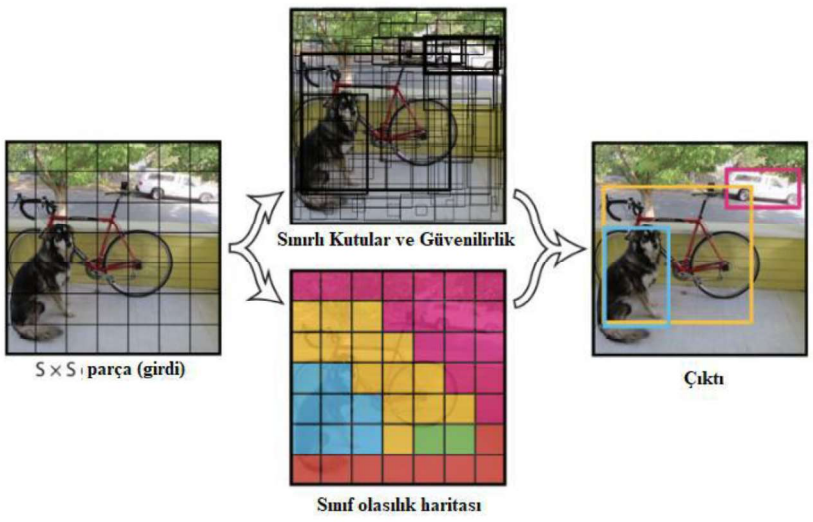
"Nesne algılamaya yeni bir yaklaşım olan YOLO'yu sunuyoruz". Sloganıyla piyasaya açık yanak olarak sürülen YOLO, nesne algılamayla ilgili önceki çalışmalar, algılama gerçekleştirmek için sınıflandırıcıları yeniden amaçlar. Bunun yerine, nesne algılamayı, uzamsal olarak ayrılmış sınırlayıcı kutulara ve ilişkili sınıf olasılıklarına bir regresyon problemi olarak çerçeveleriz. Tek bir sinir ağı, tek bir değerlendirmede doğrudan tam görüntülerden nesne sınırı karesini ve ait olduğu sınıfın olasılıklarını ortaya koyar. Algılama hattının tamamı tek bir ağ olduğundan, uçtan uca doğrudan algılama performansına göre optimize edilebilir [78].

Birleşik mimarisi son derece hızlıdır. Temel YOLO modeli, görüntüyü anlık olarak saniyede 45 kare hızında işler. Fast YOLO ağın daha küçük bir versiyonudur. Bu sayede diğer gerçek zamanlı tespit edici sistemlerin iki kat fazla haritasını elde ederken, saniyede 155 kareye varan işlem hızı gösterir. YOLO kıyaslama yapılan günümüz teknolojisindeki ürünlerden daha fazla yerelleştirme hatası yapmasına karşın arka planda yapılan tahminde yanlış pozitiflerin

ortaya çıkma olasılığı daha düşüktür. Son olarak, YOLO nesnelerin çok genel temsillerini öğrenir. Doğal görüntülerden sanat eseri gibi diğer alanlara genelleme yaparken, DPM ve R-CNN dahil olmak üzere diğer algılama yöntemlerinden daha iyi performans gösterir [81,82].

3.4.7.3. Yolo(You Only Look Once)'nun Çalışma Şekli

Yeni ve çok hızlı işleyen bir başka yaklaşım olan YOLO gerçek zamanlı üst düzey(satae-of-art) bir nesne tanıma sistemidir. Kısaltmasından da anlaşılacağı gibi; "Sadece Bir Kere Bakarsın (You Only Look Once)" cümlesinden türetilmiş olup söylenmek istenen insanların tek bir karedeki görüntüde olan objeleri hızlı bir Şekilde bir bakışta bulabilmesidir. YOLO'nun da amacı bu hıza erişmek olmuştur. YOLO tasarımcılarına göre görüntü karesini soldan sağa taramak (ve o sırada ufak bir kutu içine düşen piksellere bakmak) ya da obje yeri hakkında fikir üretmek gerekli görülmemiştir. Tüm kare ve o karedeki tüm obje tipleri, yerleri hakkında global bir karar verilebilir. YOLO'da tek bir konvolüsyonel sinir ağı, sınırlı kutuları ve bu kutuların içindeki nesnelere göre sınıf olasılıklarını tahmin edebilmektedir [83]. Bunu yaparken, Şekil 3.58'te gösterildiği gibi önce görüntüyü SxS boyutlarında parçalara böler ve her bir parça için m tane sınırlı kutu oluşturur. Her bir sınırlı kutu için ağ, sınıf olasılığını ve aralık değerini çıktı olarak verir. YOLO'nun temelinde denetimli bir eğitim verisi yatmaktadır. Burada denetimli eğitim ile kast edilen, bir görüntü için görüntü içindeki objeler, o objelerin yerleri ve büyüklük verilerinin olmasıdır. Ünlü Pascal VOC [24] eğitim verisi bu eğitim setlerinden biridir.



Şekil 3.58. YOLO'nun Çalışma Şekli [83]

3.4.7.4. Yolo Kurulumu

Linux altında YOLO kurulum adımları Tablo 3.14'te yer almaktadır.

Tablo 3.14. Yolo Kurulumu

Adım	Komut
1	git clone https://github.com/AlexeyAB/darknet
2	cd darknet
3	nano Makefile

4	# Make file değerlerini değiştirelim
	GPU=1
	CUDNN=1
	CUDNN_HALF=1
	OPENCV=1
	AVX=0
	OPENMP=0
	LIBSO=0
	ZED_CAMERA=0
	ZED_CAMERA_v2_8=0

5	Ctrl+X
6	Y
7	make

4. BULGULAR VE TARTIŞMA

Uygulamalar C++ dilinde OpenCv kitaplığı kullanılarak JetsonNano platformu üzerinde geliştirilmiş ve denenmiştir.

Tüm uygulamalarda ilk adım olarak kameraların OpenCv ile alına bilmesi için başlatma değerleri verilmelidir. İşlemleri nvarguscamerasrc uygulaması ile OpenCv nin işleyebileceği formata Tablo 4.1'de bulunan fonksiyon ile çevrilirmiştir.

Tablo 4.1. Kamera Tanımlama Fonksiyonu

Fonksiyon	Yapılan İşlem
string to_gstreamer (int sensor_id,int mode,cam_settings_t cam_settings) {return "nvarguscamerasrc sensor-id="+to_string(sensor_id)+" sensor-mode="+to_string(mode)+ " ! video/x-raw(memory:NVMM), width=(int)" + to_string(cam_settings.capture_width) + ", height=(int)" + to_string(cam_settings.capture_height) + ", format=(string)NV12, framerate=(fraction)" + to_string(cam_settings.framerate) +"/1 ! nvvidconv flip-method=" + to_string(cam_settings.flip_method) + " ! video/x-raw, width=(int)" + to_string(cam_settings.display_width) + ", height=(int)" + to_string(cam_settings.display_height) + ", format=(string)BGRx ! videoconvert ! video/x-raw, format=(string)BGR ! appsink";}	to_gstreamer fonksiyonu kameranın id, açılış modu, ve ayar bilgilerini değişken olarak alıp, OpenCv nin işleyeceği kamera algılama bilgisine çevirir.

4.1. Kenar Bulma Algoritmasıyla Uzaklık Tespiti

Standart fonksiyonlar kullanılarak uzaklık tespiti için önce nesnenin konumu tespit edilmelidir. Kamera kalibrasyondan önce ve sonra ki sonuçlar yer almaktadır.

4.1.1. Kalibrasyon Öncesi

Sol kamera 0, sağ kamera 1 olmak üzere OpenCv üzerinde VideoCapture nesnesine Tablo 4.2'de yer alan değerleri vererek program üzerinde kamera görüntülerini almaya hazır hale getirilmiştir.

Tablo 4.2. OpenCv Kamera Tanımlama

C Kodu	**Anlamı**
cam_settings_t cam_settings= { .capture_width = 640, .capture_height = 360, .display_width = 640, .display_height = 360,	Kamera ayarları Boyut 640x360 pixel Kare Hızı 30Fps Döndürme Yok

```
    .framerate = 30,
    .flip_method = 0
};
string Left_Camera_Stream=to_gstreamer(Left_Cam,3,cam_settings);      Sol kamera tanımı
string Right_Camera_Stream=to_gstreamer(Right_Cam,3,cam_settings);    Sağ kamera tanımı
VideoCapture capLeft(Left_Camera_Stream, CAP_GSTREAMER);              Sol görüntü alma
VideoCapture capRight(Right_Camera_Stream, CAP_GSTREAMER);            Sağ görüntü alma
```

Kameranın Tablo 3.2'de verilen değerlerine göre başlangıç değerleri kodda Tablo 4.3 açıklandığı şekilde yer almıştır.

Tablo 4.3. Başlangıç Değerleri

C Kodu	Anlamı
float baseline = 65.0;	İki kamera arası bilinen uzaklık
float focalLenght = 3.04;	Kameranın odak uzaklığı
float Hfov = 62.2;	Düzlemindeki ölçüm alanının yatay boyutu
float sensorWidth = 3.68;	Sensör genişliği

Bahsi geçen (5) ve (6) da verilen formüllerden yola çıkarak Şekil 4.1'de yer alan C++ fonksiyonunu yazılmıştır.

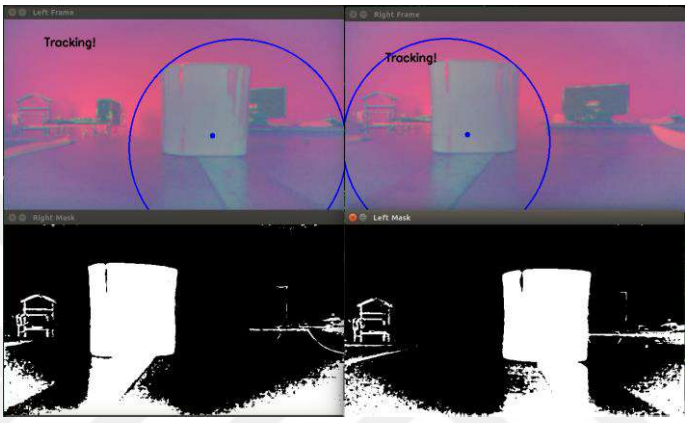
```
float find_depth(Point Left, Point Right, Mat& leftFrame, Mat& rightFrame)
{
    float focal_pixels = 0;

    if (rightFrame.cols == leftFrame.cols)
    {
        // Optik uzaklık f [mm]  [pixel] cevrimi
        //focal_pixels = (rightFrame.cols * 0.5) / tan(Hfov * 0.5 * CV_PI / 180.0);
        focal_pixels =(focalLenght / sensorWidth)*rightFrame.cols;
    }
    else
    {
        cout << "Hata Pixeller esit değil" << endl;
    }

    // Fark Hesabı
    int disparity = Left.x -  Right.x;

    // Derinlik Hesabı Z
    float zDepth = (baseline * focal_pixels) / float(disparity);

    return abs(zDepth);
}
```

Şekil 4.1. Uzaklık Bulma Fonksiyonu

Basit takip etme algoritması çevrede bulunan ışık yansımalarından etkilendiğinden dolayı nesne çizim alanı istenen doğrulukta elde edilememiştir. Elde edilen çember boyutuna göre hesaplanan nesne uzaklığı Şekil 4.2'de görüldüğü gibi 20 cm de konumlandırılmasına rağmen yapılan testlerde bu değer elde edilememiştir. Hata payı kameralara yakınlaştığında artmış uzaklaştığında azalmıştır. Kamera çerçevesi ve nesne boyutundan dolayı 20 cm den yakın

görüntüler sonuç olarak kabul edilmemiştir. Uygulamanın tam görüntüsü Şekil 4.3'te hesaplanan değer çıktısı Şekil 4.4 de yer almıştır.

Şekil 4.2. Nesne Konumu

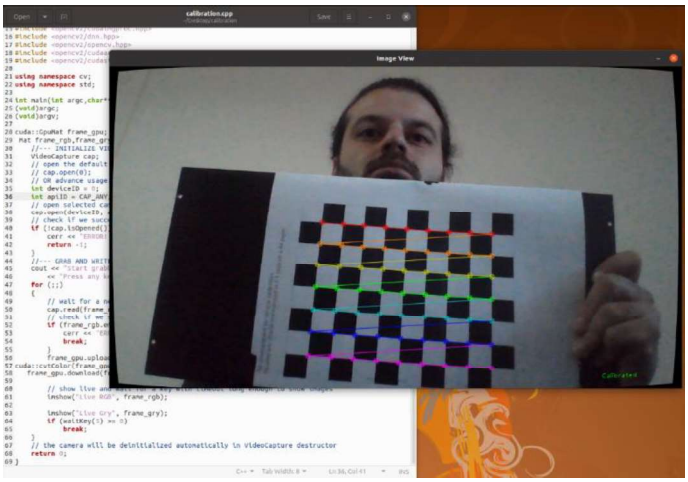


Şekil 4.3. Uygulama Ekran Görüntüsü

Şekil 4.4. Hesaplanan Çıktı

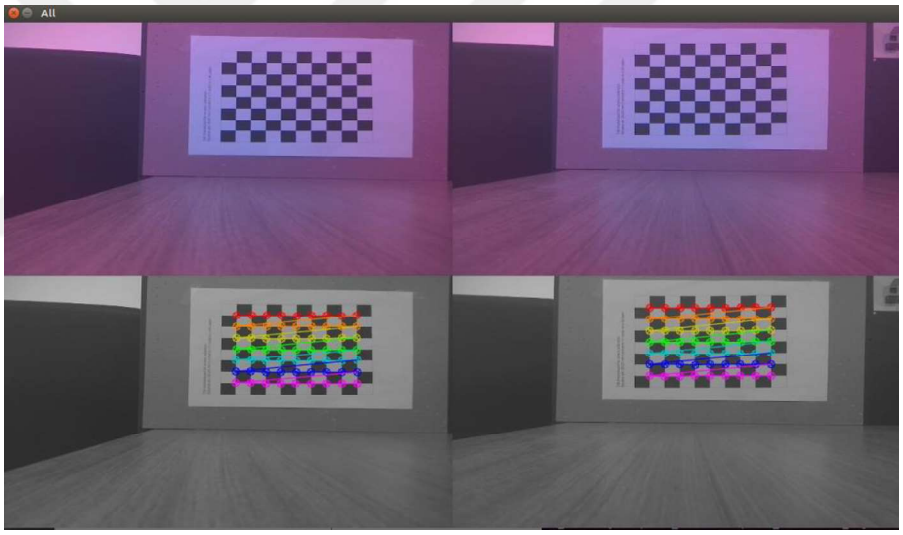
4.1.2. Kamera Kalibrasyonu

İlgili bölüm olan 3.2.2. Kalibrasyon başlığında bahsedilen yöntemin koda uyarlanmış halidir. Kalibrasyon için yazıcı için ölçeklenmiş satranç tahtasının çıktısı ve özellikleri doğru olarak ayarlanmış ve Şekil 4.5'te yer alan çıktı elde edilmiştir. Yaklaşık 800 satır olan uygulama kodları ilgili kaynaktadır [85].



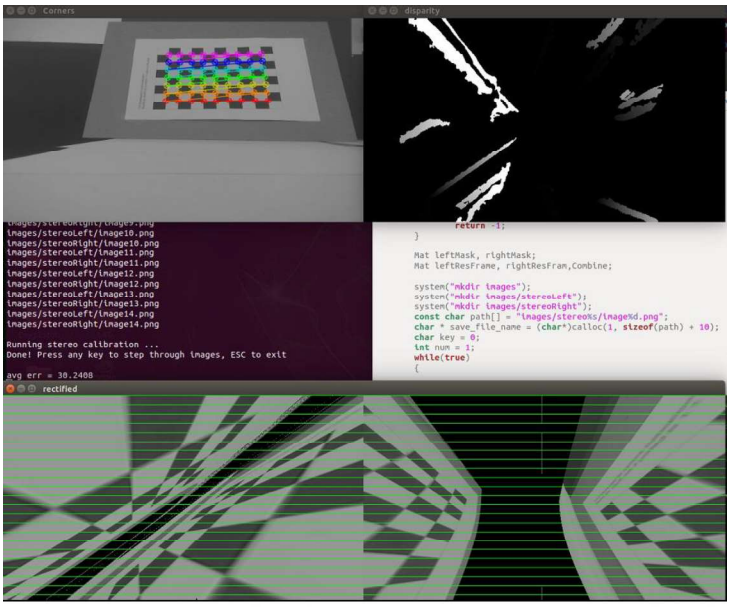
Şekil 4.5. Kalibre İşlemi

Çift kamera kalibrasyonu için iki kameranın aynı anda fotoğrafı çekilir ve cv::findChessboardCorners fonksiyonuna gönderilir. Şekil 4.6'te bulunan kenarlar görülmektedir.



Şekil 4.6. Çift Bord

Yapılan çalışmalar hata oranı çok yüksek çıktığından. Kalibrasyon adımı bu kameralar için atlanmıştır. Şekil 4.7'da bu işlem görülmektedir.



Şekil 4.7. Kalibre Hatası

4.2. YOLO Algoritmasıyla Uzaklık Tespiti

İlk önce kamera kalibrasyon işlemi gerçekleştirilmiştir. Kalibrasyon işlemi istenen sonuçları vermediğinden direk nesne tanıma aşamasına geçilmiştir. YOLO Gpu üzerinden çalıştırılarak nesnelerin konumları bulunmuş ve uzaklıkları hesaplanmıştır.

4.2.1. OpenCv de YOLO Kullanımı

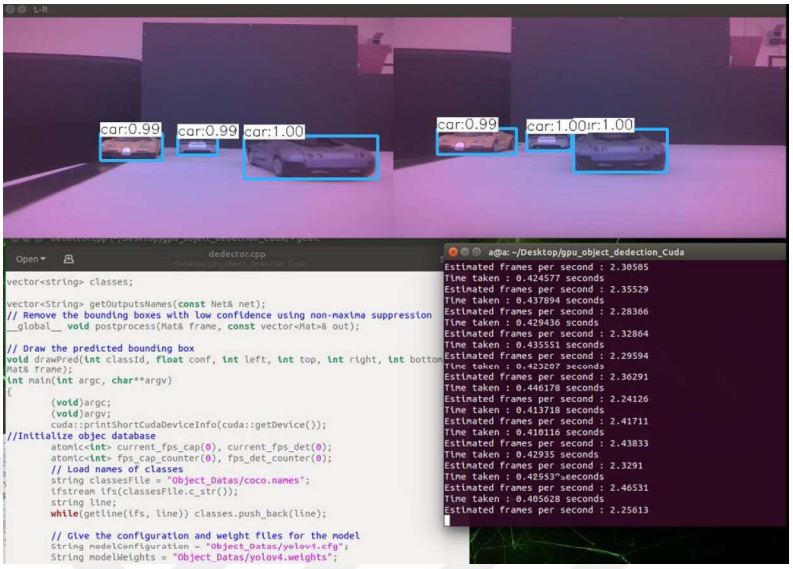
Bu işlemlerde GPU üzerinde işlem yapabilmek için OpenCv ve YOLO derleme adımlarında Cuda aktif edilmelidir. Aşağıdaki Şekil 4.8'te yer alan kod parçasında YOLO modellerinin program içine yüklenmesi gösterilmiştir.

```
// Sınıf isimlerini yükleme
string classesFile = "../../Object_Datas/coco.names";
ifstream ifs(classesFile.c_str());
string line;
while (getline(ifs, line)) classes.push_back(line);

// Ağ ve ayarlarının dosya yolu
String modelConfiguration = "../../Object_Datas/yolov4.cfg";
String modelWeights = "../../Object_Datas/yolov4.weights";

// Ağ ve ayarlarını yükleme
Net net = readNetFromDarknet(modelConfiguration, modelWeights);
net.setPreferableBackend(DNN_BACKEND_CUDA);
net.setPreferableTarget(DNN_TARGET_CUDA);
```

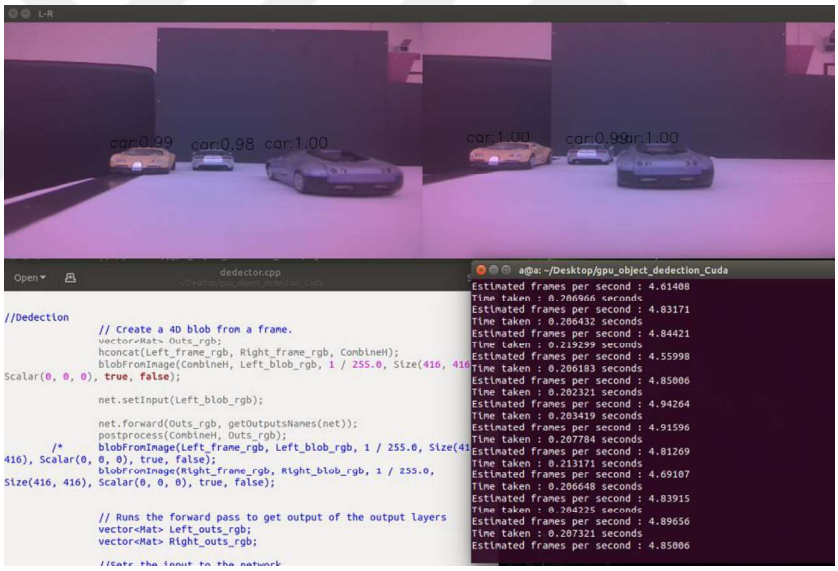
Şekil 4.8. Yolo Verileri

Yapılan bu işlemden sonra test nesneleri aşağıdaki şekilde yer almaktadır.Saniyede en yüksek 2.5 kare alındığından gerçek zamanlı işlemelere uygun değildir. Hızlandırmak için yapılan çalışma sonuç tanımlama hızı Saniyede en yüksek 5 kareye gelmiştir. Çerçeve çizimleri kapatılmış

ve tanımlama için resim birleştirilerek gönderilmiştir. İşlem sonuçları Şekil 4.9 ve 4.10 da yer almaktadır.



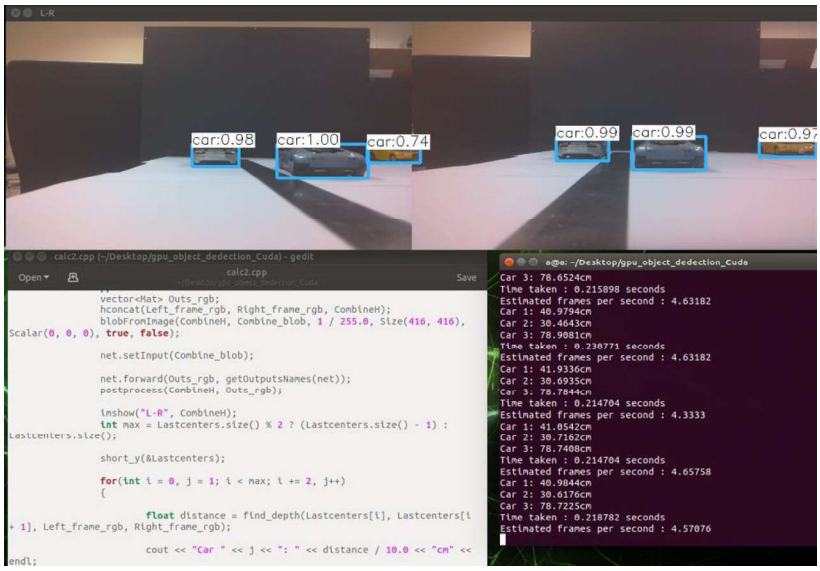
Şekil 4.9. Yolo nesne Tanıma



Şekil 4.10.FPS Artışı

4.2.2. Nesne Tabanlı Uzaklık Tespiti

Nesne tanıma ve uzaklık hesaplama adımları tek program içinde toplanmış ve ekran görüntüleri Şekil 4.11'de verilmiştir.



Şekil 4.11. Son Çalışma

4.2.3. Nesne Tabanlı Uzaklık Tespiti Sonuçları

Gerçek ve ölçülen değerler aşağıdaki Tablo 4.4 ve 4.5'te karşılaştırılmıştır.

Tablo 4.4.FPS Karşılaştırması

Nesne çizimi	Min fps	Max fps
Var	1.8	3.8
Yok	2.76	4.32
Birleşik	4.88	5.0

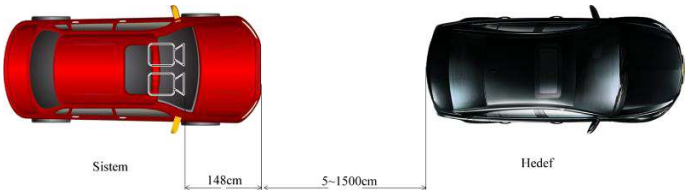
Tablo 4.5. Uzaklık Değerleri

Gerçek Uzaklık /cm	Ölçülen Uzaklık/cm	Hata/cm	Hata Yüzdesi %
10	14,72	-4,72	47,2
15	28,3	-13,3	88,66666667
20	25,82	-5,82	29,1
25	24,46	0,54	2,16
30	23,53	6,47	21,56666667
35	22,94	12,06	34,45714286
40	22,36	17,64	44,1
45	22,23	22,77	50,6
50	28,4	21,6	43,2

4.3. Gerçek Hayata Uyarlama

Yaptığımız platform Şekil 4.12' de görüldüğü gibi sistem kontrol aracına ön cama yerleştirilmiştir. Yerleşim aracın en ön uç noktasından 148 cm uzaklıkta olacak şekilde

konumlandırılırmıştır. Ölçülen hedef araç ise hareketli bir şekilde 5 ila 1500 cm arasında ileri geri hareket halinde ölçümler yapılmıştır.



Şekil 4.12. Görsel Gösterim

Araç içi montaj önden ve içten alınan görüntüleri Şekil 4.13 ve Şekil 4.14'te yer almıştır.

Şekil 4.13. Önden Görünüm

Şekil 4.14. İçeri Görünüm

Yapılan çalışmada elde edilen görüntüler Şekil 4.15,16 ve 17'de ölçüm değerleri ise Tablo 4.6 ve 7'de yer almıştır.

Şekil 4.15. Ölçüm 1500cm

Şekil 4.16. Ölçüm 1000cm

Şekil 4.17. Ölçüm 150cm

Tablo 4.6.FPS Karşılaştırması

Nesne çizimi	Min fps	Max fps
Var	1.8	3.0
Yok	2.76	3.3
Birleşik	3.5	4.9

Tablo 4.7. Ölçüm Değerleri

Gerçek Uzaklık /cm	Ölçülen Uzaklık/cm	Hata/cm	Hata Yüzdesi %
150	200,524	50,524	33,68266667
300	201,24	98,76	32,92
500	499,236	0,764	0,1528
700	702,365	20,365	2,909285714
950	943,245	349,443	36,78347368
1000	980,925	19,075	1,9075
1100	1095,256	995,256	90,47781818
1300	1269,785	30,215	2,324230769
1400	1401,354	298,646	21,33185714
1450	1402,785	47,215	3,256206897
1500	1612,1549	312,1549	20,81032667

5. SONUÇLAR VE ÖNERİLER

Günümüzde elektrikli araçların, nesnelerin internetinin ve akıllı şehir konseptlerinin gerçek hayata uygulanması için çalışmaların yaygınlaşması süreçlerine katkı olarak otonom araçların bu süreç içine entegre edilmesi en önemli bölümünü oluşturmaktadır. Otonom araçların nesne algılama doğruluğunun artması canlılara zarar vermesinin yanı sıra trafiği aksatmaması açısından önemlidir. Bu yüzden kullanılan sensörlerin, algoritmaların ve gömülü sistem platformlarının sıfıra yakın hata ile hatta hatasız çalışması gerektiğini ortaya koymaktadır. Yaptığımız bu çalışma bu tip sistemin en düşük maliyetle yapılıp yapılamayacağı üzerinedir. Ucuz kameralar kullanılarak ilk önce kamera gürültüleri düzeltilmeye çalışılmış ardından nesne takip algoritmasıyla bulunan nesnenin uzaklık bulma algoritması kullanılarak uzaklığı hesaplanmış ve ardından görüntüde bulunan sadece araçları tanıyacak şekilde YOLO tabanlı bir sistem entegre edilmiştir.

Alınan sonuçlara göre Jetson Nano platformu gerçek zamanlı nesne tanımlama ve uzaklık tahmini konusunda çok yetersiz olduğu yapılan çalışmalarca ortaya konulmuştur. Gerçek hayatta saniyede 5 kare işlem hızıyla seyir halinde hiçbir araca kullanılamayacağı verilerle sabitlenmiştir.

Ayrıca literatür taramasında yapılan çalışmalarının işlem gücü bakımından Jetson Nano platformundan kat be kat üstün olduğu sistemlerde denendiği ve bu sayede sonuçların gerçek hayata yakın olduğu tespit edilmiştir. Fakat bu sistemler boyut olarak bir araca montajlanamayacak kadar yüksek güce ve fiziksel alana ihtiyacı duymaktadır. Edinilen bilgi ve tecrübelerden yola çıkarak yeni tasarlanması düşünülen sistemlerin maliyet gözetimi yapılmadan tasarlanması ve donanım tarafında tensor ünitesi bulunan işlemciler kullanılması öngörülmektedir. Bu sayede var olan evrişimli sinir ağları çok hızlı ve gerçeğe yakın işlem kapasitesine ulaşabilir. Örnek olarak Şekil 5.1'te yer alan bir platform seçilebilir.

		Jetson TX2 Serisi				Jetson Xavier NX Serisi		Jetson AGX Xavier Serisi			Jetson Orin NX	Jetson AGX Orin
Ürün	Jetson Nano	TX2 NX	TX2 4 GB	TX2	TX2i	Jetson Xavier NX 16 GB	Jetson Xavier NX	Jetson AGX Xavier 64 GB	Jetson AGX Xavier	Jetson AGX Xavier Industrial		
Yapay Zeka Performansı	472 GFLOPS	1,33 TFLOPS			1,26 TFLOPS	21 TOPS		32 TOPS		30 TOPS	100 TOPS	200 TOPS

Şekil 5.1. Jetson Platformları [86]

Başka bir çözüm yöntemi ise akıllı veya kalibre edilmiş kameralar olabilir. Yeni nesil içinde fpga ile görüntü tanıma yapabilen ve bu görüntüdeki matematiksel değerleri üst katmana aktarabilen bir kamera donanımı kullanılabilir ya da tasarlanabilir. Örnek olarak Şekil 5.2'de yer alan ZED kamerası verilmiştir.

Şekil 5.2. ZED Kamera [87]

Günümüzde ise sensör bulutu yöntemiyle bu donanımsal maliyet yazılım ve iş güçü alanına kaydırılmıştır**.** Böylece bir araçta birden fazla radar, kamera ve konum tespit sensörlerinden gelen veri akışlarını eş zamanlı işleyecek bir yapı kullanılmaktadır.

KAYNAKLAR

[1]. Emissions-control sticker. *Low emission zone*. 15 Ocak 2022 tarihinde https://www.bmuv.de/en/topics/air-noise-mobility/air/emissions-control-sticker-low-emission-zone adresinden erişildi.

[2]. Sun, S., Petropulu, A. P., ve Poor H. V. (2020). MIMO radar for advanced driver-assistance systems and autonomous driving: advantages and challenges. *IEEE Signal Processing Magazine*, 98-117

[3]. Toyota. *Corolla Kullanım Kılavuzu*. 28 Ağustos 2019 tarihinde http://turkiye.toyota.com.tr/Files/KILAVUZ/COROLLA_HYBRID/COROLLA_HYBRID_OM12L86T.pdf adresinden erişildi.

[4]. Copeland, M. (2016). *What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning?* 16 Mart 2020 tarihinde https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machinelearning-deep-learning-ai/ adresinden erişildi.

[5]. Büber, E. (2018*). Günlük Yaşamda Makine Öğrenmesi Uygulama Örnekleri*. 16 Mart 2020 tarihinde https://medium.com/@EbubekirBbr/g%C3%BCnl%C3%BCk-ya%C5%9Famdamakine-%C3%B6%C4%9Frenmesi-uygulama-%C3%B6rnekleri-3aba1cd49f36 adresinden erişildi.

[6]. Kurt, F. (2015). *Derin Öğrenme (Deep Learning) Nedir?*. 17 Mart 2020 tarihinde https://www.derinogrenme.com/2015/07/21/derin-ogrenme-deep-learning-nedir/ adresinden erişildi.

[7]. Rezaei, M., Terauchi,M. ve Klette, R. (2015). Robust vehicle detection and distance estimation under challenging lighting conditions, *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5

[8]. Moutarde,F. ,Stanciulescu, B. ve Breheret,A. (2008) Real-time visual detection of vehicles and pedestrians with new efficient AdaBoost features. *Workshop Planning Perception Navigation Intelligent Vehicles*, 70–75.

[9]. Nguyen,T.T., Grabner, H. ,Bischof, H. ve Bruber,B. (2007). On-line boosting for car detection from aerial images. *IEEE Int. Conf. Research Innovation Vision Future*, 87–95.

[10]. Wen, X., Yuan, H., Yang, C., Song, C., Duan, B. ve Zhao, H. (2007). Improved haar wavelet feature extraction approaches for vehicle detection. *IEEE Conf. Intelligent Transportation Systems*,1050–1053.

[11]. Haar, A. (1910). Zur theorie der orthogonalen funktionensysteme. (Erste Mitteilung). *Mathematische Annalen,* 331-371.

[12]. Klette, R., Kr¨uger, N., Vaudrey, T., Pauwels, K., van Hulle, M., Morales, S., Kandil, F., Haeusler, R., Pugeault, N., Rabe, C. ve Lappe, M. (2011). Performance of correspondence algorithms in

vision-based driver assistance using an online image sequence database*. IEEE Trans. Vehicular Technology*,2012–2026.Das

[13]. Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University Press.

[14]. Koks, D. ve Challa, S. (2003). *An Introduction to Bayesian and Dempster-Shafer Data Fusion*.

[15]. KITTI Benchmark. *CAR Detection Benchmark*. 19 Mayıs 2021 tarihinde www.cvlibs.net/datasets/kitti/evalobject.php adresinden erişildi.

[16]. Felzenszwalb, P., Girshick, R. ve Mcallester, D. (2010). Visual object detection with deformable part models. *Communications of the ACM*, 2241-2248.

[17]. Felzenszwalb, P., Girshick, R. ve Mcallester, D. ve Ramanan, D. (2010). Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions On Pattern Analysis And Machine İntelligence*, 1627-1645

[18]. Zeng,L. (2009). High data rate multiple input multiple output (MIMO) optical wireless communications using white LED lighting. *IEEE J. Sel.Areas Commun*,1654-1662.Asd

[19]. Elgala, H., Mesleh, R. ve Haas,H. (2011). Indoor optical wireless communication: Potential and state-of-the-art, *IEEE Commun. Mag*.,56-62.

[20]. Jovicic, A., Li, J. ve Richardson,T. (2013). Visible light communication: Opportunities, challenges and the path to market. *IEEE Commun. Mag*.,26-32.

[21]. Do,T.H. ve Yoo,M. (2016). An indepth survey of visible light communication based positioning systems*. Sensors, vol. 16, no. 5*, 678.

[22]. Kumar, N. (2014). Visible light communication based trafic information broadcasting systems. *Int. J. Future Comput. Commun.,* 26.

[23]. O'Brien, D. , Zeng, L., Le Minh, H. , Faulkner, G., Walewski,J. ve Randel, S. (2008). Visible light communications: challenges and possibilities. *IEEE 19th Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, 1-5.

[24]. Khan, L. U. (2017). Visible light communication: applications, architecture,standardization and research challenges*. Digit. Commun. Netw*., 78-88.

[25]. Song, G., Lee,K, Y. ve Lee, J. W. (2008). Vehicle detection by edge-based candidate generation and appearance-based classication. *IEEE Intell. Vehicles Symp*.,428-433.

[26]. Tram,V. T. B. ve Yoo,M. (2018). Vehicle-to-vehicle distance estimation using a low-resolution camera based on visible light communications. *IEEE Access, vol. 6*, 4521-4527.

[27]. Haseeb, M. A., Guan, J., Ristić-Durrant, D. ve Gräser, A. (2018). DisNet: A novel method for distance estimation from monocular camera, *IEEE Institute of Automation, University of Bremen*,1-6

[28]. Shift2Rail. *SMART Projesi*. 20 Mart 2021 tarihinde http://www.smartrail-automation-project.net adresinden erişildi.

[29]. Redmon, J. ve Farhadi, A. (2018). YOLOv3: An incremental improvement. *arXiv:1804.02767* , 1-5

[30]. Veri Kümesi. *COCO dataset*. 23 Nisan 2021 tarihinde https://arxiv.org/pdf/1405.0312.pdf adresinden erişildi.

[31]. Yuxiang, G., Kumazawa, I. ve Kaku, C. (2018). Blind spot obstacle detection from monocular camera ımages with depth cues extracted by cnn. *Automotive Innovation*.

[32]. Godard, C., Aodha, O. ve Brostow, G.J. (2017). Unsupervised monocular depth estimation with left-right consistency. *arXiv:1609.03677v3*

[33]. Guo,Y.X. (2017). *Investigation ofmonocular depth cues obtained through coarse-to-fine image analysis*. Yayınlanmış yüksek lisans tezi, Tokyo Teknoloji Üniversitesi, Tokyo.

[34]. Forster, C., Zhang, Z, Gassner, M., Werlberger, M.ve Scaramuzza, D. (2017). SVO: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*,249–265.

[35]. Wang, S., Clark,R., Wen, H. ve Trigoni, N. (2017). DeepVO: Towards end-to-end visual odometry with deep recurrent convolutionalneural networks. *In International Conference on Robotics and Automation (ICRA)*, 2043–2050.

[36]. Wang, S., Clark,R., Wen, H. ve Trigoni, N. (2018). End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *International Journal of Robotics Research*, 513–542

[37]. Jiao, J., Jiao, J., Mo, Y, Liu, W. ve Deng, Z. (2018). MagicVO: End-to-end monocular visual odometry through deep bidirectional recurrent convolutional neural network. *arXiv:1811.10964*.

[38]. Xue, F., Wang, Q., Wang, X., Dong, W., Wang, J. ve Zha, H. (2019). Guided feature selection for deep visual odometry. *In Asian Conference on Computer Vision*.

[39]. Geiger, A., Lenz, P. ve Urtasun, R. (2012). Are we ready for autonomous driving? the KITTI vision benchmark suite. *Computer Vision and Pattern Recognition Conference*, 3354–3361.D

[40]. Kreuzig, R., Ochs M. ve Mester, R. (2019). Distancenet: estimating traveled distance from monocular ımages using a recurrent convolutional neural network. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 1258-1266

[41]. Zhang, J., Hu,S. ve Shi,H. (2018). Deep learning based object distance measurement method for binocular stereo vision blind area. *International Journal of Advanced Computer Science and Applications*, 9.

[42]. Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 666.D

[43]. Reghenzani, F., Massari, G. ve Fornaciari,W. (2019). The real-time linux kernel: A survey on Preempt_RT. *ACM Comput. Surv.*, 52.

[44]. Palamut, S., Gönültaş, T., Elewi, A. ve Avaroğlu, E. (2019). Task scheduling algorithms and resource access protocols in real time systems. *International Artificial Intelligence and Data Processing Symposium*, 1-6.

[45]. Kopetz, H. (2011). *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Springer.

[46]. Liu, J. W. S. W. (2000). Real-Time Systems. Prentice Hall PTR.

[47]. V. Yodaiken. *The RTLinux Manifesto*. 25 Nisan 2021 tarihinde http://www.yodaiken.com/papers/rtlmanifesto.pdf adresinden erişildi

[48]. Jacobs, I. (1978). Atlanta fiber system experiment: Overview. *Bell Syst. Tech. J.*, 1717–1721.

[49]. Yaghmour, K. (2001). *Adaptive Domain Environment for Operating Systems*. Technical Report.

[50]. Mantegazza, P., Dozio, L. ve Papacharalambous, S. (2000). RTAI: Real time application interface. *Linux J.*,10.

[51]. Gerum, P. (2004). Xenomai implementing a rtos emulation framework on Gnu/Linux. *Nucleus*, 1–12.

[52]. Finney,S. A. (2001). Real-time data collection in Linux: A case study. *Behav. Res. Methods, Instruments, Comput.*, 167–173.

[53]. Kaehler, A. ve Bradski, G. (2017). *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library (1st ed.)*. O'Reilly Media.

[54]. Vision. *Kamera Kalibrasyonu*. 20 Haziran 2021 tarihinde http://www.vision.caltech.edu/bouguetj/calib_doc/ adresinden erişildi.

[55]. OpenCv. *Calibrate Camera Fonksiyonu*. 20 Haziran 2021 tarihinde https://docs.opencv.org/4.5.2/d9/d0c/group__calib3d.html#ga3207604e4b1a1758aa66acb6ed5aa65d adresinden erişildi.

[56]. Vikipedi. *Tek Kart Bilgisayar*. 26 Ocak 2021 tarihinde https://tr.wikipedia.org/wiki/Single-board_computer adresinden erişildi.

[57]. Nvdia. *Jetson Nano Developer Kit.* 26 Ocak 2021 tarihinde https://developer.nvidia.com/embedded/jetson-nano-developer-kit adresinden erişildi.

[58]. Raspberry. *Camera Module V2*. 18 Mayıs 2021 tarihinde https://www.raspberrypi.org/products/camera-module-v2/ adresinden erişildi.

[59]. PyImageSearch. *Kameradan Uzaklık Bulma*. 20 Haziran 2021 tarihinde https://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-pythonopencv/ adresinden erişildi.

[60]. PD *How to Measure Your Pupillary Distance.* 21 Haziran 2021 tarihinde https://www.zennioptical.com/measuring-pd-infographic adresinden erişildi.

[61]. GNU Projesi. *GNU İşletim Sistemi*. 10 Aralık 2021 tarihinde https://www.gnu.org/gnu/thegnuproject.tr.htm adresinden erişildi.

[62]. Pardus Linux. *Linux'un Doğuşu*. 15 Aralık 2021 tarihinde http://pardustanbul.blogspot.com/2008/05/linuxun-douu.html adresinden erişildi.

[63]. Palamut, S. *Jetson Nano Kurulum* 17 Ocak 2022 tarihinde https://books.google.com.tr/books?id=6UojEAAAQBAJ&lpg=PR1&hl=tr&pg=PA3#v=onepage&q&f=false adresinden erişildi.

[64]. Nvdia. *Jetson Nano İşletim Sistemi*. 01 Şubat 2021 tarihinde https://developer.nvidia.com/embedded/downloads#?search=Jetson%20Nano adresinden erişildi.

[65]. Nvdia. *Jetson Nano İmaj Yazma İşlemi.* 01 Şbat 2021 tarihinde https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write adresinden erişildi.

[66]. Etcher. *İmaj Yazma Programı*. 01 Şubat 2021 tarihinde https://www.balena.io/etcher/ adresinden erişildi.

[67]. Ubuntu Login. *İlk Açılış işlemi* 01 Şubat 2021 tarihinde https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#setup adresinden erişildi.

[68]. Ubuntu Dektop*. Otomatik Açılış İşlemi*. 01 Şubat 2021 tarihinde https://vitux.com/how-to-enable-disable-automatic-login-in-ubuntu-18-04-lts/ adresinden erişildi.

[69]. Nmtui. *Network ayarları*. 01 Şubat 2021 tarihinde http://manpages.ubuntu.com/manpages/bionic/man1/nmtui.1.html adresinden erişildi.

[70]. Nvdia. *Jetson Nano bilgileri*. 01 Şubat 2021 tarihinde https://docs.nvidia.com/jetson/l4t/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/hw_setup_jetson_io.html#wwpID0E0TB0HA adresinden erişildi.

[71]. Ridgerun. *Jetson Nano derleme bilgileri*. 01 Şubat 2021 tarihinde https://developer.ridgerun.com/wiki/index.php?title=Jetson_Nano/Development/Building_the_Kernel_from_Source#3._Compile_kernel_and_dtb adresinden erişildi.

[72]. Cyclictest. *Gerçek zamanlı çekirdek testi*. 01 Şubat 2021 tarihinde https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/cyclictest/start adresinden erişildi.

[73]. OSADL. *The Open Source Automation Development Lab*. 17.01.2022 tarihinde https://www.osadl.org/ adresinden erişildi.

[74]. Vikipedi. *C programlama dili.* 14 Nisan 2019 tarihinde https://tr.wikipedia.org/wiki/C_(programlama_dili) adresinden erişildi.

[75]. OpenCv. Tanım. 17. Ocak 2022 tarihinde https://opencv.org adresinden erişildi.

[76]. OpenCv. *OpenCv Hakkında*. 18 Mayıs 2019 tarihinde https://opencv.org/about/ adresinden erişildi.

[77]. Nvdia. *Cuda Tollkit.* 16 Mart 2019 tarihinde https://developer.nvidia.com/cuda-toolkit adresinden erişildi.

[78]. Yolo. *Performans Değerleri*. 18 Mart 2019 tarihinde https://pjreddie.com/darknet/yolo/ adresinden erişildi.

[79]. Khan, S., Rahmani, H., Shah, S. A. A., Bennamoun, M., Medioni,G. ve Dickinson,S. (2018). *A Guide to Convolutional Neural Networks for Computer Vision*. Morgan & Claypool.A

[80]. Le Cun, Y., Jackel, L.D., Boser, B., Denker, J.S., Graf, H.P., Guyon, I., Henderson, D., Howard, R.E. ve Hubbard, W. (1989). Handwritten digit recognition: applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 41-46.

[81]. Yolo. *Unified, Real-Time Object Detection*. 19 Mart 2019 tarihinde https://pjreddie.com/media/files/papers/yolo_1.pdf adresinden erişildi.

[82]. Darknet. *YOLO projesi*. 14 Kasım 2020 tarihinde https://github.com/AlexeyAB/darknet adresinden erişildi.

[83]. Redmon, J., Divvala, S., Girshick, R. ve Farhadi, A. (2016). You only look once: unified, real-time object detection. *IEEE Conference on Computer Vision and Pattern Recognition*,779-788.

[84]. The PASCAL Visual Object Classes. *The PASCAL VOC Project*. 17 Temmuz 2019 tarihinde http://host.robots.ox.ac.uk/pascal/VOC/ adresinden erişildi.

[85]. Palamut, S. (2021). *Kamera Kalibrasyon Örneği*. 15 Ocak 2022 tarihinde https://github.com/SalihPalamut/OpenCvCameraCalibration adresinden erişildi.

[86]. Nvdia. Jetson Platformları Modülleri Karşılaştırın. 15 Ocak 2022 tarihinde https://www.nvidia.com/tr-tr/autonomous-machines/embedded-systems/ adresinden erişildi.

[87]. ZED. Stereo Camera Module. 18 Ocak 2022 tarihinde https://www.stereolabs.com/zed/ adresinden erişildi.