KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

İKİ BOYUTLU GÖRÜNTÜLERDE DERİNLİK BELİRLEME YÖNTEMLERİNDEKİ ZORLUKLARIN ANALİZİ

ZEYNEP AKIN

KOCAELİ 2023

KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

İKİ BOYUTLU GÖRÜNTÜLERDE DERİNLİK BELİRLEME YÖNTEMLERİNDEKİ ZORLUKLARIN ANALİZİ

ZEYNEP AKIN

Prof. Dr. Ahmet SAYAR
Danışman, Kocaeli Üniv. .....

Doç. Dr. Pınar ONAY DURDU
Jüri Üyesi, Kocaeli Üniv. .....

Doç. Dr. Hasan BULUT
Jüri Üyesi, Ege Üniv. .....

Tezin Savunulduğu Tarih: 15.12.2022

ETİK BEYAN VE ARAŞTIRMA FONU DESTEĞİ

Kocaeli Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez/proje çalışmasında,

- Bu tezin/projenin bana ait, özgün bir çalışma olduğunu,
- Çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı,
- Bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi,
- Bu çalışmanın Kocaeli Üniversitesi'nin abone olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş olduğu ölçütlere uygun olduğunu,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Tezin/Projenin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez/proje çalışması olarak sunmadığımı,

beyan ederim.

■ Bu tez/proje çalışmasının herhangi bir aşaması hiçbir kurum/kuruluş tarafından maddi/alt yapı desteği ile desteklenmemiştir.

☐ Bu tez/proje çalışması kapsamında üretilen veri ve bilgiler ..... tarafından ..... no'lu proje kapsamında maddi/alt yapı desteği alınarak gerçekleştirilmiştir.

Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

.....
(İmza)
Zeynep AKIN
(Öğrencinin Adı Soyadı)

YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI

Fen Bilimleri Enstitüsü tarafından onaylanan lisansüstü tezimin/projemin tamamını veya herhangi bir kısmını, basılı ve elektronik formatta arşivleme ve aşağıda belirtilen koşullarla kullanıma açma izninin Kocaeli Üniversitesi'ne verdiğimi beyan ederim. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin/projemin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanımı bana ait olacaktır.
Tezin/projenin kendi özgün çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin/projenin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanılması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.
Yükseköğretim kurulu tarafından yayınlanan ***"Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"*** kapsamında tezim aşağıda belirtilen koşullar haricinde YÖK Ulusal Tez Merkezi/ Kocaeli Üniversitesi Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

☐ Enstitü yönetim kurulu kararı ile tezimin/projemin erişime açılması mezuniyet tarihinden itibaren 2 yıl ertelenmiştir.
☐ Enstitü yönetim kurulu gerekçeli kararı ile tezimin/projemin erişime açılması mezuniyet tarihinden itibaren 6 ay ertelenmiştir.
■ Tezim/projem ile ilgili gizlilik kararı verilmemiştir.

.....
(İmza)
Zeynep AKIN
(Öğrencinin Adı Soyadı)

ÖNSÖZ VE TEŞEKKÜR

İki boyutlu görüntülerde derinlik belirleme yöntemlerindeki zorlukların analizi konusunda bana çalışma fırsatı veren ve aynı zaman beni teşvik eden değerli hocam Prof. Dr. Ahmet SAYAR'a teşekkür ederim. Ayrıca hayatım boyunca beni destekleyen eşim ve aileme de sonsuz minnet duygularımı sunarım.

Kasım – 2022 Zeynep AKIN

İÇİNDEKİLER

ETİK BEYAN VE ARAŞTIRMA FONU DESTEĞİ i
YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI ii
ÖNSÖZ VE TEŞEKKÜR iii
İÇİNDEKİLER iv
ŞEKİLLER DİZİNİ v
TABLOLAR DİZİNİ vi
SİMGELER VE KISALTMALAR DİZİNİ vii
ÖZET viii
ABSTRACT ix
1. GİRİŞ 1
2. İKİ BOYUTLU GÖRÜNTÜLERDE DERİNLİK TAHMİNİ YÖNTEMLERİNE GENEL BAKIŞ 4
2.1. Temel Kavramlar 4
2.2. İki Boyutlu Görüntülerde Derinlik Tahmini Yöntemleri 7
2.2.1. Üç Boyutlu Sınırlayıcı Kutu Tahmini 7
2.2.2. Homografi Tabanlı IPM 9
2.2.3. Görüntü Segmentasyonu Kullanımı 10
2.2.3.1 Görüntü Segmentasyonu Türleri 12
2.2.4. Referans Nesne Kullanımı 14
3. REFERANS NESNE KULLANIMI İLE GERÇEK ZAMANLI DERİNLİK TAHMİNİ UYGULAMASI 16
3.1. Mimari 16
3.1.1. Akış Seması 16
3.1.2. Use Case Diyagramı 17
3.2. Malzeme ve Yöntem 17
3.3. Bulgular ve Tartışma 22
4. GÖRÜNTÜ İŞLEME VE DERİN ÖĞRENME İLE İKİ BOYUTLU GÖRÜNTÜDEN MESAFE TAHMİNİ 26
4.1. Mimari 26
4.1.1. Akış Seması 26
4.1.2. Sınıf Diyagramı 27
4.1.3. Use Case Diyagramı 27
4.2. Malzeme ve Yöntem 28
4.3. Bulgular ve Tartışma 32
5. ÇÖZÜM YAKLAŞIMLARININ ÖZETİ VE KARŞILAŞILAN ZORLUKLAR 34
5.1. Literatürdeki Çözüm Yaklaşımlarının Özeti 34
5.1.1. 3D Nesne Algılama ile Derinlik Tahmini 36
5.1.2. Segmentasyon Kullanımı ile Derinlik Tahmini 37
5.1.3. IPM Kullanımı ile Derinlik Tahmini 37
5.1.4. Referans Nesne Kullanımı ile Derinlik Tahmini 39
5.2. Derinliği Belirlemede Karşılaşılan Zorluklar 40
6. SONUÇ 45
KAYNAKLAR 47
EKLER 53
KİŞİSEL YAYIN VE ESERLER 61
ÖZGEÇMİŞ 62

ŞEKİLLER DİZİNİ

Şekil 2.1. Bir çizimin x ve y koordinatları üzerindeki noktaları hakkında 4
Şekil 2.2. RGB görüntüden derinlik görüntüsüne dönüşüm 5
Şekil 2.3. Kamera koordinat sistemi 6
Şekil 2.4. RGB görüntüye IPM dönüşümü 9
Şekil 2.5. Bilineer enterpolasyon 10
Şekil 2.6. Eşik bazlı segmentasyon örneği 11
Şekil 2.7. Canny yöntemi kullanılarak görüntü segmentasyonu 11
Şekil 2.8. K-means algoritması ile görüntü segmentasyonu 12
Şekil 2.9. Görüntü segmentasyon türleri 13
Şekil 2.10. Orijinal Görüntü- Nesne Tespiti- Derinlik Tahmini- Semantik Segmentasyon- Örnek Segmentasyon 14
Şekil 3.1. Referans nesne ile gerçek zamanlı derinlik tahmini akış şeması 16
Şekil 3.2. Referans nesne kullanımı ile gerçek zamanlı derinlik tahmini use case diyagramı 17
Şekil 3.3. Aruco Marker 5x5 18
Şekil 3.4. Aruco Marker üzerinde piksel/cm oranı uygulaması 19
Şekil 3.5. Aruco Marker Ekran Uzaklığı Tespiti 20
Şekil 3.6. Kupa ekran uzaklığı tespiti 20
Şekil 3.7. Kahve fincanı ile 3 boyutlu derinlik gösterimi 22
Şekil 3.8. Kupa ile 3 boyutlu derinlik gösterimi 22
Şekil 3.9. Uygulamada kullanılan kupa çeşitleri 23
Şekil 3.10. Kupa kulpu uzunluk tespiti 25
Şekil 4.1. İki boyutlu görüntüden mesafe tahmini uygulaması akış diyagramı 26
Şekil 4.2. Görüntü işleme ve derin öğrenme ile iki boyutlu görüntüden mesafe tahmini sınıf diyagramı 27
Şekil 4.3. Görüntü işleme ve derin öğrenme ile iki boyutlu görüntüden mesafe tahmini use case diyagramı 27
Şekil 4.4. Bursa Mobese kameralarından bir görüntü 28
Şekil 4.5. Bursa Mobese kamera görüntüsü örneği 29
Şekil 4.6. Şekil 4.5' deki görüntünün transformu 29
Şekil 4.7. Mobese görüntülerinden nesne ve mesafe tespiti 30
Şekil 4.8. Maskelenmiş görüntüden nesne ve mesafe tahmini 30
Şekil 4.9. İki boyutlu görüntüden araç ve mesafe tespiti örneği 31
Şekil 4.10. Maskelenmiş görüntüden araç ve mesafe tespiti örneği 31
Şekil 4.11. Uygulamada hatalı koordinat seçimi ve sonucu 32
Şekil 4.12. Uygulamada hatalı koordinat seçimi ve sonucu 32
Şekil 4.13. Uygulamada görüntü engeli ve sonucu 33
Şekil 5.1. Sun ve arkadaşlarının çalışmasından görüntü 39
Şekil 5.2. Derinlik tahminindeki küçük hatalar 41
Şekil 5.3. Nesnelerin gölgeleri sayesinde algılanması. 41
Şekil 5.4. Değiştirilmemiş, gri tonlamalı, yanlış renkli ve semantik rgb görüntüleri için örnek görüntüler ve derinlik haritaları 42
Şekil 5.5. Trafik akışı sırasında nesne tanıma yapan videodan bir görüntü 42
Şekil 5.6. Derinlik tahmini ayrıntı eksikliği hatası 43

TABLOLAR DİZİNİ

Tablo 3.1. Objectron veri seti içeriği ve dağılımı 21
Tablo 3.2. Uygulamada kullanılan kupa çeşitlerine göre bulunan sonuçlar 23
Tablo 3.3. Referans nesne ile gerçek zamanlı derinlik tahmininde karşılaşılan zorluklar 24
Tablo 5.1. Literatürdeki çözüm yaklaşımlarının özeti 35
Tablo 5.2. İki boyutlu görüntüden derinlik tahmini sırasında karşılaşılan zorluklar 44

SİMGELER VE KISALTMALAR DİZİNİ

θ	: Azimut
φ	: Fi
α	: Alfa

Kısaltmalar

AR	: Augmented Reality (Artırılmış Gerçeklik)
BEV	: Bird's Eye View (Kuş Bakışı Görüntü)
CNN	: Convolutional Neural Network (Evrişimli Sinir Ağları)
CPU	: Central Process Unit (Merkezi İşlem Birimi)
DNN	: Deep Neural Network (Derin Sinir Ağı)
EPSNET	: Efficient Panoptic Segmentation Network (Verimli Panoptik Segmentasyon Ağı)
FASTER R-CNN	: Faster Region Based Convolutional Networks (Daha Hızlı Bölge Tabanlı Evrişimli Ağlar
FCM	: Fuzzy C-Means (Bulanık C-Means Kümeleme)
FPSNET	: Fast Panoptic Segmentation Network (Hızlı Panoptik Segmentasyon Ağı)
IPM	: Inreverse Perspective Mapping (Ters Perspektif Haritalama)
KITTI	: Karlsruhe Institute of Technology and Toyota Technological Institute (Karlsruhe Teknoloji Enstitüsü ve Toyota Teknoloji Enstitüsü)
MASK R-CNN	: Mask Region Based Convolutional Neural Network (Maske Bölge Tabanlı Evrişimsel Sinir Ağı)
OPENCV	: Open Source Computer Vision Library (Açık Kaynak Bilgisayar Görüsü Kütüphanesi)
SEGNET	: Semantic Segmentation Model (Anlamsal Bölümleme Modeli)
SLAM	: Simultaneous Localization and Mapping (Eşzamanlı Konumlama ve Haritalama)
SMOKE	: Single-Stage Monocular 3D Object Detection via Keypoint Estimation (Anahtar Nokta Tahmini ile Tek Aşamalı Monoküler 3D Nesne Algılama)
U-NET	: Convolutional Networks for Biomedical Image Segmentation (Biyomedikal Görüntü Segmentasyonu için Evrişimsel Ağlar)
UPSNET	: Unified Panoptic Segmentation Network (Birleşik Panoptik Segmentasyon Ağı)
RGB	: Red Green Blue (Kırmızı Yeşil Mavi)
YOLACT	: You Only Look At CoefficientTs (Siz Sadece Katsayılara Bakın)
YOLO	: You Only Look Once (Sadece Bir Kez Bak)
2D	: Two Dimensional (İki Boyutlu)
3D	: Three Dimensional (Üç Boyutlu)

İKİ BOYUTLU GÖRÜNTÜLERDE DERİNLİK BELİRLEME YÖNTEMLERİNDEKİ ZORLUKLARIN ANALİZİ

ÖZET

Bir nesne resmedildiğinde, projeksiyonu üç boyutlu sahneden iki boyutlu düzleme yansıtılmaktadır. Bu gibi durumlarda, nesnenin derinlik bilgisi kaybolmaktadır. Derinlik, bir nesnenin mesafesinin ve üç boyutlu algısının hesaplanmasını sağlamaktadır. Aynı sahnenin çeşitli perspektiflerinden iki veya daha fazla iki boyutlu resmini kullanmak, göreceli derinlik bilgilerinin tahmin edilmesine yardımcı olmaktadır. Bununla birlikte, yalnızca bir iki boyutlu görüntü kullanılırsa, derinlik tahmini probleminin çözülmesi çok zor hale gelmektedir. Bu çalışmada, bu problemin teorik temeli sunulmakta, önerilen çözüm yaklaşımları özetlemekte ve karşılaşılan zorlukları incelenmektedir. Ek olarak referans nesne kullanımı ile gerçek zamanlı bir derinlik tahmini uygulaması oluşturulmaktadır. Geliştirilen bir diğer uygulama ise iki boyutlu bir görüntüden görüntü işleme ve derin öğrenme kullanılarak mesafe tahminidir. Bu uygulamalarda elde edilen bulgular doğrultusunda karşılaşılan zorluklar anlatılmaktadır.

Anahtar Kelimeler: Derin Öğrenme, Derinlik Tahmini, Görüntü İşleme, Homografi, Üç Boyutlu Sınırlayıcı Kutu.

ANALYSIS OF CHALLENGES IN DETERMINATION METHODS IN 2-D IMAGES

ABSTRACT

When an object is pictured, it's projection is changed from 3D scene to 2D plane. In these cases, the depth information of the object is lost. Depth provides the calculation of the distance and three-dimensional perception of an object. Using two or more two-dimensional images of the same scene from various perspectives helps to estimate relative depth information. However, if only one two-dimensional image is used, the depth estimation problem becomes very difficult to solve. In this study, the theoretical basis of this problem is presented, the proposed solution approaches are summarized and the difficulties encountered are examined. In addition, a real-time depth estimation application is created with the use of reference objects. Another application developed is distance estimation from a two-dimensional image using image processing and deep learning. The difficulties encountered in these applications are explained in line with the findings obtained.

Keywords: Deep Learning, Depth Estimation, Homography, Image Processing, 3D Bounding Box.

1. GİRİŞ

Bilgisayar görmesi alanında nesne tanıma çalışmaları önemli bir yere sahiptir fakat teknolojinin ilerlemesiyle birlikte nesneleri yalnızca tespit etmek yetersiz kalmaktadır. Bulunan nesnenin boyut, mesafe ve derinlik bilgisi gibi detay özelliklerine de ihtiyaç her geçen gün giderek artmaktadır. Bu özellikleri kullanan uygulamalar ise artık insan hayatında amaçlara ulaşılmak için kullanılan araç olmaktan çok daha fazlası olmaktadır. Örneğin, görme engeline sahip bireylere hareketleri esnasında etrafındaki nesneler ve boyutları hakkında sesli yönlendirme desteği sağlanabilmesi ya da otonom araçlardaki mesafe kontrolü yapılması, olası kazaları önlemek adına hayati önem taşımaktadır. Bu özelliklerden biri olup aynı zamanda bu çalışmanın temelini oluşturan derinlik kavramı ise nesnelerin perspektifimizin dışında kalan kısımlarını ölçümlememizi sağlamaktadır. Derinlik kavramı fiziki olarak ifade edildiğinde ise bir nesnenin görünen boy ve en kavramları dışında kalan üçüncü boyuta karşılık gelmektedir. Derinlik kavramını sorun haline getiren şey ise içerdiği belirsizliktir.

Kameralar üç boyutlu dünyayı iki boyuta dönüştürürken projeksiyon gibi bir düzleme yansıtılmakta ve böylelikle görüntüdeki derinlik bilgisi kaybedilmektedir. Bu belirsizliğe rağmen insan her şeyi üç boyutta mükemmel bir şekilde algılayabilmektedir. Sol ve sağ gözden elde edilen görüntüleri karşılaştırarak, beyin orada ne kadar derinlik olduğunu belirlemektedir. Bir derinlik haritası tahmini olarak, iki görünüm arasındaki hafif hareket sonucu oluşan değişiklik kullanılarak hesaplanabilir. Bu şekilde elde edilen görüntü çiftine stereo görüntü denmektedir. Beynin, görüntünün boyutu, yapısı ve hareketi gibi kalıpları tanımlarken elde ettiği görsel bilgiler ile gözlerin değiştirilebilir odak uzunluğu birleştiğinde insanların üç boyutlu mükemmel bir görüşe sahip olması sağlanmaktadır.

Gelişen teknolojide, insanın görüş yapısından faydalanarak derinlik ve mesafe tespiti üzerindeki payını artırmaktadır. Çalışmalarda (Duran ve Kaya, 2018), (Akarun, 2020) bahsedilen donanım cihazlarından bazıları şunlardır; çift kamera (stereo) teknolojisi, ultrasonik sensörler, lidar sistemleri (URL-8), kinect teknolojisi (URL-9). Bu teknolojiler ile her ne kadar doğruya yakın ölçüm yapılsa da işlem güçleri, hesaplama süreleri, menzil sınırlamaları ve maliyetleri gibi sebeplerden ötürü tüketici cihazları için tercih edilmeyebilir ve alternatif bir yol olarak iki boyutlu görüntülerden nesnelerin derinlik tahminini sağlayan yazılımlara ihtiyaç doğabilmektedir.

İnsanın derinlik tahmini üzerine literatürde (Cutting ve Vishton, 1995), (Foley ve Matlin, 2015), (Mertan ve Duff, 2019), mesafe ve derinliği tahmin etmek için kullanılabilecek görsel ipuçları hakkında bazı fikirlerden bahsedilmektedir. İki boyutlu bir görüntüden derinliği tespit etmek için kullanılan bazı ipuçları vardır. İlk ipucu, bir öğenin önünde bulunarak diğerinin görünümünü engellediğinde ortaya çıkan tıkanmadır. Buna dayanarak, arkasındaki nesnenin daha uzakta olduğu düşünülmektedir. Tıkanıklık, derinlik sırası hakkında bilgi sağlar, ancak mesafe hakkında bilgi vermez. Perspektif bir diğer ipucudur. Bunu görmek için, uzaktan yakınsıyor gibi görünen paralel çizgilere bakmak gerekir. Aynı nesne, mesafesine bağlı olarak farklı boyutlara sahip olabilmektedir. Örneğin, iki boyutlu bir görüntüde aynı modelden iki araba olduğunu varsayarsak, uzaktaki araba daha küçük görünecektir. Derinliği anlamak için kullanılan bir başka ipucu da hava perspektifi olarak adlandırılmaktadır. Gökyüzü gibi bizden uzak olan nesneler bulanık bir görüntüye sahip olmaktadır. Ek olarak, derinliği algılarken ışık ve gölge desenleri de kullanılır. Güneş ışınları veya herhangi bir ışık yüzeye yansıdığında, yüzeyin daha parlak görünmesini sağlamaktadır. Gölgelendirmede, bir nesnenin yere yansımasıyla oluşan değişken mesafeler yardımıyla derinlik tahminine katkıda bulunmaktadır. Yükseklik ipucunda ise, yüksek uzunluktaki nesneler görüntülerde daha uzak görünmektedir. (Mertan ve diğ., 2022) tarafından yapılan çalışmada ipuçlarına detaylıca yer verilmektedir. Araştırmacılar tüm bu ipuçlarını ve daha fazlasını öğrenilmiş ön bilgiler olarak işleyip, kullanır ve bu bilgiler doğrultusunda araştırmalara katkıda bulunurlar.

Bu çalışma, iki boyutlu bir görüntüde tespit edilen bir nesnenin derinliğini belirlemek için kullanılan 3D nesne algılama, segmentasyon kullanımı, IPM kullanımı ve referans nesne kullanımı olmak üzere dört yöntemi içermektedir. İki boyutlu görüntülerde derinlik tahmini yapılırken kullanılan bu dört yönteme ait çözüm yaklaşımları bu çalışma ile başlıklar altında toplanarak gruplandırılmakta ve tablo halinde özetlenmektedir. Bu çözüm yaklaşımlarının uygulanması sonucunda karşılaşılan zorluklar ile tespitin gerçekleştirilememesine neden olan engellerden bahsedilmektedir. Yine bu çalışma ile karşılaşılan engeller bir başlık altında toplanarak gelecekteki çalışmaların kolaylaştırılması hedeflenmektedir. Literatüre katkı sağlaması açısından hangi faktörlere dikkat edilmesi gerektiği ve hangilerine öncelik verilmesi gerektiği önceden bilinmektedir. Son olarak bu tezde anlatılan çözüm yöntemleri kullanılarak iki farklı

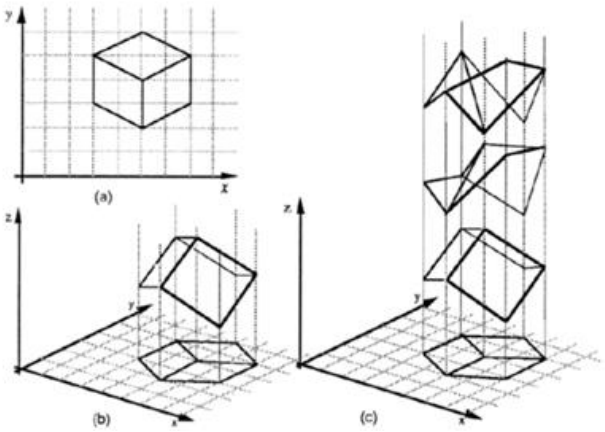
uygulama geliştirilmiştir. Bu uygulamalarda kullanılan mimari, yöntem ve karşılaşılan zorluklar açıklanmakta ve uygulamalardan yapılan çıkarımlara yer verilmektedir.

Tezin ikinci bölümünde, literatürde yer alan iki boyutlu görüntülerde derinlik tahmini için hangi ölçüm tekniklerinin kullanıldığı açıklanmaktadır. Çalışmanın üçüncü bölümünde referans nesne kullanımı ile gerçek zamanlı derinlik tahmini uygulamasının gerçekleştiriminin detaylı anlatımı, mimarisi, malzeme ve yöntem, bulgular ve tartışma sırasıyla yer almaktadır. Çalışmanın dördüncü bölümünde ikinci uygulama olan görüntü işleme ve derin öğrenme ile iki boyutlu görüntüden mesafe tahmini uygulamasının gerçekleştiriminin detaylı anlatımı, mimarisi, malzeme ve yöntem, bulgular ve tartışma sırasıyla yer almaktadır. Bu çalışmanın beşinci bölümünde ise öncelikle literatürde tek bir görüntüden derinlik tahmini yapan çalışmalar açıklanmakta ve alt başlıklar altında toplanarak gruplandırılmakta, daha sonra derinlik ölçümü sırasında karşılaşılan sorunlar ve nedenleri incelenmektedir. Son olarak, Bölüm 6'da bu çalışmadan çıkarılan sonuca ve gelecekteki beklentilere değinilmektedir.

2. İKİ BOYUTLU GÖRÜNTÜLERDE DERİNLİK TAHMİNİ YÖNTEMLERİNE GENEL BAKIŞ

2.1. Temel Kavramlar

Derinlik tahmininin amacı, bir sahnenin uzamsal yapısının bir temsilini elde ederek, nesnelerin üç boyutlu şeklini ve gerçekteki görünümünü geri kazanmaktır. Bu aynı zamanda Szeliski (2011) tarafından da anlatılan, çözüm için yetersiz bilgi verilen ve bazı bilinmeyenleri kurtarmaya çalışılan ters problem olarak da bilinir. 2 boyutlu görünüm ve 3 boyutlu görünüm arasındaki eşleşmenin benzersiz olmadığı, birden fazla nesnenin bir araya gelerek birbirini kapsaması sonucu düzlemde aynı yansımanın elde edilebileceği Şekil 2.1'de görülmektedir. Derinlik tespiti üzerine çalışılmaya başlanıldığından beri burada olduğu gibi bakış açısı dahil birçok problem ile karşılaşılmaktadır.



Şekil 2.1. (a) Bir çizimin x ve y koordinatları üzerindeki noktaları hakkında bilgi sağlar. (b) İnsan görme sistemi buradaki gibi genellikle sadece tek bir 2D projeksiyon verildiğinde bir nesneyi üç boyutlu olarak yeniden oluşturabilir. (c) Birden fazla 3 boyutlu yapı tek bir düzlemsel çizime ait olabilir (Town, 2011)

İki boyutlu görüntü olarak verilen bir görüntüyü kurtarmak ve yeniden oluşturmak için, karşılığı olan her pikselin derinliğinin veya Z bileşeninin bilinmesi gerekmektedir. Derinlik, Şekil 2.2'de sağ resimde gösterildiği gibi, açık ve koyu renkler ile temsil edilmektedir. Daha uzaktaki noktayı ifade eden alanlar daha açık renk gözükmektedir. Derinlik algısını anlamak, birçok bilgisayarla görü uygulamasında kullanılmaktadır. Örneğin; otonom bir araç, diğer araçlar ve nesneler arasındaki mesafenin farkında olduğunda daha iyi karar verebilmektedir. Bunun üzerine Şekil 2.2'ye bakıldığında sol şeritteki iki araba arasındaki mutlak mesafenin ne kadar olduğunu söylemek zordur.

Ayrıca, soldaki ağaçların gerçekte eve çok yakın mı yoksa evden çok uzak mı olduğunu algılamakta zordur. Bu zorluk derinlik tahmin yöntemleri ile çözülebilir.



Şekil 2.2. RGB görüntüden derinlik görüntüsüne dönüşüm

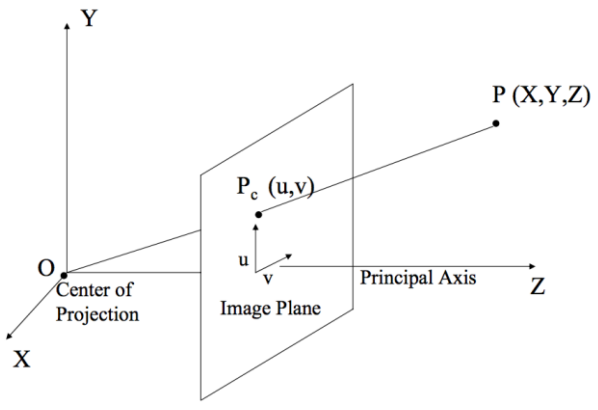
Derinlik tahmini için kullanılan yöntemler literatürde farklı gruplandırmalar şeklinde yer almaktadır. Bunlardan ilk gruplandırma şekli (Sanz ve diğ., 2012), (Khan ve diğ., 2020) çalışmalarında yer alan aktif ve pasif yöntemler olarak ikiye ayrılmaktadır. Aktif yöntemler olarak adlandırılan grup nesnelere olan mesafeyi tahmin etmek için aktif ışık aydınlatmasını kullanarak derinlik tahmini yapar. Ultrason ve time of flight (ToF) (Schuon ve diğ., 2008), aktif yöntemlerin diğer örnekleridir. Pasif yöntemler, elde edilen görüntülerin optik özelliklerinden yararlanır. Bu yöntemler, hesaplamalı görüntü işleme yoluyla derinlik bilgisinin çıkarılmasını amaçlar. Pasif yöntemlerde kendi içinde çoklu görüntülü derinlik tahmini ve monoküler derinlik tahmini olarak ikiye ayrılır. Farklı bir çalışmada (Zhao ve diğ., 2020) ise derinlik tahmin yöntemleri geometrik tabanlı, sensör tabanlı ve derin öğrenme tabanlı olmak üzere üç başlık altında incelenmektedir.

Geometri tabanlı yöntemler: geometrik kısıtlamalara dayalı olarak birkaç görüntüden 3B yapıları kurtarmaya çalışmaktadır. Hareketten gelen yapı (SFM) (Ullman, 1979), bir dizi iki boyutlu görüntüden 3B yapıları tahmin etmek için kullanılan bir yöntemdir ve SLAM' da (Mur-Artal ve diğ., 2015) başarıyla uygulanır. Görüntü dizileri arasındaki nitelik karşılaştırmaları ve geometrik kısıtlamalar SFM tarafından hesaplanır. Derinlik tahmininin doğruluğu nitelik eşleştirmeye ve yüksek kaliteli görüntü dizilerine dayanır. Diğer bir yöntem olan stereo görüntü eşleştirme, iki kamera kalibre edilir ve görüntülerin eşitsizlik haritaları bir maliyet fonksiyonu aracılığıyla hesaplanır. Geometrik tabanlı bu yöntemler görüntü çiftlerine veya görüntü dizilerine bağlı olduğundan tek bir görüntüden derinlik haritası elde etmek zordur.

Sensör tabanlı yöntemler: RGBD kameralar ve Lidar gibi derinlik sensörleri kullanarak karşılık gelen görüntünün derinlik bilgisi doğrudan alınabilir. Otonom araçlarda yaygın olarak kullanılan Lidar, lazer ışığı ile nesnelerin ışığa olan mesafesini tespit etmeye yarayan bir teknolojidir. Lidar sistemleri, ortamdaki nesnelere lazer ışıkları göndermektedir ve bu gönderimin geri yansıması için geçen süreyi (uçuş süresi olarak da bilinir) ölçmektedir. Ek olarak, lazer ışıklarının dalga boyundaki değişiklik ile derinlik bilgisine doğru bir şekilde ulaşılabilmektedir. RGB-D kameralar, RGB görüntüsünün piksel düzeyindeki yoğun derinlik haritasını doğrudan elde etme yeteneğine sahiptir. Bu derinlik sensörlerinin büyük boyutu küçük robotlara uygulanmasını engellerken, güç tüketimi, hesaplama süreleri, menzil sınırlamaları ve maliyetleri gibi sebeplerden ötürü tercih edinimini azaltmakta ve alternatif bir yol olarak iki boyutlu görüntülerden nesnelerin derinlik tahminini sağlayan derin öğrenmeye tabanlı yöntemlere ilgiyi artırmaktadır.

Derin öğrenme tabanlı yöntemler: CNN, RNN gibi yapay sinir ağlarını kullanarak veri giriş türüne göre çoklu veya monoküler görüntünün derinliği tahmin etmektedirler. Ayrıca eğitim türü bakımından da denetimli öğrenme, denetimsiz öğrenme ve yarı denetimli öğrenme ayrıştırılabilmektedir.

Derinlik probleminin zorluğunu azaltmak için kullanılan yöntemlerden biri 3 boyutlu sınırlayıcı kutu tahminidir. Bu yönteme değinmeden önce, Şekil 2.3'de verilen ve temelini oluşturan kamera projeksiyonunun geometrik modelini anlamak gerekir. İğne deliği (pinhole) kamera modeli burada eğrilik olmadan en temel haliyle incelenmektedir.



Şekil 2.3. Kamera koordinat sistemi

Aşağıda verilen Denklem 2.1'de, 3D noktalar görüntü düzlemine (u, v) = f (X, Y, Z) şeklinde eşlenmektedir. (u, v) görüntüdeki piksel koordinatları, (X, Y, Z) görüntü üzerinde izdüşümü olan üç boyutlu noktanın koordinatlarıdır.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (2.1)$$

Bu dönüşümü tanımlayan matematiksel model ' p = K [R| T] * P ' olarak yazılmaktadır.

- 'p' görüntü düzleminde yansıtılan noktadır.
- 'K' kameranın içsel matrisidir. K matrisi, görüntü düzlemine 3D noktaların yansıtılmasından sorumludur. Bunu yapmak için, tanımlanması gereken alanlar vardır. Bunlardan ilki odak uzaklığı olan (f_x, f_y): görüntü düzleminin kamera merkezine göre konumunun ölçülmesidir. İkinci nokta olan (c_x, c_y): optik eksenin ve görüntü düzleminin kesişimidir.
- '[R| T]', dünya çerçevesindeki noktanın kamera çerçevesine göre dönüşümünü tanımlayan dışsal parametrelerdir. 'T' çeviri vektörüdür. 'R' dönme matrisidir.
- 'P ([X, Y, Z, 1]) ', önceden tanımlanmış bir dünya koordinat sisteminde ifade edilen 3D noktayı temsil etmektedir.
- En-boy oranı ölçekleme 's', odak uzaklığı değiştikçe piksellerin x ve y yönünde nasıl ölçeklendiğini kontrol etmektedir (URL-3).

2.2. İki Boyutlu Görüntülerde Derinlik Tahmini Yöntemleri

Bu çalışmada monoküler görüntülerde derinlik tahmininde karşılaşılan zorlukları incelerken kullanılan yöntemler, üç boyutlu sınırlayıcı kutu tahmini, homografi tabanlı IPM, görüntü segmentasyonu ve referans nesne kullanımı şeklinde sınıflandırılmaktadır.

2.2.1. Üç Boyutlu Sınırlayıcı Kutu Tahmini

Derinlik tespitinde sıkça kullanılan yöntemlerden bir olan monoküler 3D nesne algılama (Saxena ve diğ., 2006), iki boyutlu RGB görüntüsündeki aranan nesnelerin etrafına üç boyutlu sınırlayıcı kutu oluşturma işlevidir. Bu işlev, düşük maliyeti ve kullanılabilirliği nedeniyle otonom sürüş endüstrisi için oldukça ilginçtir. Tek bir 2D görüntüyü 3D

görüntüye dönüştürme görevi oldukça zordur ve aracın yönünü tahmin etmek bu önemli göreve doğru atılmış önemli bir adımdır.

Bu yöntemde 3D sınırlayıcı kutunun perspektif projeksiyonunun 2D algılama penceresine sıkıca sığması gerektiği gerçeği kullanılmaktadır. 2D nesne algılayıcının, yansıtılan 3D kutunun sınırlayıcı kutusuna karşılık gelen kutular üretmek üzere eğitildiği varsayılmaktadır. 3D sınırlayıcı kutu, merkezi T =$[t_x, t_y, t_z]^T$, boyutları D = $[d_x, d_y, d_z]$ ve oryantasyonu R (θ, φ, α) parametreleri ile tanımlanmaktadır. Kameranın iç matrisi K verildiğinde, nesnenin koordinat çerçevesindeki $X_0 = [X, Y, Z, 1]^T$, 3D noktasının $\mathrm{x} = [\mathrm{x}, \mathrm{y}, 1]^T$ görüntüsüne izdüşümü Denklem 2.2'deki gibidir.

$$x = K[R \quad T]\, X_0 \tag{2.2}$$

Nesne koordinat çerçevesinin başlangıcının 3D sınırlayıcı kutunun merkezinde olduğu ve nesne boyutlarının (D) bilindiği varsayıldığında, 3D sınırlayıcı kutu köşelerinin koordinatları basitçe Denklem 2.3'deki gibi tanımlanabilmektedir.

$$X_1 = [d_x/2, d_y/2, d_z/2]^T,\ X_2 = [-d_x/2, d_y/2, d_z/2]^T, \ldots,$$
$$X_8 = [-d_x/2, -d_y/2, -d_z/2]^T \tag{2.3}$$

Bu yöntemin detaylı bahsedildiği çalışmada (Mousavian ve diğ., 2017), x_{min} koordinatıyla 2D sınırlama kutusunun sol tarafına temas eden bir $X_0 = [d_x/2, -d_y/2, d_z/2]^T$ köşesinin izdüşümü düşünülürse, bu noktadan noktaya yazışma kısıtlaması Denklem 2.4'deki gibi sonuçlanmaktadır.

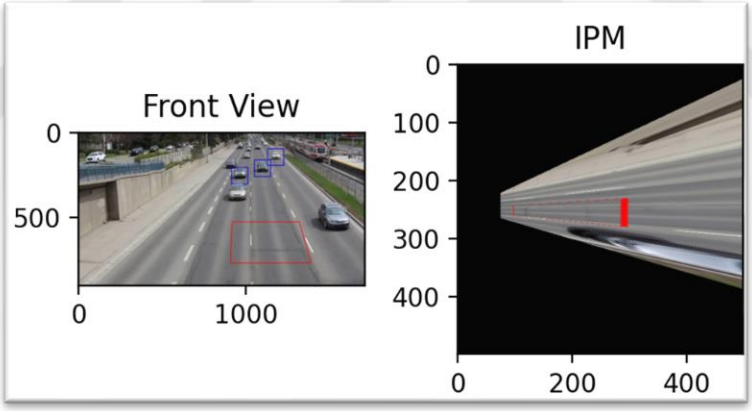
$$x_{min} = \left(K \quad [R \quad T] \quad \begin{bmatrix} d_x/2 \\ -d_y/2 \\ d_z/2 \\ 1 \end{bmatrix} \right)_x \tag{2.4}$$

Kalan 2D kutunun diğer taraf parametreleri olan x_{max} , y_{min}, y_{max} için benzer denklemler türetilebilir. Toplamda 2D sınırlayıcı kutunun kenarları, 3D sınırlayıcı kutu üzerinde dört kısıtlama sağlamaktadır. Noktalar arasındaki mesafe bu dört parametre kullanılarak hesaplanır. Kenarları hesaplanan nesnenin tahmini derinliği, referans nesne kullanımı bölümünde belirtildiği gibi ölçülebilir.

2.2.2. Homografi Tabanlı IPM

Bilgisayarlı görüde homografi, yansıtmalı bir düzlemde uygulandığında onu başka bir düzleme (veya görüntüye) eşleyen bir H dönüşüm matrisidir. Ters perspektif haritalama (IPM), sahnenin öne bakan görüntü düzleminden kuş bakışı bir görüntüsü (BEV) üretilmek istenmektedir. IPM metoduna göre önce dünyanın bir düzlemde düz olduğu varsayılmaktadır. Daha sonra homografi projeksiyonu aracılığıyla belirli bir bakış açısından tüm pikselleri bu düz düzleme eşlenmektedir.

IPM metodu kameranın yakınındaki alanı iyi yansıtırken, uzak özellikler için daha az sayıda piksel temsil edildiğinden, perspektif projeksiyon sırasında sahnenin bulanıklaşması ve bozulması daha belirgin hale gelmektedir. Bu istenmeyen bozulma Şekil 2.4'de kırmızı alan ile belirtilen çerçevede gözlemlenebilmektedir. RGB görüntüye IPM dönüşümü uygulanmasıyla sadece kameraya yakın alan kuş bakışına dönüştürülebilmiştir.



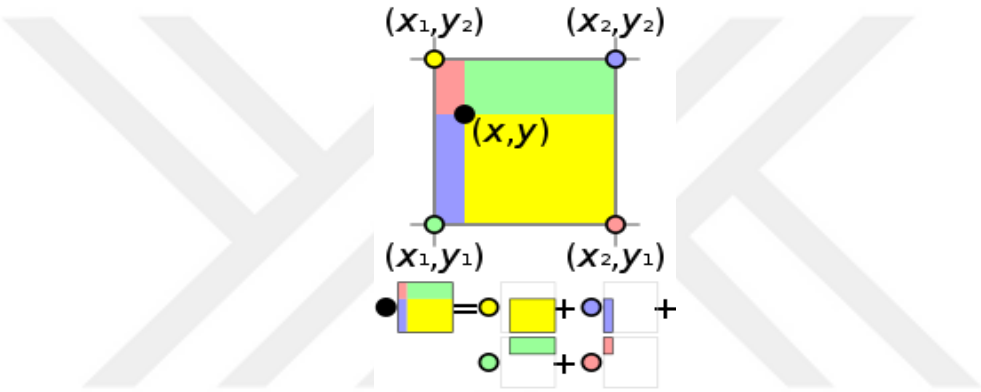
Şekil 2.4. RGB görüntüye IPM dönüşümü

IPM metodunu gerçekleştirebilmek için ilk olarak kamera sabit bir konumda olmalıdır. Yolun konumu kameraya duyarlı olduğundan, konum veya yöndeki hafif bir bozulma, 3D sahnenin bir görüntü düzlemine nasıl yansıtıldığını değiştirecektir. İkinci olarak yüzey düzlemsel olmalıdır. Yüksekliği olan herhangi bir nesne bu koşulu ihlal edecektir. Düzlemsel olmayan yüzeyler kuş bakışı görünüm görüntüsünde bozulma yaratacaktır.

Dönüşüm gerçekleşirken yolun zeminde düz olduğu varsayılmaktadır. Bu nedenle, $Z = 0$ denklemi yolda bulunan her nokta için geçerlidir. Bu da sorunu etkili bir şekilde

düzlemsel homografiye dönüştürmektedir. Bu işlem, görüntü çarpıtma gerçekleştirmek için OpenCV 'de kullanılan yöntemlerden biridir.

Projeksiyon modelinde, sahneyi kameraya göre tanımlamak gerektiğinden bazı özellikler girilerek içsel ve dışsal parametreler oluşturulmaktadır. Son aşama olarak kameradan uzaklaştıkça, perspektifin bir sonucu olarak seyreklik daha gözlemlenebilir hale gelir. Bu nedenle pikselleri yeniden örneklendirmek adına bir yaklaşım olarak bilineer enterpolasyon kullanılabilir. Bu işlem, Şekil 2.5'de de görüldüğü üzere hesaplanacak noktayı (x, y) çevreleyen en yakın 4 köşenin ağırlıklı (mesafeye göre) toplamı alınarak yapılmaktadır.



Şekil 2.5. Bilineer enterpolasyon (URL-1)

2.2.3. Görüntü Segmentasyonu Kullanımı

Nesneleri kapsayan sınırlayıcı kutular bazen yetersiz kalır ve yerine tam olarak nesnelerin sınırlarını tespit eden Görüntü segmentasyonu (bölütlemesi) kullanılır. Görüntü bölütleme, bir görüntünün benzer bölümlerini kendi sınıf etiketleri altında gruplamayı amaçlayan dijital görüntü işlemenin bir alt alanıdır. Kısaca segmentasyon piksellere etiket atamaktır. Görüntü segmentasyonunda kullanılan tekniklerinin başlıca türleri şunlardır:

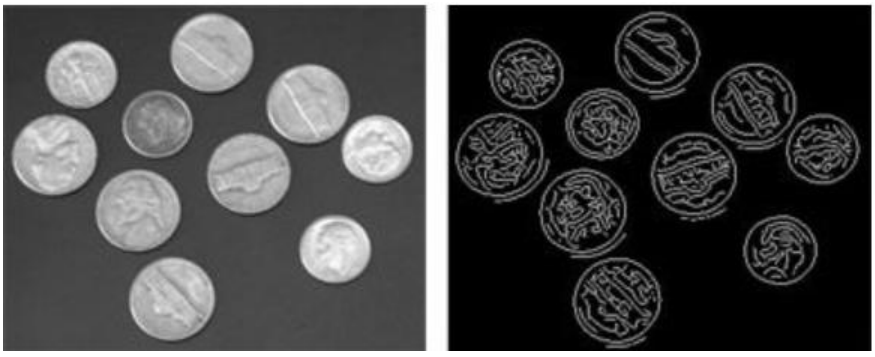
- Eşik Bazlı Segmentasyon
- Kenar Tabanlı Segmentasyon
- Bölge Bazlı Segmentasyon
- Kümeleme Tabanlı Segmentasyon
- Havza Segmentasyonu
- Derin Öğrenmeye Dayalı Segmentasyon

Eşik Bazlı Segmentasyon: Pikselleri belirli bir değere veya yoğunluklarına göre bölen en basit görüntü bölütleme yöntemidir. Diğer nesnelerden veya arka planlardan daha yüksek yoğunluklu nesneleri bölümlere ayırmak için uygundur. Eşik değerinden büyük değerlere sahip pikseller 1, eşik değerinden küçük değerlere sahip pikseller 0 olarak ayarlanır ve Şekil 2.6'daki gibi bir sonuç elde edilir. Eşikleme genellikle görüntü ikilileştirme için kullanılır, böylece yalnızca ikili görüntüler üzerinde çalışan kontur algılama ve tanımlama gibi başka algoritmalar tarafından kullanılabilir.



Şekil 2.6. Eşik bazlı segmentasyon örneği

Kenar Tabanlı Segmentasyon: Belirli bir görüntüdeki çeşitli nesnelerin kenarlarını tanımlayan popüler bir görüntü işleme tekniğidir. Kenar tabanlı segmentasyon algoritmaları, kontrast, doku ve renk değişimlerine dayalı olarak kenarları tanımlar. Tek tek kenarları içeren kenar zincirlerini kullanarak bir görüntüdeki nesnelerin sınırlarını doğru bir şekilde temsil edebilirler. İstenmeyen paraziti gidermek ve kenarları algılamayı kolaylaştırmak için görüntünün önceden işlenmesi gerekir. Şekil 2.7'de örneği görülen Canny ve Prewitt en popüler kenar algılama algoritmalarından bazılarıdır.



Şekil 2.7. Canny yöntemi kullanılarak görüntü segmentasyonu (URL-4)

Bölge Bazlı Segmentasyon: Bir görüntünün benzer özelliklere sahip bölgelere bölünmesini içerir. Her bölge, algoritmanın bir tohum noktası aracılığıyla yerleştirdiği bir piksel grubudur. Algoritma çekirdek noktaları bulduğunda, diğer komşu noktalarla birleştirerek bölgeleri büyütebilir veya eksilterek küçültebilir.

Kümeleme Tabanlı Segmentasyon: Görüntüyü benzer özelliklere sahip piksel kümelerine (ayrık gruplara) böler. Veri öğelerini, bir kümedeki öğelerin diğer kümelerde bulunan öğelere kıyasla daha benzer olduğu kümelere ayırır. Popüler kümeleme algoritmalarından en bilinenleri fuzzy c-means (FCM) ve k-means algoritmalarıdır. Görüntü segmentasyonunda, oldukça basit ve verimli olduğu için çoğunlukla Şekil 2.8'de görülen k-means kümeleme algoritması kullanılmaktadır.



Şekil 2.8. K-means algoritması ile görüntü segmentasyonu (URL-5)

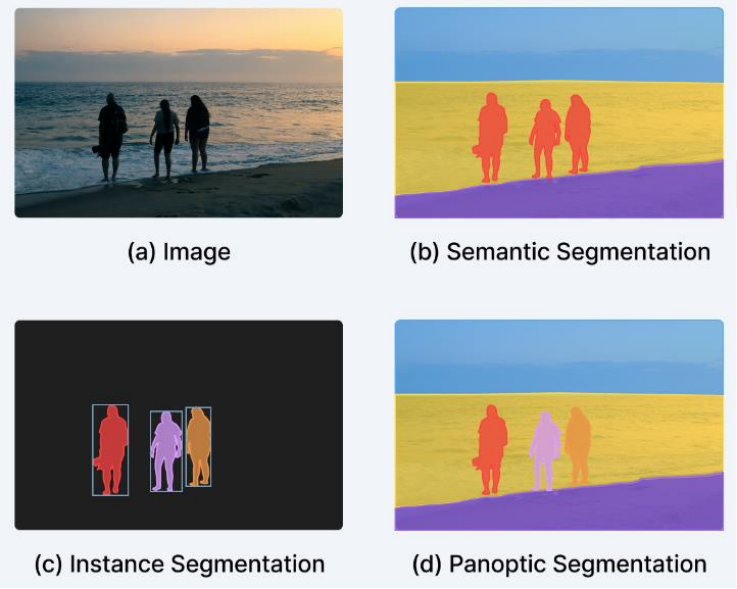
Havza (Watershed) Segmentasyonu: gri tonlamalı bir görüntüye uygulanan dönüşümlerdir. Havza segmentasyon algoritmaları, pikselin parlaklığını yüksekliği olarak kabul eder ve topografik haritalar gibi görüntüleri ele alır. Aynı gri değere sahip pikselleri gruplandırarak, görüntüleri piksel yüksekliğine göre birden çok bölgeye böler.

Derin Öğrenmeye Dayalı Segmentasyon: Bir görüntüyü analiz etmek ve yüzler, nesneler vb. gibi farklı bileşenlerini tanımlamak için kullanılmaktadır. Evrişimli Sinir Ağları (CNN) yukarıda bahsedilen 5 klasik segmentasyon tekniğine göre görüntü verilerini çok hızlı ve daha doğru bir şekilde tanımlayıp işleyebildikleri için kullanılmaktadır.

2.2.3.1 Görüntü Segmentasyonu Türleri

Segmentasyon türleri temelde üçe ayrılır. Bunlardan ilki olan semantik segmentasyon, bir görüntüde bulunan nesneleri piksel düzeyinde algılar ve tür bazında sınıflandırmaktadır.

Yaygın olarak kullanılan semantik segmentasyon teknikleri arasında U-Net, SegNet bulunur. İkinci tür olan örnek (Instance) segmentasyonda, her piksele ait olduğu sınıf değeri atanır ve sınırlayıcı kutu içine alınır fakat semantik segmentasyondan farklı olarak aynı sınıftaki iki farklı nesne birbirlerinden farklı renklere sahip olmaktadır. Yaygın olarak kullanılan örnek segmentasyon teknikleri ise Faster R-CNN, Mask R-CNN ve YOLACT' tır. Kirillov ve arkadaşları (2019) tarafından geliştirilen segmentasyon olan panoptik bölütleme, görüntüdeki bir nesnenin her bir örneğinin birbirinden ayrıldığı ve nesnenin kimliğinin tahmin edildiği, anlamsal bölütleme ve örnek bölütlemenin birleşimi olarak ifade edilebilmektedir. Panoptik segmentasyon modellerinin çoğu Mask R-CNN yöntemini temel almaktadır. Yaygın olarak kullanılan mimarileri arasında UPSNet, FPSNet ve EPSNet bulunur. Şekil 2.9'da bu üç tür segmentasyonun sonucu görülmektedir.



Şekil 2.9. Görüntü segmentasyon türleri (URL-6)

Bir girdi görüntüsü verildiğinde, ilk önce kullanıma hazır nesne detektörlerini kullanarak bir dizi aday nesne örneği oluşturulmaktadır. Görüntüdeki her pikselin derinliğini tahmin etmek için bir derinlik tahmin algoritması uygulanır. Derinlik tahmininin sonucu, aday örneklerin kameraya olan uzaklıklarına göre derinlik sıralamasını oluşturmak için kullanılır. Daha sonra, her aday maskedeki piksellerin piksel bazında etiketlenmesini gerçekleştirmek için kategoriye özel bir segmentasyon ağı uygulanır. Son olarak, aday

örneklerden oluşturulan segmentasyon maskeleri, derinlik sırasına göre bir çıktı görüntüsüne yerleştirilir.

Segmentasyon yönteminin derinlik tahminine katkısı ve birlikte kullanımı anlamak için Valdez-Rodriguez ve arkadaşlarının (2022) çalışmasına baktığımızda ilk olarak, SSegDep-Net modeli, anlamsal bölümleme ve derinlik tahmini elde edebilen iki modülden oluşur. İlk bölüm olan semantik bölütleme modülü U-Net (Ronneberger ve diğ., 2015) modeli kullanılarak elde edilir. Daha sonrasında tek bir görüntüden CNN'leri ve ek giriş kanalları olarak eklenen semantik bölümlendirme modülünden beslenerek derinlik tahmin modülü 3D CNN-UP modelini kullanarak oluşturulur. Sonuç olarak segmentasyonun sağladığı ek bilgilerle ortamdan nesneler ayırılır ve böylece sahnedeki nesnelerin şeklini ve konumunu bilen bir CNN, derinliklerini tahmin edebilir.



Şekil 2.10. Orijinal Görüntü- Nesne Tespiti- Derinlik Tahmini- Semantik Segmentasyon- Örnek Segmentasyon (Ye ve diğ., 2017)

Şekil 2.10'da PASCAL VOC 2012 doğrulama veri setine ait bazı nitel örneklere yer verilmektedir. Her giriş görüntüsü için nesne algılama, derinlik tahmini, anlamsal bölümleme ve nesne örneği bölümlemenin sonuçları gösterilmektedir. Böylelikle görüntüde üst üste gelerek kesişen nesnelerin piksellerinden kaynaklanan belirsizliğin çözülmesi örnek segmentasyonu ve derinlik tahmin bilgisinin birlikte kullanılmasıyla sağlanabilmektedir.

2.2.4. Referans Nesne Kullanımı

Bu yöntemde ilk olarak nesnenin görüntü üzerinde tespit edilmesi ve arka plandan nesnenin ayrılması gerekmektedir. Sonrasında görüntüdeki nesnenin boyutunu belirlemek için referans bir nesne kullanarak bir kalibrasyon yapılması gerekmektedir. Bu referans nesne görüntünün örneğin sol üst köşesi gibi bir noktasına yerleştirilmektedir.

Referans nesne ayırt edici bir renk veya şekle sahip olmalı ve diğer nesnelerden benzersiz olmalıdır. Örneğin OpenCV ile birlikte kullanılan Aruco işaretleyicisi 5x5 cm standart referans nesnelerdendir. Kameradan nesneye olan mesafeyi ölçmek için, belirli sayıda inç, metre vb. piksel sayısını tanımlayan “piksel / metrik” oranını belirlenmesi gerekmektedir. Bu oran kullanılarak nesnenin genişlik ve yükseklik boyutları bulunmaktadır. Bu yöntem tek başına derinliği bulmak için yetersiz kalsa da üç boyutlu kutulama yöntemi gibi farklı yöntemlerle birlikte kullanılarak bir çözüme ulaşmak için denenebilir.

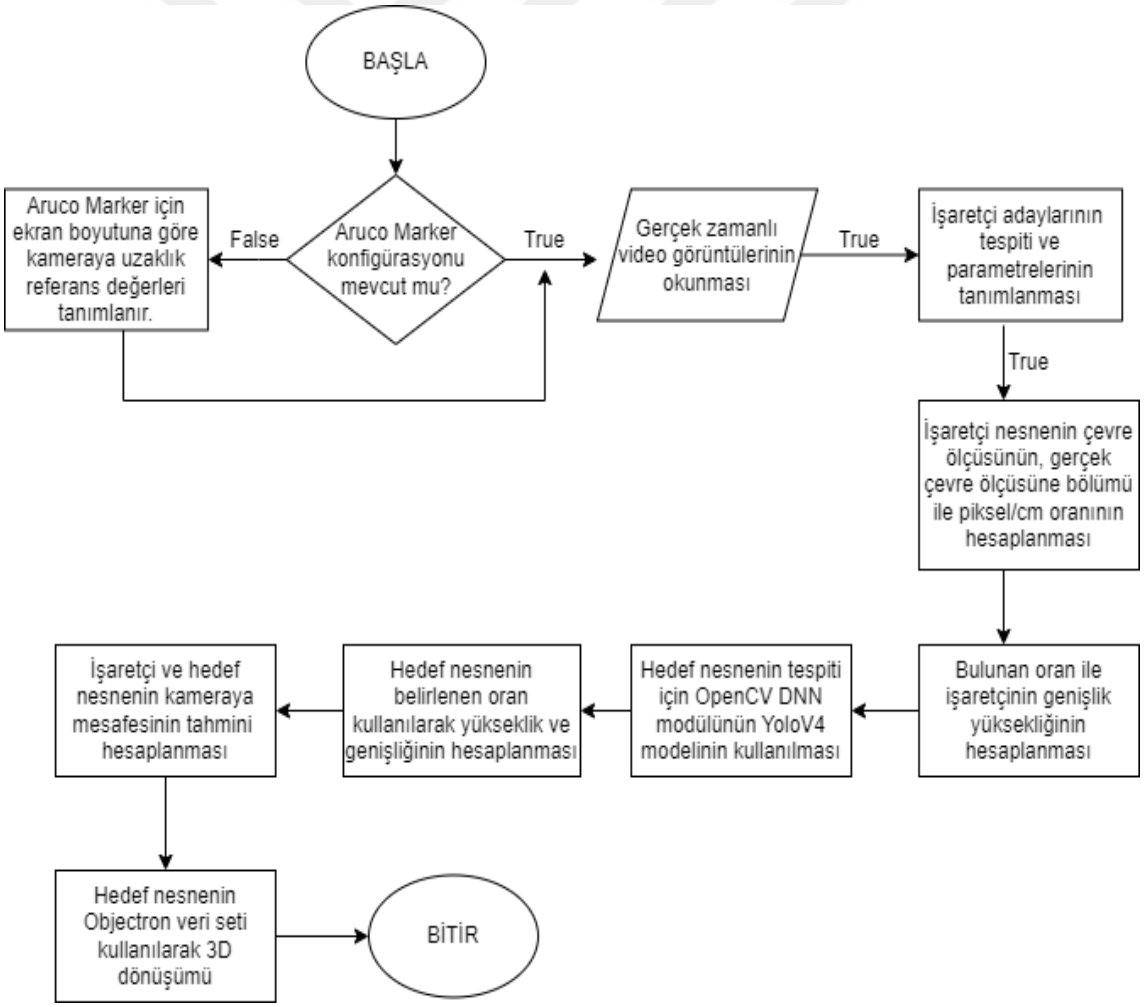
3. REFERANS NESNE KULLANIMI İLE GERÇEK ZAMANLI DERİNLİK TAHMİNİ UYGULAMASI

3.1. Mimari

Referans nesne kullanımı ile gerçek zamanlı derinlik tahmini uygulamasının mimarisi akış seması ve use case diyagramı olmak üzere iki diyagram kullanılarak açıklanmaktadır. Ayrıca uygulamanın kod yapısına Ek-A bölümünden açıklamalı bir şekilde erişilebilmektedir.

3.1.1. Akış Şeması

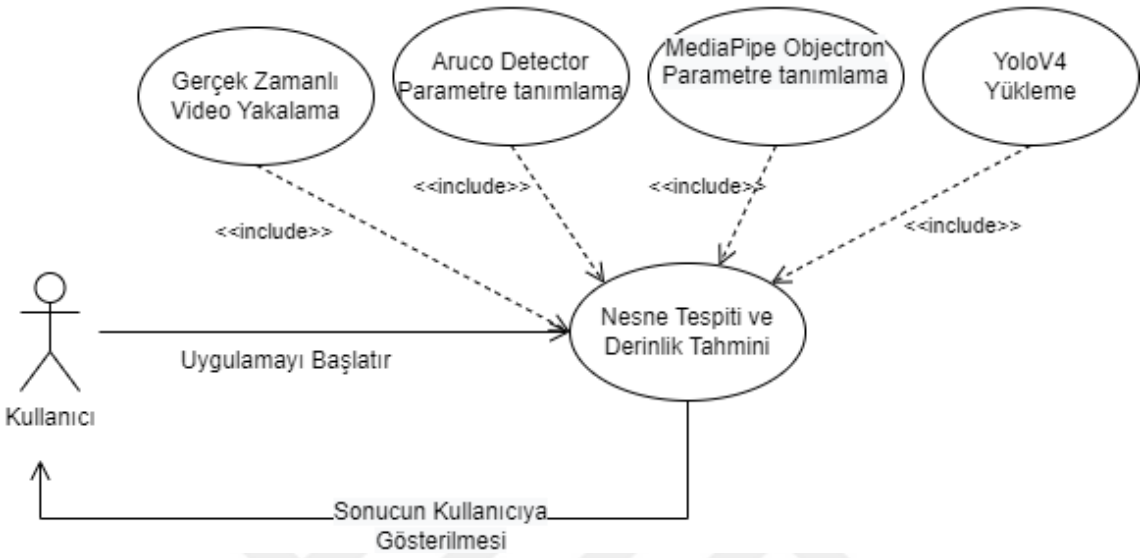
Referans nesne kullanımı ile gerçek zamanlı derinlik tahmini uygulamasına ait akış şeması Şekil 3.1'de gösterilmektedir.



Şekil 3.1.Referans nesne ile gerçek zamanlı derinlik tahmini akış şeması

3.1.2. Use Case Diyagramı

Referans nesne kullanımı ile gerçek zamanlı derinlik tahmini uygulamasına ait use case diyagramı ile kullanıcı ve uygulama arasındaki ilişki Şekil 3.2'de gösterilmektedir.

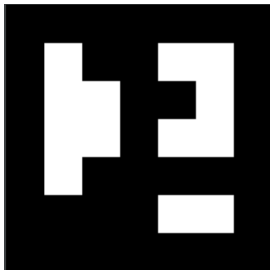


Şekil 3.2. Referans nesne kullanımı ile gerçek zamanlı derinlik tahmini use case diyagramı

3.2. Malzeme ve Yöntem

Derinlik tahminindeki karmaşıklığı çözmek için kullanılan yöntemler Bölüm 2'deki başlıkta anlatılmaktadır. Bu yöntemlerden referans nesnesi kullanımı ve üç boyutlu sınırlayıcı kutu yöntemi derin öğrenme ile birlikte kullanılarak bu uygulama gerçekleştirilmiştir. Uygulama Windows 10 işletim sistemine sahip bir bilgisayarda Pycharm geliştirme ortamı kullanılarak Python dilinde CPU kullanılarak yazılmıştır.

Uygulama ilk olarak gerçek zamanlı görüntüde referans nesnesinin tanımlanması ve tespit edilmesi ile başlamaktadır. Referans nesnesi olarak Rafael Munoz ve Sergio Garrido tarafından geliştirilen kare referans işaretlerinin tespiti için OpenCV' de popüler bir kitaplık olan Aruco kullanılmıştır. Aruco Marker, geniş siyah kenarlıklı ve tanımlanmasını (id) belirleyen bir iç ikili matristen oluşan sentetik bir kare işaretçidir. Siyah kenarlık, görüntüdeki hızlı tespitini kolaylaştırır ve ikili kodlama, hata tespit ve düzeltme tekniklerinin tanımlanmasına ve uygulanmasına izin verir. İşaret boyutu, matrisin boyutunu belirler. Örneğin 4x4'lük bir işaretleyici boyutu 16 bitten oluşur. Uygulamada Şekil 3.3'de görülen 5x5'lik işaretleyici kullanılmaktadır.



Şekil 3.3. Aruco Marker 5x5

İşaretçi tespit süreci iki adımdan oluşmaktadır. İlk aşama işaretleyici adaylarının tespitidir. Bu adımda, işaretçi olmaya aday olan kare şekilleri bulmak için görüntü analiz edilir. İşaretçileri segmentlere ayırmak için adaptive thresholding(eşikleme) yapılır, daha sonra eşiklenmiş binary görüntüden konturlar(kenar) çıkarılır ve kare şekline yakın olmayanlar atılır. Bazı ekstra filtrelemeler de uygulanır (çok küçük veya büyük konturların kaldırılması, birbirine çok yakın konturların kaldırılması vb.).

İkinci olarak tespit edilen adayların, iç kodlamalarını analiz ederek bunların gerçekten işaretçi olup olmadıklarını belirlemek gerekir. Bunu yapmak için perspektif dönüşüm uygulanır, işaretçi kanonik (canonical) denilen kare halini alır. Ardından, beyaz ve siyah bitleri ayırmak için görüntüye Otsu eşikleme uygulanır.

Görüntü, işaretleyici boyutuna ve kenarlık boyutuna göre farklı hücrelere bölünür. Daha sonra, beyaz mı yoksa siyah mı olduğunu belirlemek için her hücredeki siyah veya beyaz piksel sayısı sayılır. Son olarak, işaretçinin belirli bir sözlüğe ait olup olmadığını belirlemek için bitler analiz edilir (URL-7). Ek-A'daki 1 numaralı kod parçası uygulamadaki Aruco tanımlanmasına aittir.

Aruco tespit ve tanımlamasından sonra bu Aruco nesnesinin bilinen ölçülerini kullanarak, uygulamada derinliğinin bulması hedeflenen kupa nesnesinin ölçülerinin kaç cm olduğunu ölçümlemek amacıyla Ek-A'daki 2 numaralı kod parçası kullanılır. Bu kod parçasında Aruco nesnesinin görüntüdeki piksel çevre ölçümünün gerçek çevre ölçüsü olan 20 cm bölümü ile elde edilen oran yer almaktadır. Daha sonrasında iki nokta arası uzaklık formülü ile hesaplanan w(genişlik) ve h(yükseklik) değerleri bu orana bölünerek Aruco nesnesinin yükseklik ve genişliği Şekil 3.4'deki gibi ekrana yazdırılır.



Şekil 3.4. Aruco Marker üzerinde piksel/cm oranı uygulaması

Bu oranın belirlenen hedef nesne olan kupaya uygulanması için önce görüntüde kupanın tespitinin yapılması gerekmektedir. Uygulamada kupanın tespiti için OpenCV DNN (Deep neural network) modülü ile desteklediği Darknet çerçevesinin YOLOv4 modeli kullanılmaktadır. Kullanılan DNN modülünün içeriği 3 dosyadan oluşmaktadır. Darknet modellerini yüklemek için ".weights" uzantısına sahip bir model ağırlıkları dosyası kullanılır. Darknet modelleri için ağ yapılandırma dosyası olarak ".cfg" dosyası kullanılır. Birde nesnelerin isimlerinin yer aldığı bir .txt dosyası kullanılmaktadır.

Uygulamada OpenCV 'de bir DNN modelinin tercih edilmesinin sebebi çıkarım yapmayı oldukça kolaylaştırmasıdır. Eski bir nesne algılama modeli ile derin öğrenme çıkarımı gerçekleştirmek için birden fazla kitaplık yüklemek gerekirken OpenCV DNN modülü çok az bağımlılığa sahip olduğu için kolaylık sağlamaktadır. Ayrıca OpenCV DNN ile koda çok az değişiklik yaparak modeli en son modelle değiştirerek güncelleyebilmektedir. Son olarak, OpenCV, CPU uygulaması Intel işlemciler için yüksek düzeyde optimize edilmiştir, buda tercih için başka bir neden olabilmektedir.

Görüntüde yalnızca kupa nesnesinin tespitini sağlayarak diğer nesneleri eleyen ve Aruco işaretçisinden elde piksel/cm oranı ile kupanın genişlik, uzunluklarını bulmayı sağlayan kod parçacığı Ek-A'daki 3 numarada yer almaktadır.

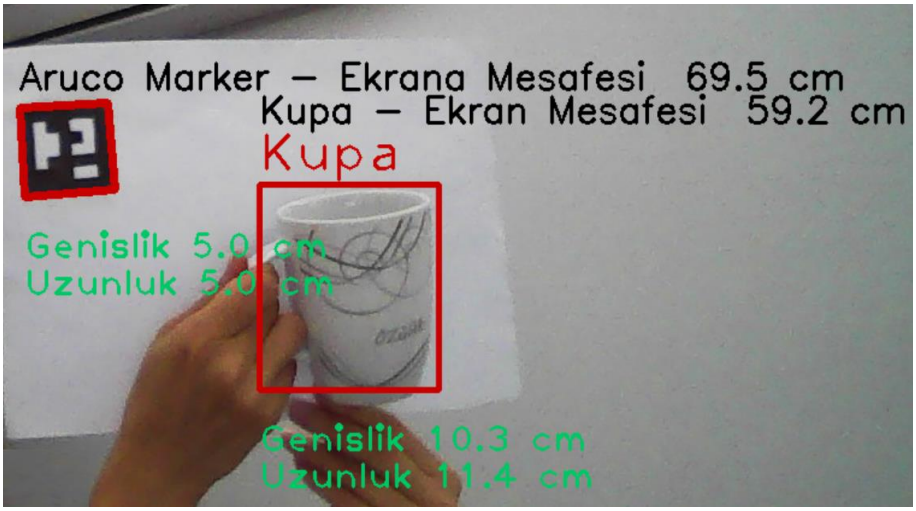
Uygulamada kupanın ölçüleri de tespit edildikten sonra derinlik tahmini için yine Aruco işaretçisine başvurulmaktadır. Gerçek zamanlı video görüntülerinde kupa nesnesinin kameradan ne kadar derinlikte olduğunu saptayabilmek için Aruco işaretçisi kullanılır. Aruco işaretçisinin 5 cm olan kenar uzunluklarından birini seçerek yine iki nokta arası uzaklık ile hareket ettiğinde değişen piksel değeri ve o anki gerçek uzaklığı bulunarak aşağıdaki kodda görüldüğü gibi birer diziye atanır. Bu değerlerle oluşturulan polinom

aracılığıyla ekrana olan uzaklık distanceCM olarak Ek-A'daki 4 numaralı kod parçacığı aracılığıyla bulunur ve Şekil 3.5'de işlem sonucu görülmektedir.



Şekil 3.5. Aruco Marker Ekran Uzaklığı Tespiti

Görüntülerde görünen kupa nesnesinin derinlik değerini hesaplamak için izlenen yol şu şekildedir. Kupa nesnesinin bulunan genişlik değeri aynı zamanda kupanın çapı olduğundan dolayı Aruco işaretçisinin bulunan uzaklık değerinden çıkartıldığında yaklaşık olarak bir tahmin sonucu üretmektedir. Bu sonuç bu çalışma için özeldir çünkü kupa gerçek zamanlı videoda elde tutulduğu için kulpu nesne tanımlamada Şekil 3.6'da görüldüğü üzere dahil edilmemiştir.



Şekil 3.6. Kupa ekran uzaklığı tespiti

Uygulamada tespit edilen kupa nesnesine derinlik efekti kazandırmak için son zamanlarda Google tarafından geliştirilmiş olan MediaPipe Objectron teknolojisi kullanılmaktadır. MediaPipe Objectron, nesneler için gerçek zamanlı üç boyutlu nesne algılama çözümüdür. İki boyutlu görüntülerdeki nesneleri algılar ve Objectron veri kümesinde eğitilmiş bir makine öğrenimi modeli aracılığıyla pozlarını tahmin etmektedir.

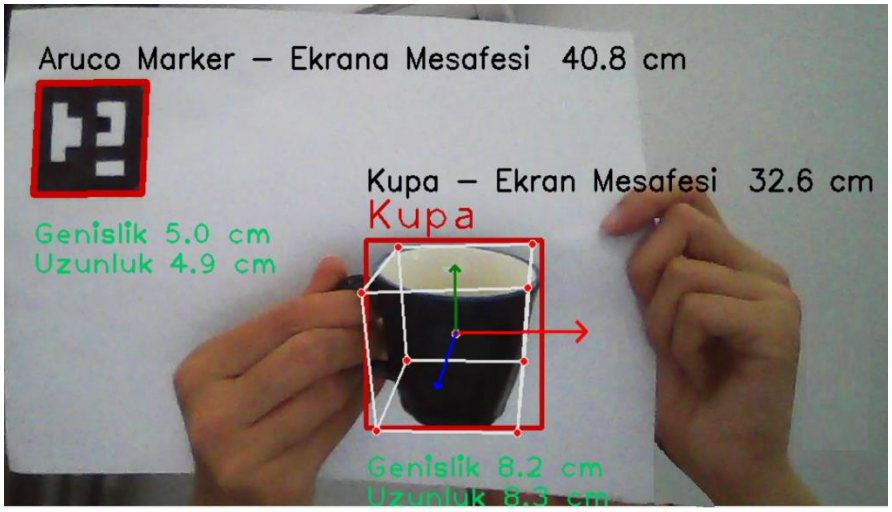
Objectron veri kümesi, kamera pozlarını, seyrek nokta bulutlarını ve çevreleyen ortamdaki düzlemsel yüzeylerin karakterini içeren AR (Arttırılmış Gerçeklik) verilerinin eşlik ettiği, nesne merkezli video kliplerin bir bütünüdür. Kamera çekilen her videoda, nesnenin etrafında hareket ederek onu farklı açılardan yakalamaktadır. Veriler ayrıca, her nesne için, nesnenin konumunu, yönünü ve boyutlarını açıklayan, manuel olarak açıklamalı 3D sınırlayıcı kutular içermektedir. Veriler, Google Cloud depolama alanındaki Objectron paketinde depolanmaktadır.

Objectron veri seti, 4 milyondan fazla görüntüyle desteklenen 15 bin açıklamalı video klipten oluşmaktadır. Veri setindeki kategoriler Tablo 3.1'de görüldüğü üzere: bisikletler, kitaplar, şişeler, kameralar, sandalyeler, bardaklar, dizüstü bilgisayarlar ve ayakkabılar şeklindedir. Coğrafi çeşitliliği sağlamak için veri seti beş kıtada 10 ülkeden toplanmaktadır.

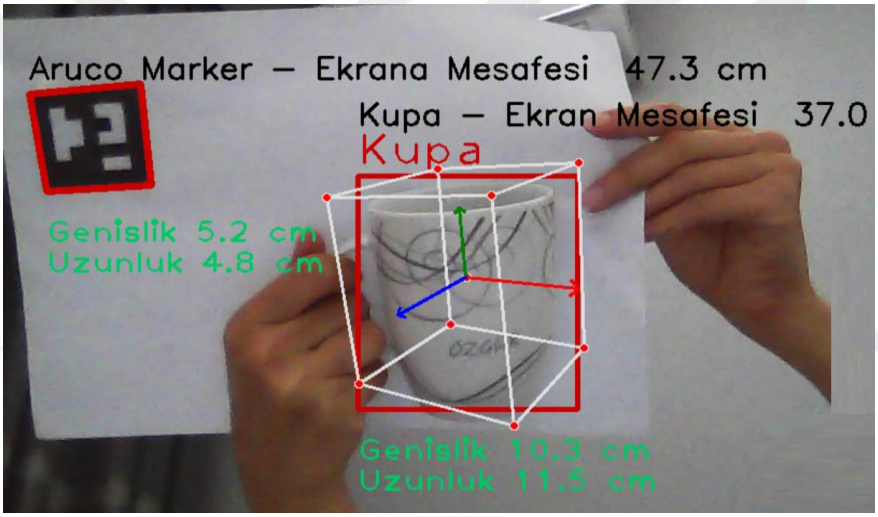
Tablo 3.1. Objectron veri seti içeriği ve dağılımı (Ahmadyan, 2021)

Sınıf	**Bisiklet**	**Kitap**	**Şişe**	**Kamera**	**Kutu**	**Sandalye**	**Kupa**	**Laptop**	**Ayakkabı**
Video	476	2024	1928	815	1609	1943	2204	1473	2116
Frame	150k	576k	476k	233k	396k	488k	546k	485k	557k

Bu veri setini kullanarak ayakkabı, sandalye, kupa ve kamera olmak üzere dört nesne kategorisi için bir 3D nesne algılama çözümü Google tarafından paylaşılmaktadır. Bu çalışmada kupa için 3D nesne algılaması uyguladığında Şekil 3.7 ve Şekil 3.8'deki gibi sonuçlar elde edilmektedir.



Şekil 3.7. Kahve fincanı ile 3 boyutlu derinlik gösterimi



Şekil 3.8. Kupa ile 3 boyutlu derinlik gösterimi

3.3. Bulgular ve Tartışma

Bu çalışmada gerçek zamanlı videodan elde edilen test verilerine göre Aruco işaretçisinin ölçüleri en düşük 4,8 cm ve en yüksek 5,2 cm olmak üzere 0,2 cm artma ve azalma hata payı tespit edilmiştir. Kupalar üzerindeki yanılma payı ise kupaların gerçek ölçüleri olan genişlik (kulp hariç) ve uzunluk ile çalışmadaki elde edilen ortalama genişlik ve uzunluk kullanılarak bulunur. Çalışmadaki değerlere Tablo 3.2'de yer verilmektedir. Aruco işaretçisine oranla nesnelerin boyutlarındaki hata oranının fazla çıkmasının nedeni ise bu nesnelerin Aruco işaretçisinden farklı olarak derinliğe sahip olmalarıdır. Böylelikle Aruco işaretçisine göre ekrana daha yakın gözüktüklerinden kapladıkları alan fazla çıkmaktadır.



Şekil 3.9.Uygulamada kullanılan kupa çeşitleri

Şekil 3.9'da uygulamada kullanılan kupa çeşitleri görülmektedir. Görüntüde kupaların numaralandırılması yer almaktadır ve bu numaralandırma Tablo 3.2 ve Tablo 3.3'de kullanılmaktadır.

Tablo 3.2. Uygulamada kullanılan kupa çeşitlerine göre bulunan sonuçlar

Kupa No	K.G.U	K.G.G	K.B.U	K.B.G	A.G.U	A.G.G	A.B.U	A.B.G	G.E.M	T.E.M
Kupa-1	5,5 cm	5,8 cm	7,3 cm	7,4 cm	5 cm	5 cm	5,1 cm	4,9 cm	70,5 cm	71,1 cm
Kupa-2	6 cm	6,5 cm	6,9cm	7,5 cm	5 cm	5 cm	4,9 cm	5 cm	68,2 cm	69,4 cm
Kupa-3	8,5 cm	8 cm	8,8 cm	8,2 cm	5 cm	5 cm	4,9 cm	5 cm	57,7 cm	58,2 cm
Kupa-4	10 cm	8 cm	12,1 cm	10,4 cm	5 cm	5 cm	5 cm	5 cm	82 cm	81,9 cm
Kupa-5	13 cm	8,5 cm	15,8 cm	13,2 cm	5 cm	5 cm	4,9 cm	5 cm	64,9 cm	66 cm

* K.G.U: Kupa Gerçek Uzunluk, K.G.G: Kupa Gerçek Genişlik, K.B.U: Kupa Bulunan Uzunluk, K.B.G: Kupa Bulunan Genişlik, A.G.U: Aruco Gerçek Uzunluk, A.G.G: Aruco Gerçek Genişlik, A.B.U: Aruco Bulunan Uzunluk, A.B.G: Aruco Bulunan Genişlik, T.E.M: Tahmini Ekran Mesafesi, G.E.M: Gerçek Ekran Mesafesi

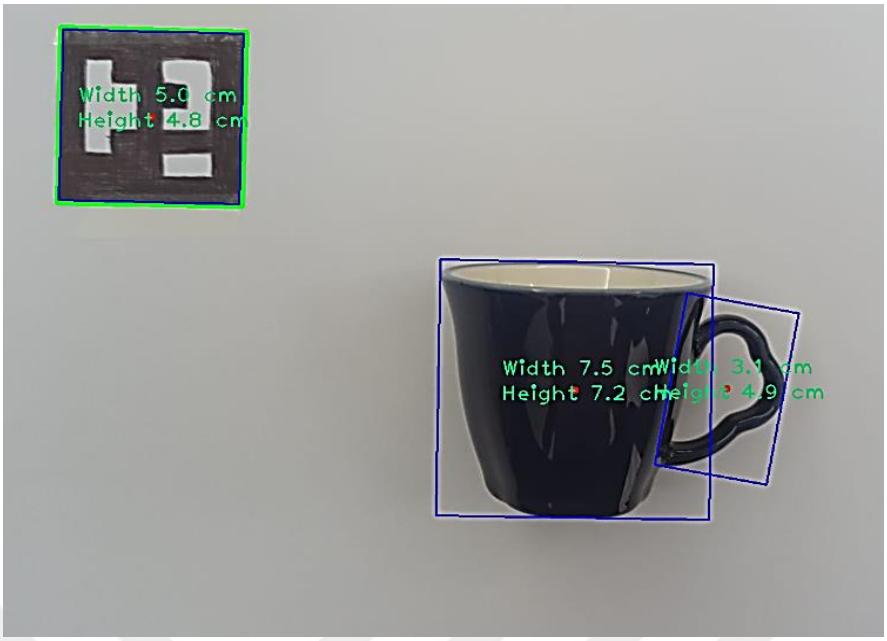
Bu çalışma esnasında kupaların cinsine göre karşılaştıkları birtakım zorluklar olmuştur. Örneğin gece çekimlerinde ışığın kupaların yüzeyinde oluşturduğu parlama dolayısıyla hata oranı artış göstermiştir. Yine bu çalışmada birçok kupa modeli denenmiş ve şeffaf renkte olanları tespit ederken nesneyi bulamama hatası alınmıştır. Bu ve benzeri hatalar Tablo 3.3'de belirtilmiştir.

Tablo 3.3. Referans nesne ile gerçek zamanlı derinlik tahmininde karşılaşılan zorluklar

Karşılaşılan Zorluklar	Kupa 1	Kupa 2	Kupa 3	Kupa 4	Kupa 5
Kamera pozlamasındaki değişiklikler	X	X	X	X	X
Aydıtlamadan etkilenen şeffaf nesne			X		X
Nesne çıkıntıları(kulp)	X	X	X	X	
Aruco nesnesinin derinliği olmadığı için kupalarla mesafe farkı oluşması	X	X	X	X	X

Görüntülerde görünen kupa nesnesinin kameraya mesafesini tespit ederken bu çalışmada kulpunun neden dahil edilmediği bir önceki bölümde açıklanmıştır. Çekim esnasında kupa elde tutulmasaydı bu sefer kupanın bulunan genişliğinden kulpun genişliği çıkartılarak çap elde edilmesi gerekmektedir.

Elde edilen çap Aruco işaretçisinin kameraya uzaklığından çıkartılarak kupanın kameraya uzaklığı için kullanılmaktadır. Kulpların genişlikleri ise genelde 2,5-3 cm arasında değişmektedir. Bu sabit bir değer olarak alınıp hesaplanabilir ancak hata payını daha da düşürmek için Şekil 3.10'da görüldüğü gibi bir kenar algoritması kullanılarak kulpta ayrı şekilde hesaplanabilir.



Şekil 3.10. Kupa kulpu uzunluk tespiti

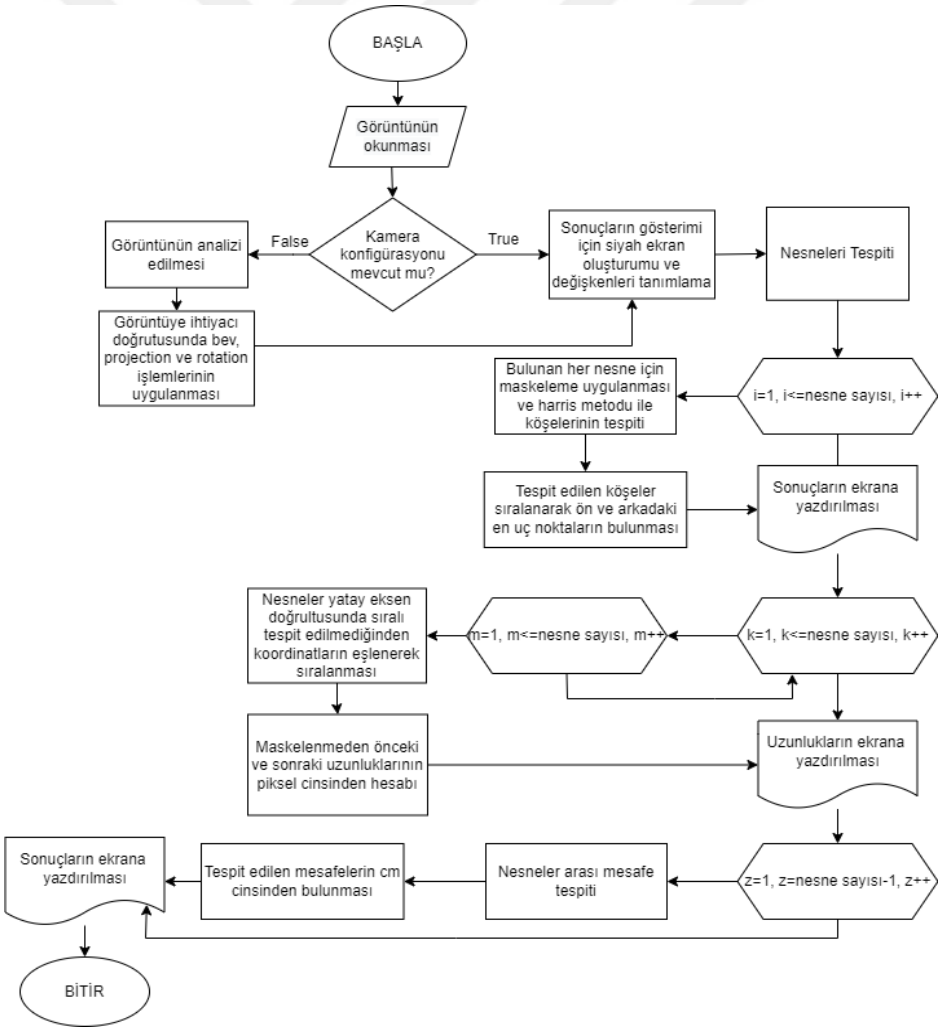
4. GÖRÜNTÜ İŞLEME VE DERİN ÖĞRENME İLE İKİ BOYUTLU GÖRÜNTÜDEN MESAFE TAHMİNİ

4.1. Mimari

Görüntü işleme ve derin öğrenme ile iki boyutlu görüntüden mesafe tahmini uygulamasının mimarisi akış seması, sınıf ve use case diyagramı olmak üzere üç diyagram kullanılarak açıklanmaktadır. Ayrıca uygulamanın kod yapısına Ek-B bölümünden açıklamalı bir şekilde erişilebilmektedir.

4.1.1. Akış Şeması

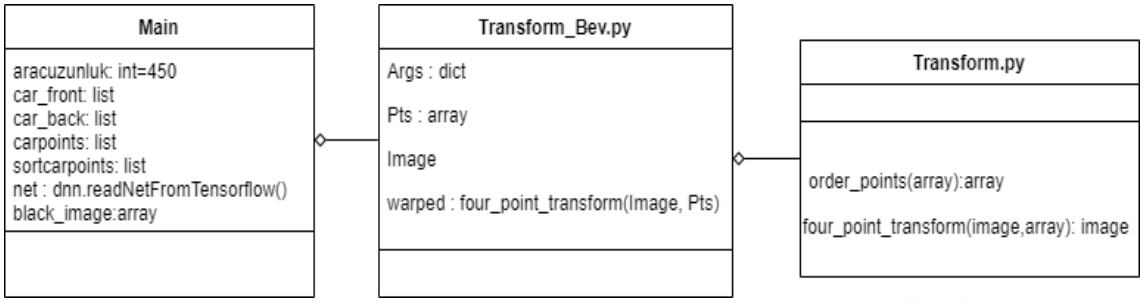
Görüntü işleme ve derin öğrenme ile iki boyutlu görüntüden mesafe tahmini uygulamasına ait akış şeması Şekil 4.1'de gösterilmektedir.



Şekil 4.1. İki boyutlu görüntüden mesafe tahmini uygulaması akış diyagramı

4.1.2. Sınıf Diyagramı

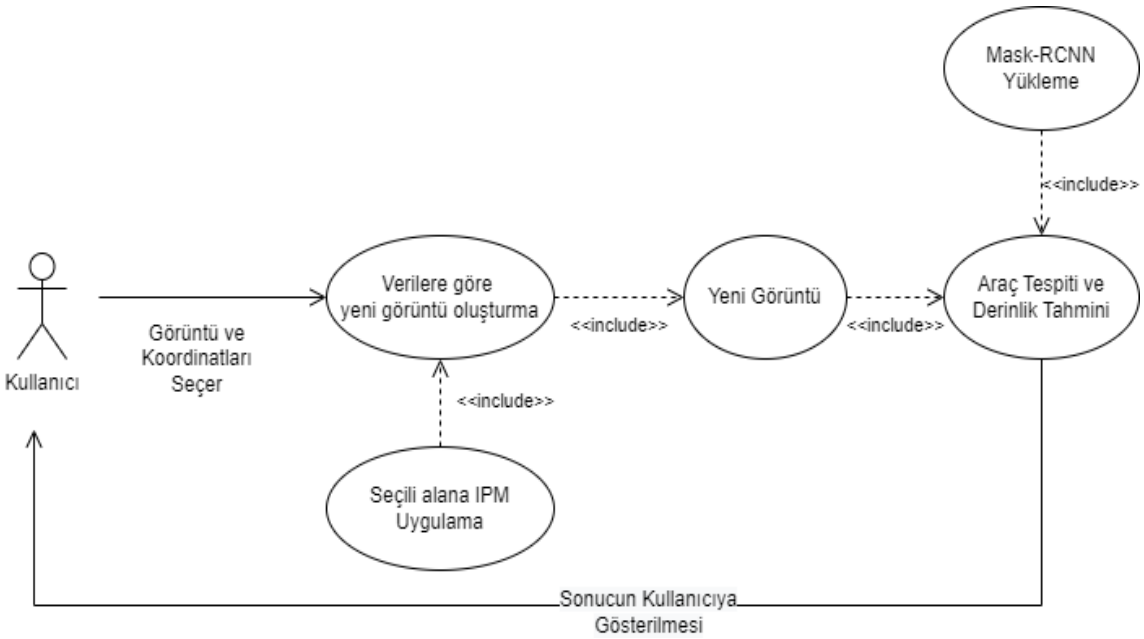
Görüntü işleme ve derin öğrenme ile iki boyutlu görüntüden mesafe tahmini uygulamasına ait üç sınıfın (Main, Transform_Bev, Transform) değişken ve yöntemleri Şekil 4.2'de gösterilmektedir.



Şekil 4.2. Görüntü işleme ve derin öğrenme ile iki boyutlu görüntüden mesafe tahmini sınıf diyagramı

4.1.3. Use Case Diyagramı

Görüntü işleme ve derin öğrenme ile iki boyutlu görüntüden mesafe tahmini uygulamasına ait use case diyagramı ile kullanıcı ve uygulama arasındaki ilişki Şekil 4.3'de gösterilmektedir.



Şekil 4.3. Görüntü işleme ve derin öğrenme ile iki boyutlu görüntüden mesafe tahmini use case diyagramı

4.2. Malzeme ve Yöntem

Uygulamanın amacı Mobese kameralarından alınmış sokak görüntülerindeki park etmiş araçlar arasındaki mesafeyi hesaplamaktır. Ayrıca ilerleyen süreç için il bazlı Mobese kameralarının konfigürasyonu tanımlanarak bu mesafe kullanılıp sokaklarda boş park yeri uygulaması oluşturularak belediyelerin kullanması sağlanabilir. Derinlik tahmininde kullanılan yöntemlere Bölüm 2'de değinilmektedir. Bu yöntemlerden görüntü segmentasyonu ve ters perspektif metodu, derin öğrenme ile birlikte kullanılarak bu uygulama gerçekleştirilmektedir. Uygulama Windows 10 işletim sistemine sahip bir bilgisayarda Pycharm geliştirme ortamı kullanılarak Python dilinde CPU kullanılarak yazılmıştır. Görüntü elde edinimi çeşitli (Çorlu, Bursa) Mobese kameralarındaki gerçek zamanlı yayınlardan kırpılmış ya da internet ortamından elde edilmiştir. Uygulamada elde edilen görüntülerin çekimindeki kameranın yerden yüksekliği, açısı ve diğer parametreleri hakkında bilgi sahibi olunmamaktadır. Bu yüzden uygulamaya ilk olarak görüntünün segmentasyona uygun açılı hale gelebilmesi için görüntünün konfigürasyonu ile başlamaktadır. Görüntüde, kameranın çekim açısına göre nesneler önden, yandan ve üstten görülebilmektedir. Örneğin Şekil 4.4'de görüntü araçların karşısındaki bir kameradan çekilmekte ve perspektif git gide daralmaktadır. Görüntüdeki araçlar aşağı yukarı aynı uzunluğa sahip olmalarına rağmen kameradan uzaklaştıkça daha küçük ve birbirini kapsayacak şekilde görülmektedir. Buda görüntü konfigürasyonunu zorunlu hale getirmektedir.



Şekil 4.4. Bursa Mobese kameralarından bir görüntü

Uygulamada işleme tabi tutulacak görüntüde araçların yatay bir eksene hizalanması sağlanmaktadır. Bunu yaparken döndürme, projeksiyon matrisi ve BEV kullanılmaktadır. Buradaki amaç belirli bir bakış açısından tüm pikselleri bu düz düzleme eşlemektir.



Şekil 4.5. Bursa Mobese kamera görüntüsü örneği

Örnek görüntü olarak Şekil 4.5'deki görüntü kullanılmaktadır. Bu görüntüye belirlenen koordinatlar doğrultusunda ters perspektif haritalama uyguladığında Şekil 4.6'daki sonuca ulaşılmaktadır. Bu işlem OpenCV' deki getPerspectiveTransform() ile oluşturulan matrisin, warpPerspective() fonksiyonuna gönderimi ile gerçekleştirilmektedir.



Şekil 4.6. Şekil 4.5' deki görüntünün transformu

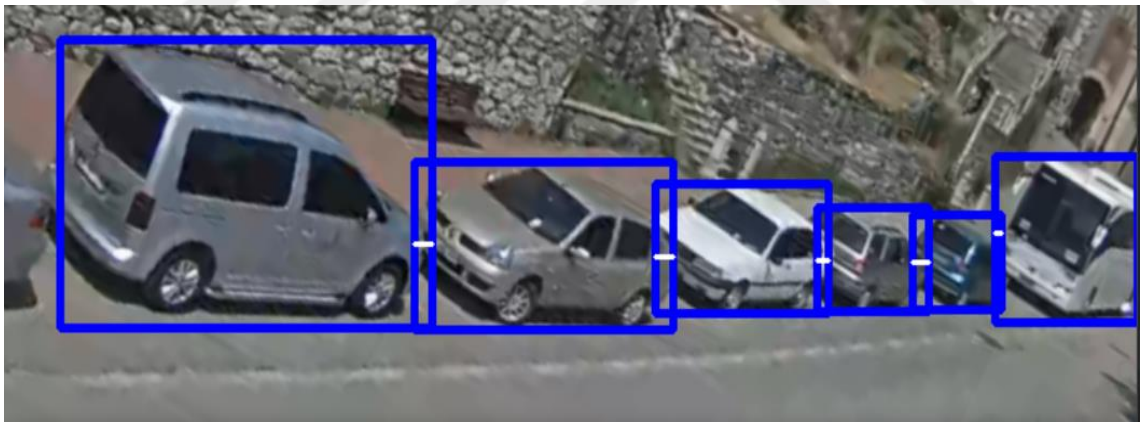
İstenilen görüntüye ulaşıldıktan sonra 4.1.1. Akış şeması başlığında yer alan akış şemasındaki sıra ile ilerlendiğinde ilk olarak değişkenler tanımlanır. Daha sonrasında segmentasyon işleminin belirgin gözükmesi için siyah ekran oluşturulur. Bu aşamadan sonra nesne tespiti için derin öğrenme kullanılmaktadır. Uygulamada OpenCV DNN modülü ile desteklediği Mask R-CNN modeli kullanılmaktadır. Kullanılan DNN modülünün içeriği 2 dosyadan oluşmaktadır. Uygulamada OpenCV' de bir DNN modelinin tercih edilmesinin sebepleri 3. Bölüm' deki uygulamada da kullanıldığından,

3.2. numaralı bölümde detaylıca yer verilmektedir. Mask R-CNN modeli yüklendikten sonra nesneler tespit edilmektedir. Tespit edilen her bir araç için maskeleme işlemi yapılmaktadır. Maskelenen aracın Harris metodu ile ön ve arka olmak üzere uç köşeleri tespit edilir ve daha sonrasında kullanmak amacıyla koordinatları değişkenlere atanır.

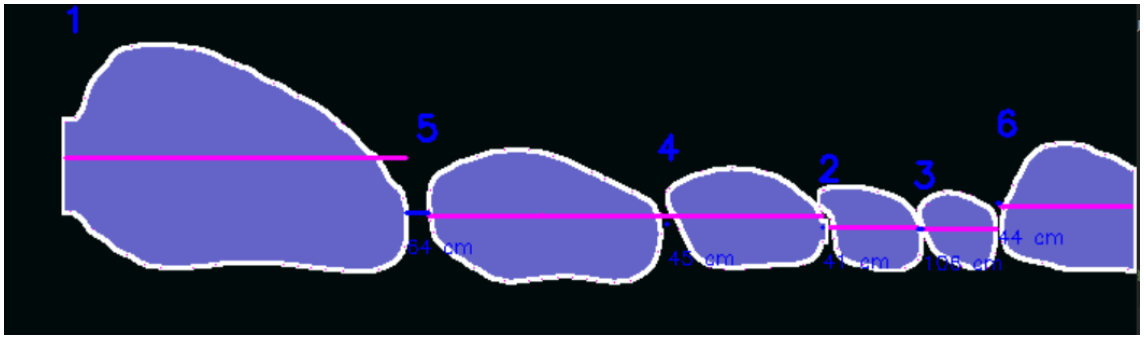
Nesne tanımlama sırasında araçlar görseldeki sırayla soldan sağa gibi bir sırayla tespit edilmediğinden eşleme gerektirmektedir. Bu işlem ilk tespit aşamasındaki dikdörtgen kutuların koordinatlarının maskelenme yapılan roi koordinatlarıyla eşleştirilmesi ile yapılmaktadır. Sıralanan araçların maskeleme öncesi ve sonrası uzunlukları piksel cinsinden hesaplanıp sonuçlar yazdırılmaktadır. Uzunluklar hesaplanırken Denklem 4.1'de belirtilen iki nokta arası uzaklık formülü kullanılmaktadır.

$$|AB| = \sqrt{(X_2 - X_1)^2 - (Y_2 - Y_1)^2} \tag{4.1}$$

Araçlar arası mesafe Şekil 4.7'de beyaz çizgi ile belirtilmektedir. Bu değer Şekil 4.8'de maskelenme uygulandığında dikdörtgenlerden kaynaklanan boşluklar kaybolduğundan artmaktadır.



Şekil 4.7. Mobese görüntülerinden nesne ve mesafe tespiti



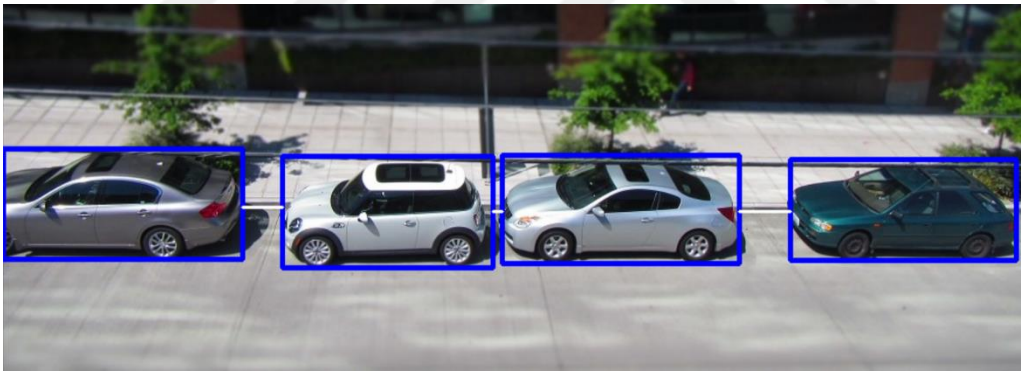
Şekil 4.8. Maskelenmiş görüntüden nesne ve mesafe tahmini

Şekil 4.8'deki görüntüde araçlar arası mesafeyi hesaplarken, dikdörtgenden maskelemeye geçerken oluşan kaybı (ilk aracın arkasından, ikinci aracın önünden) Şekil 4.7'deki bulunan mesafeyle topladığımızda elimizde piksel cinsinden yeni bir mesafe edilmektedir. Daha sonrasında araçlar giderek daha küçük gözüktüğünden mesafedeki oranda değişmektedir. Genel araç uzunlukları değerlendirilip ortalaması alındığında yaklaşık olarak 450 cm gibi bir değer elde edilmektedir. Bu değer ve öncesinde elde ettiğimiz veriler kullanılarak Denklem 4.2 ve Denklem 4.3'deki gibi bir oran orantı kurulmaktadır. Denklemlerden bulunan Mesafe_1 ve Mesafe_2'nin ortalaması alınarak elde edilen mesafe tahmini Şekil 4.8'deki gibi cm cinsinden görüntüye yazdırılır.

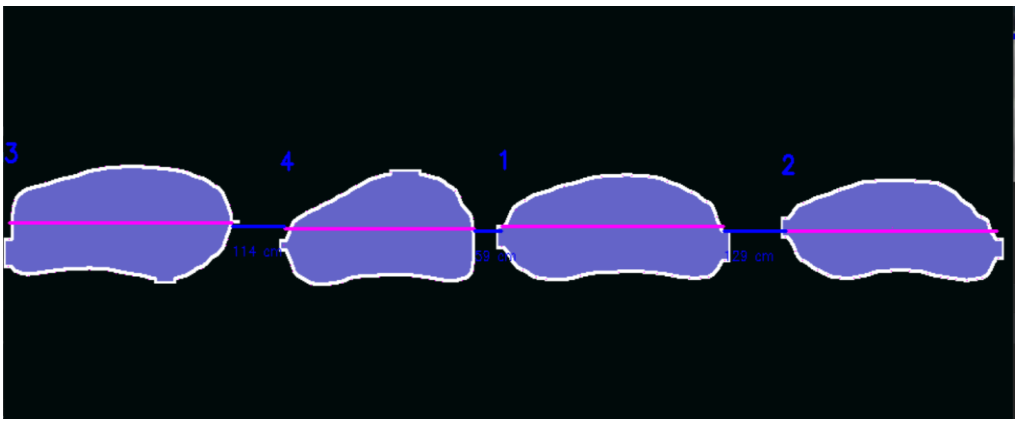
$$Mesafe_1 = (MaskelenmişMesafe * 450)/MaskelenmişAraçUzunluk1 \quad (4.2)$$

$$Mesafe_2 = (MaskelenmişMesafe * 450)/MaskelenmişAraçUzunluk2 \quad (4.3)$$

Uygulamada direk karşı açıdan çekilmiş başka bir örnek görüntü kullanıldığında da Şekil 4.9 ve Şekil 4.10'daki sonuçlar elde edilmektedir. Ve araçlar arası aralık değerleri sırasıyla 114 cm, 59 cm, 129 cm olarak bulunmaktadır.



Şekil 4.9. İki boyutlu görüntüden araç ve mesafe tespiti örneği



Şekil 4.10. Maskelenmiş görüntüden araç ve mesafe tespiti örneği

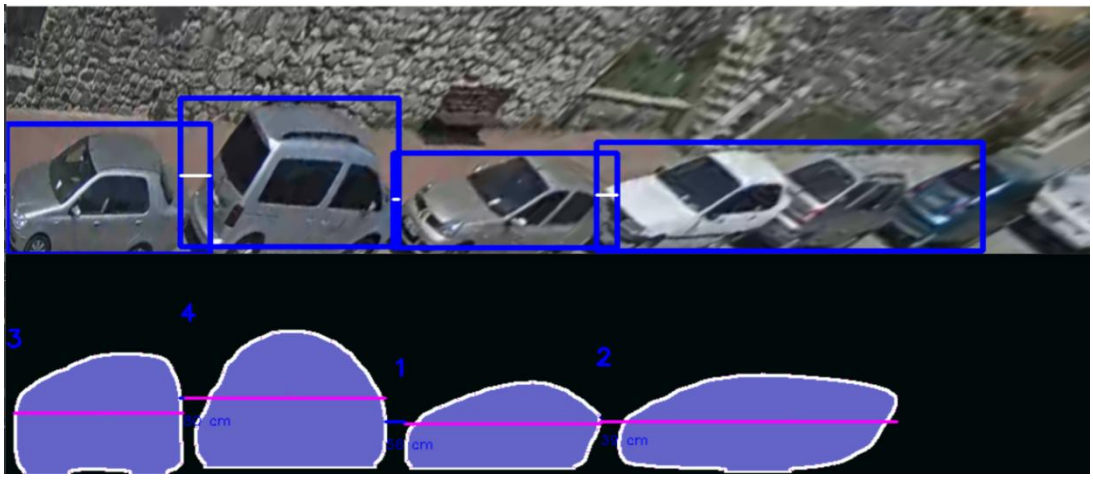
4.3. Bulgular ve Tartışma

Bu bölümdeki çalışmada en önemli çıkarım görüntülerin konfigürasyonun doğru gerçekleştirilmesi gerektiğidir çünkü bu adım sonraki süreçlerin başlangıcıdır. Eğer bu aşamada hata yapılırsa ve doğru koordinatlar seçilmezse alınacak sonuçlar Şekil 4.11'deki gibi hatalı mesafe tespitine neden olmaktadır.



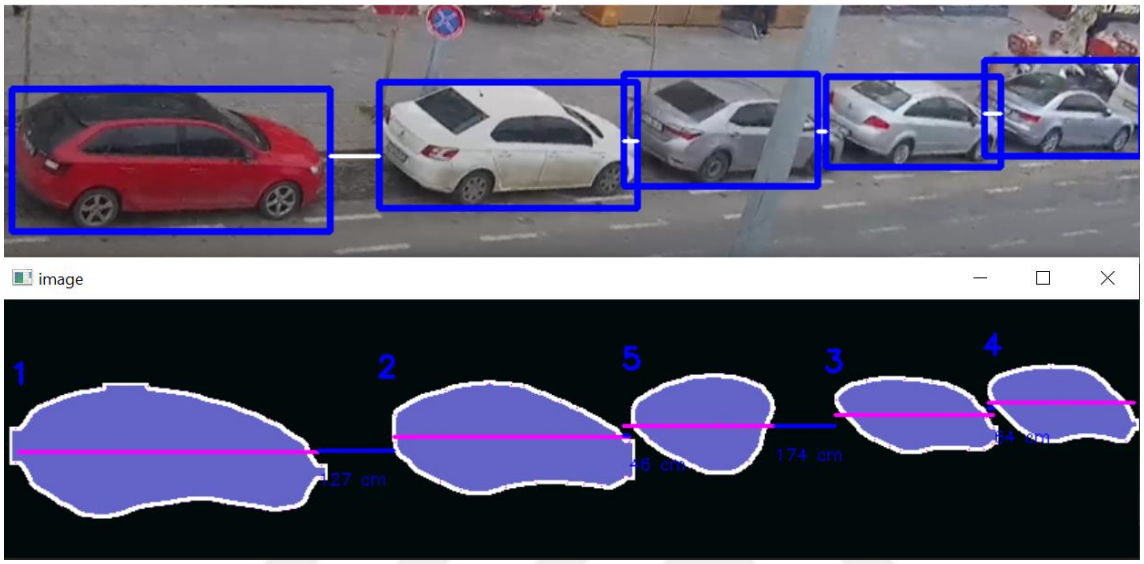
Şekil 4.11. Uygulamada hatalı koordinat seçimi ve sonucu

Yine aynı şekilde hatalı koordinat seçimi sonucunda ters perspektif haritalama uyguladığında Şekil 4.12'deki sonuca ulaşılmaktadır. Şekil 4.12'de uzaktaki araçların görüntüde kapladıkları yer neredeyse orijinal boyutlarına yaklaşırken bu seferde büyütme ve döndürme sonucu bulanıklaşmaya sebep olmaktadır. Görüntü kalitesini düşüren bulanıklaşma uygulamanın ikinci aşaması olan nesne tespiti aşamasında birden fazla aracın bir araç gibi algılamasına sebep olmaktadır.



Şekil 4.12. Uygulamada hatalı koordinat seçimi ve sonucu

Çorlu Mobese görüntüsüne ait bir başka uygulamada kameranın önüne gelen aydınlatma direği yüzünden hata oluşmaktadır. Direk görüntüdeki araçlardan birinin doğru tespit edilmesini engellemekte ve hatalı segmentasyona sebep olmaktadır. Şekil 4.13'de görüldüğü üzere aracın ön kısmı algılanamadığından boş park mesafesine katılmıştır.



Şekil 4.13. Uygulamada görüntü engeli ve sonucu

5. ÇÖZÜM YAKLAŞIMLARININ ÖZETİ VE KARŞILAŞILAN ZORLUKLAR

5.1. Literatürdeki Çözüm Yaklaşımlarının Özeti

İki boyutlu görüntülerdeki cisimlerin tespitine yönelik literatürde birçok görüntü işleme çalışması bulunmaktadır (Şentaş ve diğ., 2019), (Eken ve diğ., 2018), (Şentaş ve diğ., 2020), (Kul ve Sayar, 2021). Bu gibi gerçek dünyadaki 2D görüntü işleme uygulamalarıyla yapılan çalışmaların (Bal ve diğ., 2021), (Kul ve diğ., 2015), (Kul ve Sayar, 2022) çoğunda, nesnelerin derinliğini belirlemek çok önemlidir.

Derinlik belirleme çalışmalarının başlangıcından bu yana, perspektif de dahil olmak üzere birçok problemle karşılaşılmıştır. Sinir ağlarını eğiterek iki boyutlu görüntülerden derinlik tahmini için kullanılan derin öğrenme, birçok çalışma (Liu ve diğ., 2015), (Mousavian ve diğ., 2017), (Zhao ve diğ., 2020), (Ming ve diğ., 2021), (Mertan ve diğ., 2022) için karşılaşılan sorunları çözmede kullanılan popüler bir seçimdir. Ayrıca, görüntüde küçük bir bozulma (piksel kaybı) olsa bile derinlik tahmin edilebilmektedir. Sonuç olarak, derinlik tahmini için derin öğrenme teknikleri literatür gövdesinde önemli bir rol oynamaktadır.

Monoküler derinlik tahmini, tek bir RGB görüntüsü verilen her pikselin derinlik değerini (kameraya göre mesafe) tahmin etme görevidir. Literatür Saxena ve arkadaşları (2006) tarafından, monoküler derinlik tahmini için makine öğrenimini kullanan ilk uygulamalardan biri olarak başlamıştır. Regresyon kaybını en aza indirerek ve denetimli öğrenme yoluyla 2D görüntülerde monoküler ipuçları kullanarak derinliği doğrudan azaltmayı amaçlamaktadır. Eigen ve arkadaşları (2014) ise ilk kez monoküler derinlik tahminine dayanan derin öğrenmeyi kullanmışlardır. Bahsedilen bu çalışmada (Eigen ve diğ., 2014) derinlik haritasını oluşturmak ve geliştirmek için, ağların yerel özellikleri öğrendiği bir yapı vardır.

Stereo çiftleri kullanarak ağın doğrudan öğrenilmesini sağlayan yöntem, Garg ve arkadaşları (2016) tarafından tanıtılmıştır. Buna paralel olarak, tek bir görüntüden denetimsiz derinlik tahminine sıralı monoküler görüntüler kullanan yöntemler Zhou ve arkadaşları (2017) tarafından geliştirilmiştir. Birçok çalışmada (Liu ve diğ., 2015), (Godard ve diğ., 2017), artan popülaritesinin bir sonucu olarak monoküler derinlik

tahmini için derin öğrenme tekniklerini kullanılmıştır. Tablo 5.1'den çalışmamızda yer verdiğimiz literatürdeki çözüm yaklaşımlarının özetine ulaşılmaktadır.

Tablo 5.1. Literatürdeki çözüm yaklaşımlarının özeti

Çalışma	**Yıl**	**3D Nesne Algılama**	**Segmentasyon Kullanımı**	**Referans Nesne Kullanımı**	**IPM (BEV) Kullanımı**
Xie ve diğ.	2016	X			
Chen ve diğ.	2016	X	X		
Mousavian ve diğ.	2017	X			
Qin ve diğ.	2019	X			
Liu ve diğ.	2020	X			
Liu ve diğ.	2010		X		
Ladicky ve diğ.	2014		X		
Eigen ve Fergus	2015		X		
Ye ve diğ.	2017		X		
Lyu ve diğ.	2019		X		
Guizilini ve diğ.	2020		X		
Pinto ve diğ.	2020		X		
Masoumian ve diğ.	2021			X	
Kul ve Sayar	2022			X	
Auty ve Mikolajczyk	2022		X	X	
Schulter ve diğ.	2018				X
Mani ve diğ.	2020				X
Gosala ve diğ.	2022		X		X
Sun ve diğ.	2022				X

5.1.1. 3D Nesne Algılama ile Derinlik Tahmini

Son zamanlarda yapılan çalışmalar, KITTI (URL-2) gibi veri tabanlarının değerlendirilmesinden elde edilen sinir ağlarının aslında tek görüntülerden doğru derinlik haritaları üretebileceğini göstermektedir. Aynı zamanda, bu veri tabanları monoküler derinlik tahmininin doğruluğuna odaklanmaktadır. Monoküler görüntü tabanlı 3D nesne algılama, nesnelerin derinliğini ve işgal ettikleri alanları hesaplamak için kullanılan bir tekniktir. Bu yöntem, robotik ve otonom sürüş araçlarına olan ilginin artmasıyla yaygınlaşmıştır.

Literatürde Mousavian ve arkadaşlarının (2017) çalışması, araçların 3D sınırlayıcı kutularının hesaplanmasıyla derinlemesine bir genel bakış sağlar. 2D dedektör tarafından oluşturulan iki boyutlu sınırlayıcı kutuyu kullanarak, nesneleri içeren görüntü segmentlerinden nesnelerin hangi yöne baktığını tahmin eder. Ayrıca nesneleri algılar ve bir 2D-3D dönüşüm çerçevesi önerir. Son olarak, öğelerin 3D yerleşimleri 2D-3D kutu geometrik kısıtlamaları kullanılarak belirlenir. Xie ve diğerleri (2016), 2D videoları ve görüntüleri stereoskopik 3D formatına otomatik olarak dönüştürmek için derin sinir ağlarını kullanmaktadır. Mono3D (Chen ve diğ., 2016), amacı otonom sürüş alanında tek bir monoküler görüntüden 3D nesne algılama gerçekleştirmektir. Nesnelerin yer düzleminde olması gerektiği gerçeğini kullanarak nesne adaylarını 3D'ye yerleştiren bir enerji minimizasyon yaklaşımı önermektedir.

MonoGRNet (Qin ve diğ., 2019), 2D nesne algılama, örnek derinlik tahmini, 3D yerelleştirme ve yerel köşe regresyonundan sorumlu, göreve özel dört alt ağdan oluşan tek, birleşik bir ağdır. Bu bileşenler daha sonra 3D kutuyu küresel bir bağlamda iyileştirmek için bir araya getirilir. Ağ, aşamalı bir şekilde eğitilir ve ardından sonucu elde etmek için ağ uçtan uca eğitilir. Farklı bir çalışmada (Liu ve diğ., 2020) 2D algılama ağının gereksiz olduğunu ve 3D algılama için ihmal edilemez gürültü oluşturduğunu savunmaktadırlar. Bu nedenle, 'SMOKE' adını verdikleri yeni bir 3D nesne algılama yöntemi önermektedirler. Bu yöntem tek bir keypoint tahminini regresyon 3D değişkenleriyle birleştirerek algılanan her nesne için bir 3D sınırlayıcı kutu tahmin eder. Diğer 3D algılama tekniklerinin aksine, bu yöntem karmaşık ön hazırlık gerektirmemektedir.

5.1.2. Segmentasyon Kullanımı ile Derinlik Tahmini

Monoküler görüntülerden derinlik haritaları tahmininde son teknoloji yöntemler hem problemlerin benzerliği hem de içsel zorlukları nedeniyle anlamsal bölümleme gibi diğer benzer bilgisayarlı görme görevleri için de kullanılır. Semantik(anlamsal) bilgide, 2D görüntüden derinlik tahmini için önemlidir. Bazı çalışmalar (Eigen ve Fergus, 2015), segmentasyonun spesifik ipuçları sağlayarak derinlik tahminine rehberlik etmesi gerektiği fikriyle bu yönü takip etmektedir. Semantik sınıf bilgisi kullanılarak, yamaların hangi nesneye ait olduğu bilinerek derinlik tahmini yapılabilir.

Liu ve diğerleri (2010), tek görüntüden derinlik tahminini anlamsal bölümleme göreviyle birleştirir. Anlamsal etiketlerin tahmin edildiği ve derinlik tahmini görevi için ek kısıtlamalar uygulamak için kullanıldığı iki aşamalı bir sistem tasarlarlar. Ladicky ve diğerleri (2014), piksel başına derinlik tahminlerini geliştirmek için modellerine semantiği dahil etmekte çözüm buldular. Ye ve diğerleri (2017), söz konusu makaledeki nesne örneği segmentasyon problemini çözmeye çalışırken örtüşen nesneler için derinlik bilgilerini kullanmaktadır. Lyu ve arkadaşları (2019), çalışmalarında büyük oranda hesaplama yükünü azaltabilecek gerçek zamanlı anlamsal bölümlemeye odaklanmaktadır. Segmentasyon sürecini hızlandırmak için geleneksel yöntemler, örnek görüntüyü azaltma eğilimindedir. Ancak, bu işlem bilgi kaybına neden olur. Bu nedenle, yüksek çözünürlüklü küresel kenar bilgilerini düşük çözünürlüklü sınıflandırma düzeyinde anlamsal bilgilerle birleştiren gerçek zamanlı kenar tabanlı bir segmentasyon ağı (ESNet) önerilmektedir. Guizilini ve arkadaşları (2020), geometrik öğrenime rehberlik etmek için semantik modelin kullanmasını önermektedir. Ağın farklı aşamalarında anlamsal bulguları birleştiren bir semantik rehberlik özelliği tasarlamaktadır. Pinto ve arkadaşları (2020), örnek segmentasyonu ve derinlik tahminini aynı anda yapabilen çok görevli bir öğrenme ağını kullanarak, paralaks hareket etkisinin sonucundaki verimliliği araştırmak için diğer algoritmalara karşı bir karşılaştırma sunmaktadır.

5.1.3. IPM Kullanımı ile Derinlik Tahmini

Bu çalışmada bahsettiğimiz yöntemlerden biri olan IPM farklı yöntemlerle birleşerek derinlik tahminine katkı sağlamaktadır. Buna örnek verecek olursak yakın zamanda

Gupta ve arkadaşları (2022) tarafından gerçekleştirilen çalışmada stereo görüntü çiftlerinden yerleşim tahmini için uçtan uca bir sinir mimarisi olan SBEVNet önerilmektedir. Burada, eğitim süresi boyunca bilgi IPM özelliklerinden stereo özelliklere aktarmak için modlar arası damıtma kullanılarak yöntemler birleştirilir ve BEV (Bird Eye View) düzeni tahmini için yeni bir temsil önerilir. Ancak bu çalışma monoküler görüntü üzerine yapılan araştırmalar bazında gerçekleştirildiğinden stereo görüntü çifti ve lidar gibi diğer yöntemleri ele almamaktadır.

BEV haritalarının, yorumlanması ve işlenmesi kolay olmakla birlikte zengin mekânsal bağlam sağlama yetenekleri nedeniyle sahne anlayışı için en güçlü metotlardan biri olarak ortaya çıkmıştır. Bu haritalar, işlemler için BEV uzayında nesne örneği tanımlamasının yanı sıra doğru sahne bölümlemesine de büyük ölçüde dayanan birçok gerçek dünya görevinde kullanım alanı bulmaktadır. Monoküler derinlik tahminine katkılarına örnek verecek olursak Gosala ve arkadaşları (2022), önden görünüme sahip monoküler görüntüleri girdi olarak alan ve BEV' deki tutarlı panoptik segmentasyon maskelerini tahmin eden ilk uçtan uca eğitilebilir BEV panoptik segmentasyon mimarisini sunmaktadırlar. PanopticBEV mimarisi, giriş ön görüntüsündeki dikey ve düz bölgelere ait özellikleri bağımsız olarak BEV' e dönüştürmek için iki ayrı dönüştürücü kullanır. Ayrıca, BEV alanındaki pikselleri akıllı bir şekilde ağırlıklandırarak ön görüntü boyunca değişen tanımlayıcılık düzeylerini hesaba katmak için hassasiyete dayalı bir ağırlıklandırma şeması sunar.

Mani ve arkadaşları (2020) ise bir sürüş platformundan alınan tek renkli bir görüntü verildiğinde, yolun ve diğer trafik katılımcılarının kuş bakışı yerleşimini tahmin etmeye çalışır. Önerdikleri MonoLayout, bir monoküler girdi görüntüsü kullanarak kuşbakışı görünüm düzenini tahmin etmek için bir kodlayıcı-kod çözücü modeli kullanır. Schulter ve arkadaşları (2018), görüntü semantiğini kuşbakışı görünümüne yansıtmak için derinlik tahminini kullanır. Ayrıca, çekişmeli öğrenme yoluyla BEV görüntülerini iyileştirmek için Açık Sokak Haritaları verilerini kullanırlar. Sun ve arkadaşları (2022), Şekil 5.1'de görüldüğü üzere birden fazla insan içeren tek bir görüntü verildiğinde, tüm insanların pozunu ve şeklini ve göreceli derinliklerini doğrudan tahmin edebilmek için vücut modelleri ve BEV yöntemini kullanmaktalar.



Şekil 5.1. Sun ve arkadaşlarının (2022) çalışmasından görüntü

5.1.4. Referans Nesne Kullanımı ile Derinlik Tahmini

Bu çalışmada bahsettiğimiz yöntemlerden biri olan referans nesne kullanımından Kul ve Sayar (2022) çalışmalarında yararlanmaktadır. Çalışmada bir referans nesnesi kullanarak 2 boyutlu bir görüntüdeki bir duvarın boyutunun nasıl hesaplanacağı hakkında bir yaklaşım sunmaktadırlar. Yine güncel olan bir çalışmada (Auty ve Mikolajczyk, 2022), biyolojik görüş sistemlerinden faydalanarak, göreceli boyut, tanıdık boyut ve mutlak boyutun biyolojik ipuçlarını taklit etmek için anlamsal bilgi ve nesne boyutları ve ilişkileri hakkında ön bilgilere odaklanmaktadır. Dış bilgi sağlamak için örnek segmentasyon modellerini kullanılıyor ve nesnelerin ortalama gerçek dünya boyutu hakkında bir ön bilgi sağlıyor.

Masoumian ve arkadaşları (2021), yaptıkları çalışmada biri nesne tespiti ve diğeri derinlik tahmini için iki paralel derin ağ kullandılar. Tahmini derinlik, DepthNet' ten çıkarılır. YOLOv5 ile görüntünün içindeki nesneler yerelleştirilir ve sınıflandırılır. Tahmini derinlik görüntüsü üzerinde sınırlanmış kutular tarafından tanımlanan her bir nesnenin lokalizasyonu tespit edilir ve nesnenin ilgili mesafesi, tanımlanan sınırlı kutu içindeki tüm piksellerin ortalama tahmini mesafesi ile hesaplanır.

Günümüze bakacak olursak yöntemler birlikte ya da tek başına kullanılabilmektedir. Ming ve diğerleri (2021) tarafından monoküler derinlik tahmini için derin öğrenme tabanlı modelleri tek tek kategorize edilip, özetlenmekte ve özellikleri karşılaştırılmaktadır.

Derin öğrenme yöntemleri ve kullanılan veri tabanları açısından genel bir bakış açısı sağlamayı amaçlayan çalışmalarda (Khan ve diğ., 2020), (Xiaogang ve diğ., 2020) karşılaştırmalara yer vermektedir. Son olarak literatürdeki çalışmalar, yeni mimariler ve kayıp fonksiyonları geliştirerek birçok yaklaşım önermektedir.

Anlatılan çözüm yaklaşımlarının çoğunun derinlik tahmini için derin öğrenme yöntemlerinde kullandığı bazı yaygın veri kümeleri şu şekildedir.

KITTI, veri seti derin öğrenmeye dayalı derinlik tahmini, nesne tespiti ve takibi için kullanılan bir dış mekân veri setidir. Veri seti, "Yol", "Şehir", "Konut", "Kampüs" ve "Kişi" olmak üzere beş kategoride toplam 93.000 RGBD eğitim örneği içermektedir.

NYU Depth V2 veri seti, Silberman ve arkadaşları tarafından geliştirilen, derin öğrenmeye dayalı monoküler derinlik tahmini için bir iç mekân veri setidir. NYU Depth V2'nin orijinal görüntü boyutu 640x480 olup veri setinin derinliği 0,5 m ile 10 m arasında değişmektedir. RGB ve derinlik kamerası arasındaki konumsal sapma nedeniyle, derinlik haritaları eksik parçalar içerebilmektedir.

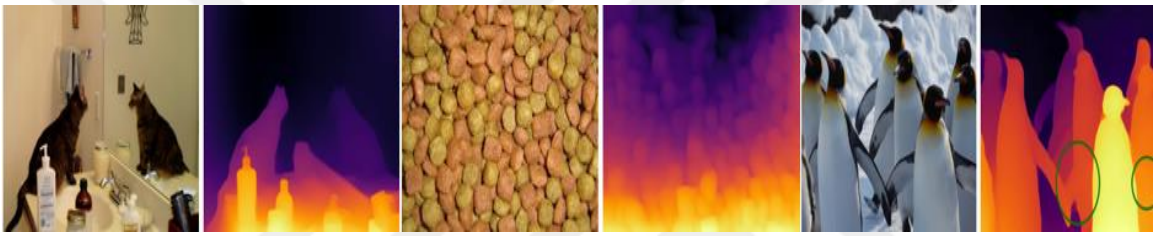
Make3D veri seti, Saxena ve arkadaşları tarafından geliştirilen, derin öğrenmeye dayalı monoküler derinlik tahmini için açık hava veri setidir. Make3D veri seti, bir lazer tarayıcı tarafından toplanan derinlik haritalarını içermektedir. Derinlik 5 m ila 81 m arasında değişmektedir.

5.2. Derinliği Belirlemede Karşılaşılan Zorluklar

Şimdiye kadarki bölümde literatürde iki boyutlu görüntü kullanılarak derinlik tahmini ile çalışılan çalışmalara yer verilmiştir. Bu yayınlar, bu çalışmada açıklanan yöntemler de dahil olmak üzere birçok yöntemi birleştiren çözümler içerir. Bu çözümler uygulandığında, sonuç her zaman başarılı bir şekilde elde edilemez. Bu bölümde literatürde yapılan araştırmaların karşılaştığı zorluklar anlatılmaktadır.

Karşılaşılan zorluklardan ilki görüntü segmentasyonu kullanımı bölümünde de anlatıldığı üzere semantik segmentasyonda aynı türden nesnelerin, aynı renkle işaretlenmesinden kaynaklı nesne çakışmaları meydana gelmesidir. Nesne sınırları düzgün belirlenemediğinden derinlik tahmininde hataya yol açmaktadır.

Karşılaşılan bir diğer sorun, görüntüdeki sahnenin ölçekleme aşamasındadır. Örneğin, odak uzunluğunun yanlış ayarlanması, kamera matrisinin doğru ölçeklendirilememesi vb. durumlar yanlış izdüşümlere neden olmaktadır. Ranftl ve arkadaşlarının (2022) eğittikleri model ile gerçekleştirilen testler sonucu, Şekil 5.2'de görüldüğü üzere ilk olarak yansıtıcı bir yüzey olan ayna ayırt edilemediğinden olmayan bir nesne (yansıyan kedi) tespit edilmiştir. Ortadaki görüntüde ise mamanın yukarıdan çekilmiş aynı derinliğe sahip görüntüsü hatalı derinlik sonuçları yansıtmaktadır. Penguenlerin bulunduğu sağ görüntüde ise yeşil dairelerle işaretlenmiş olan alanda öndeki penguenin yüzgeçleri arkasındaki penguenin bedeninin algılanması sebebiyle kaybedilmiş ve hata sebep olmuştur.



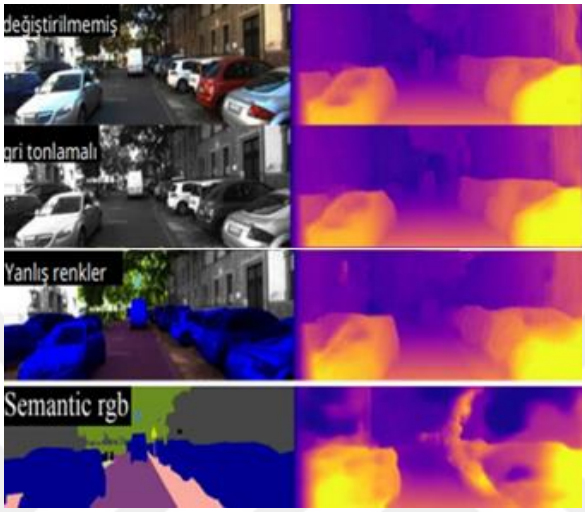
Şekil 5.2. Derinlik tahminindeki küçük hatalar (Ranftl ve diğ., 2022)

Tahmini derinlik ile sahnenin yapısı arasındaki ilişkiyi kullanarak yapılan bir çalışmada (Van Dijk ve De Croon, 2019) ulaşılan deneysel sonuçlardan şu problemler tespit edilmektedir. Eğitim setinde bulunmayan nesneler (köpek, sandalye vb.) görüntüde güvenilir bir şekilde algılanmamaktadır. Aynı çalışmada (Van Dijk ve De Croon, 2019) araba bir engel olarak tanınırken, diğer nesneler tanınmaz ve derinlik haritasında düz bir yol yüzeyi olarak görünmektedir. Ancak Şekil 5.3'de görüldüğü üzere yapay gölgeli rastgele bir nesne yerleştirerek, nesne eğitimde mevcut olmasa bile derinliği tahmin edebilmektedir. Burada gölgenin önemi vurgulanmaktadır.



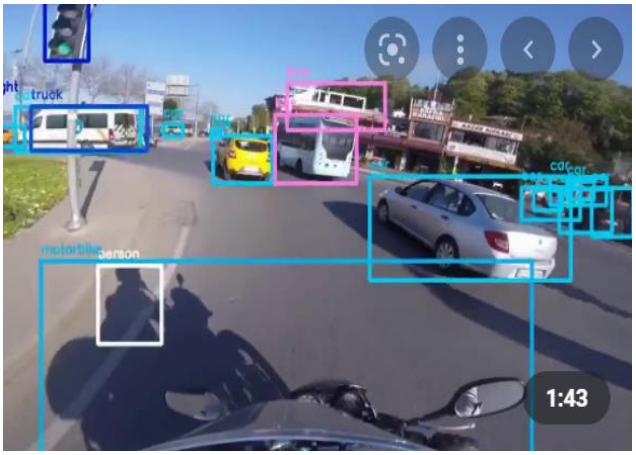
Şekil 5.3. Nesnelerin gölgeleri sayesinde algılanması (Van Dijk ve De Croon, 2019).

Yine aynı çalışmada (Van Dijk ve De Croon, 2019) Şekil 5.4'de görüldüğü üzere görüntünün gri tonlama ve yanlış eşleştirilmiş renklerden etkilenmediğini gösterirken, semantik RGB uygulandığında derinlik tahmininin bozulduğu görülmektedir. Buradan hareketle neden semantik segmentasyonu değil, örnek segmentasyonu tercih etmemiz gerektiği anlaşılmaktadır.



Şekil 5.4. Değiştirilmemiş, gri tonlamalı, yanlış renkli ve semantik rgb görüntüleri için örnek görüntüler ve derinlik haritaları (Van Dijk ve De Croon, 2019).

Derinlik tahmininde hataya neden olan problemlerden biride sahnenin geometrik ayrıntı bakımdan zengin olmasıdır. Bu nesne tespitinde benzerlikten ötürü yanlış nesnelerinde tespit edilmesine sebep olmaktadır. Örneğin derinliğini ölçeceğimiz nesne bir insan olsaydı Şekil 5.5'de görüldüğü üzere yerdeki motosiklet sürücünün gölgesi işaretlenmiş olduğundan hata yapılmasına neden olacaktır.



Şekil 5.5. Trafik akışı sırasında nesne tanıma yapan videodan bir görüntü

CNN ağlarının yoğun tahmin görevlerinde kullanılırken karşılaştıkları temel sorun kaba çıktılardır. Şekil 5.6'da görüldüğü gibi genellikle tahmin edilen görüntünün dış hatları çevresinde fark edilebilir ayrıntı eksikliğinden oluşur. Orijinal görüntüde 7 adet valiz görülebilirken derinlik haritasında bavulların ayrıntıları ve sayısı net olarak hesaplanamamaktadır.



Şekil 5.6. Derinlik tahmini ayrıntı eksikliği hatası (Ranftl ve diğ., 2022)

Gerçek zamanlı bir video akışından derinliği tahmin edebilmek için çok yüksek kriterlere sahip olmak gerekmektedir. Büyük bir doğruluk elde etmek için, araştırmacılar sıklıkla ek hesaplama süresi, parametreler ve sınırlamalara ihtiyaç duyan derin ağlar geliştirmektedirler. Bu, tahminin doğruluğunu sağlarken, gerçek zamanlı tahminler için bir gecikmeye neden olmaktadır.

Nesne ve özellik algılamanın yetersizliği derinlik tahminini önemli ölçüde etkilemektedir. Bunun için görüntüdeki her nesne ile çalışabilecek bir modele sahip olmak ve çeşitli veri kümelerine sahip olmak çok önemlidir. Ancak yüksek performanslı bilgi işlemek ile ilgili sorun ise belleğin doluluğudur ve cihazlar daha hafif derin ağ mimarileri tercih etmek istemektedirler. Böylelikle hem yüksek performansla çalışacak hem de küçük veri setiyle mükemmel modeli oluşturması beklenen sistemde şu an gerçekleştirilmeye çalışılmaktadır.

Literatürde derinlik tahminleri, değerlendirme matrisleri ve kriterleri gibi değerlerle yorumlanmaktadır. Çalışmaların çoğunda başarı oranı dikkate alınır ve karşılaştırılır. Tablo 5.2'de literatürden farklı olarak çalışmaların karşılaştığı zorluklar verilmiştir.

Tablo 5.2. İki boyutlu görüntüden derinlik tahmini sırasında karşılaşılan zorluklar

Karşılaşılan Zorluklar	**Yıl**	**Yöntem**	**Veriseti**
Birbirine yakın nesnelerin bir bütün olarak algılanması	2017	Ye ve diğ.	Pascal Voc 2012
	2018	Qi ve diğ.	KITTI and SUN RGB-D
Kamera pozlamasındaki değişiklikler ve nesnelerin gölgelerinin algılanması	2019	Van Dijk ve De Croon	KITTI
Ortamda sert ışıklandırma kullanılması ya da yeterli aydınlatma sağlanamaması	2015	Eigen ve Fergus	NYU Depth v2
Derinlikte çok benzer renkler	2015	Eigen ve Fergus	NYU Depth v2
	2018	Wu ve diğ.	NYU Depth dataset
Nesne boyutunu belirleyememe	2022	Kul ve Sayar	Özel Veri Seti
Veri setinde insan görsel sistemi tarafından kullanılan ipuçlarının örnekleri mevcut olsa da, ağın bu ipuçlarını öğrenememesi	2019	Mertan ve Duff	NYU Depth V2
Ortamdan kaynaklı yansıyan veya şeffaf nesne (cam, ayna)	2022	Ranftl ve diğ.	Multiple datasets
Nesnenin zeminle temas noktasının olmaması	2022	Sun ve diğ.	Özel Veri Seti (Relative Human)
Görüntünün parazitli olması (Kamera lensinin çizik olması, hava, toz, vb.)	2022	Görüntü işleme ve derin öğrenme ile iki boyutlu görüntüden mesafe tahmini uyg.	Özel Veri Seti
Kameranın sınırlı çözünürlüğü ile uzak piksel bulanıklaşması	2018	Schulter ve diğ.	KITTI and Cityscapes
Kamera açılarındaki bozulmalar	2022	Referans nesne kullanimi ile gerçek zamanli derinlik tahmini uyg.	Özel Veri Seti

6. SONUÇ

Monoküler görüntülerden derinlik tahmini, bilgisayarlı görme alanında önemli bir araştırma konusudur. Görüntülerden belirli bakış açılarında piksel bazında derinlik haritaları üretmeyi amaçlar. Bu tür derinlik bilgileri, üç boyutlu sahnelerin daha iyi anlaşılmasına yardımcı olur ve aynı zamanda iç mekânda konum belirleme, yükseklik tahmini, eşzamanlı konum belirleme ve haritalama (SLAM), görsel sınıflandırma gibi birçok bilgisayarla görme görevini kolaylaştırır.

Bu tezde, monoküler görüntülerde derinlik tahmininde literatürde sık kullanılan yöntemlerden bahsedilmiş ve gruplandırılmış olup, güncel çözüm ve zorluklara genel bir bakış sunulmuştur. İki boyutlu görüntüde karşılaşılan derinlik problemi, yöntemleri ve bu soruna neden olan faktörler anlatılmaya çalışılmıştır. Tek görüntü derinliği tahmini teknikleri, perspektif, nesne özellikleri, zemin dokusu ve engeller gibi yararlı ipuçlarından yararlanmaktadır. Bu ipuçlarından biri olan referans nesne kullanımı ve derinlik algısını kuvvetlendiren üç boyutlu kutu tahmini ile bir uygulama geliştirilerek tezde anlatılan zorluklar gerçek zamanlı olarak deneyimlenmektedir. Tezde anlatılan diğer iki yöntem olan görüntü segmentasyonu ve ters perspektif haritalama kullanılarak da ikinci bir uygulama geliştirilmektedir. Bu uygulama ile de iki boyutlu görüntüden mesafe çıkarımının zorlukları üzerine çalışılmaktadır. Bu iki uygulamadan yapılabilecek genel çıkarım ise derinliğin karmaşık olmasının aslında birçok parametreye sahip olması ve bu parametrelerin birbirine bağlı olmasıdır. Parametrelerden birinde gelişen hatanın zincirleme hataya sebep olarak sonucu bozması muhtemeldir. Bu parametrelerden ilki, benzer sahnelerde aynı kameradan çekilen görüntüler kullanılarak eğitildiğinde ve test edildiğinde iyi sonuç verir. Yalnız, gerçek dünya sahnelerinde, görüntüler genellikle çeşitli kameralar kullanılarak çeşitli açılardan çekilir ve bu da mevcut yöntemlerin önemli ölçüde performans düşüşüne yol açar. Bu sorunları hafifletmek için, gelecekteki çalışmalarda modelin çevrimiçi öğrenmesi veya alanın sisteme çevrimiçi adaptasyonu tekniklerinden yararlanılabilir. Ağ eğitimi için düzenleyiciler olarak geometrik veya istatistiksel kısıtlamalardan birlikte yararlanmak gelecekte sorunun çözümü açısından umut vericidir.

Yapay zekayı kullanan robotik, otonom sürüş ve sanal gerçeklik gibi alanlar, artık tek bir görüntü ile derinlik tahminini kolayca gerçekleştirmek istemektedir. Bu istemlerindeki

amaç çoklu görüntü çıkarımlarındaki zaman kaybını en aza indirmektir. Bundan dolayı yapay zekanın, insanlar gibi görüntüdeki ipuçlarını bulup kullanması ve derinlik hakkında çıkarımlarda bulunması beklenmektedir. Her geçen gün ilerleyen yapay zekâ, etkileşimlere cevap vermekte, hatalara neden olabilecek sorunları kontrol etmekte ve sahnenin durumuna göre kendi kararını verebilmekte ve gelecekte en doğru modeli uygulayarak mükemmele yakın bir sonuca ulaşabilmesi beklenmektedir. Bunun yanı sıra gelecekte, üç boyutlu verilerinde depolanabileceği büyük veri tabanlarının keşfedilmesi bekleniyor. Kendilerini gerçek zamanlı olarak ayarlayabilen, minimum denetimle yeni koşullara uyum sağlayabilen, kendilerini besleyebilen modeller, derinlik tahmini açısından da heyecan vericidir.

Son olarak, bu makalede iki boyutlu görüntülerdeki derinliği algılamadaki önceki deneyimler ve karşılaşılan zorluklar açıklanmaktadır. İki boyutlu bir görüntüden derinlik tahmininin gelecekte karşılaşabileceği zorluklar hakkında da çıkarımlar yapılmıştır. Aynı zamanda bu çıkarımlar iki uygulama geliştirilerek deneyimlenmiştir. Günümüzde ise monoküler derinlik tahmininin ana zorlukları, çoğu yaklaşımın, derinlik haritalarını yüksek doğrulukla tahmin etmesini sağlayan çok sayıda eğitilmiş parametreye sahip yüksek çözünürlüklü görüntülere ve büyük boyutlu derin öğrenme modellerine bağlı olmasındandır. Ancak bu modeller, yüksek hesaplama zamanı ve kaynakları gerektirdiğinden şu an için gerçek zamanlı uygulamalarda başarılı çalışamazlar. Bu nedenle, tahminin doğruluğunu sağlarken gerçek zamanlı tahmin için daha hafif bir ağın nasıl uygulanacağı bizlere gelecekteki bir araştırma yönüdür. Ancak, ağların karmaşıklığının azaltılması, sonuçların kalitesini ve doğruluğunu önemli ölçüde azaltacağından bu alanda hala keşfedilmesi ve çözülmesi gereken bir boşluk oluşturacak ve yeni zorlukları beraberinde getirecektir. Buda bir sonraki çalışmamıza yön verecektir.

KAYNAKLAR

Ahmadyan, A., Zhang, L., Ablavatski, A., Wei, J., Grundmann, M. (2021). Objectron: A Large Scale Dataset of Object-Centric Videos in the Wild with Pose Annotations. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

Akarun, L. (2020). *Üç boyutlu görme ve derinlik kameraları*. https://sarkac.org/2020/09/uc-boyutlu-gorme-ve-derinlik-kameralari/, (Ziyaret tarihi: 20 Nisan 2022).

Auty, D., Mikolajczyk, K. (2022). Monocular Depth Estimation Using Cues Inspired by Biological Vision Systems.

Bal, B., Erdem, T., Kul, S., Sayar, A. (2021). Image-based locating and guiding for unmanned aerial vehicles using scale invariant feature transform, speeded-up robust features, and oriented fast and rotated brief algorithms. *Concurrency and Computation Practice and Experience, 12*. DOI:10.1002/cpe.6766

Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R. (2016). Monocular 3D Object Detection for Autonomous Driving. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2147-2156. DOI: 10.1109/CVPR.2016.236.

Cutting, J.E., Vishton, P.M. (1995). Perceiving Layout and Knowing Distances: The Integration, Relative Potency, and Contextual Use of Different Information about Depth, *Perception of Space and Motion*, 69–117.

Duran, F., Kaya, A. (2018). Duruş ve Hareket Algılama Teknolojileri: Stereo, Time Of Flight ve Yapısal Işık Algılayıcılar. *Bilişim Teknolojileri Dergisi*, *11* (1), 57-73. DOI: 10.17671/gazibtd.327215

Eigen, D., Puhrsch, C., Fergus, R. (2014). Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. *In Advances in Neural Information Processing Systems*, *27*, 2366–2374.

Eigen, D., Fergus, R. (2015). Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-scale Convolutional Architecture. *2015 IEEE International Conference on Computer Vision (ICCV)*, 2650-2658, DOI:10.1109/ICCV.2015.304

Eken, S., Atay, B., Sönmez, B.C., Sayar, A. (2018). DocDig: Dijitalleştirilmiş Dokümanlarda İçerik Tabanlı Figür Arama. *Düzce Üniversitesi Bilim ve Teknoloji Dergisi, 6* (1), 68- 78.

Foley, H., Matlin, M. (2015). *Sensation and Perception* (5th ed.). Google-Books-ID: jLBmCgAAQBAJ

Garg, R., Kumar, B.G., Carneiro, G., Reid, I. (2016). Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue. *European Conference on Computer Vision*, Springer International Publishing, 740–756.

Godard, C., Mac Aodha, O., Brostow, G.J. (2017). Unsupervised monocular depth estimation with left-right consistency. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* DOI: 10.1109/CVPR.2017.699

Gosala, N., Valada, A. (2022). Bird's-Eye-View Panoptic Segmentation Using Monocular Frontal View Images. *IEEE Robotics and Automation Letters, 7* (2), 1968-1975. DOI: 10.1109/LRA.2022.3142418

Guizilini, V., Hou, R., Li, J., Ambrus, R., Gaidon, A. (2020). Semantically-Guided Representation Learning For Self-Supervised Monocular Depth. *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020).* Arxiv:2002.12319

Gupta, D., Pu, W., Tabor, T., Schneider, J. (2022). SBEVNet: End-to-End Deep Stereo Layout Estimation. *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV),* 667-676. DOI: 10.1109/WACV51458.2022.00074.

Khan, F., Salahuddin, S., Javidnia, H. (2020). Deep Learning Based Monocular Depth Estimation Method- A State of the Art Review. *Sensors, 20*(8),2272. DOI:10.3390/s20082272

Kirillov, A., He, K., Girshick, R., Rother, C., Dollar, P. (2019). Panoptic Segmentation. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). DOI:10.1109/CVPR.2019.00963

Kul, S., Eken, S., Sayar, A. (2015). Service oriented warning system for 48etection of abondened object in video surveillance. *23nd Signal Processing and Communications Applications Conference (SIU),* Malatya, Turkey, Mayıs 2015, 911-914. DOI:10.1109/SIU.2015.7129978

Kul, S., Sayar, A. (2021). Yapay Sinir Ağları, Destek Vektör Makineleri ve AdaBoost Algoritması ile Araç Sınıflandırmasının Değerlendirilmesi. *European Journal of Science and Technology*, *12*. DOI:10.31590/ejosat.1023889

Kul, S., Sayar, A. (2022). Wall Size Prediction from 2D Images with the Help of Reference Objects. *Innovations in Smart Cities Applications, 5*, 925-933. DOI:10.1007/978-3-030-94191-8_75

Ladicky, L., Shi, J., Pollefeys, M. (2014). Pulling Things out of Perspective. *2014 IEEE Conference on Computer Vision and Pattern Recognition,* 89-96. DOI: 10.1109/CVPR.2014.19.

Liu, B., Gould, S., Koller, D. (2010). Single image depth estimation from predicted semantic labels. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 1253–1260.

Liu, F., Shen, C., Lin, G. (2015). Deep Convolutional Neural Fields for Depth Estimation from a Single Image. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 5162-5170.

Liu, Z., Wu, Z., Toth, R. (2020). SMOKE: Single-Stage Monocular 3D Object Detection via Keypoint Estimation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* 996-997.

Lyu, H., Fu, H., Hu, X., Liu, L. (2019). Esnet: Edge-Based Segmentation Network for Real-Time Semantic Segmentation in Traffic Scenes. *2019 IEEE International Conference on Image Processing (ICIP),* 1855-1859. DOI: 10.1109/ICIP.2019.8803132.

Mani, K., Daga, S., Garg, S., Narasimhan, S. S., Krishna, M., Jatavallabhula, K. M. (2020). MonoLayout: Amodal scene layout from a single image. *The IEEE Winter Conference on Applications of Computer Vision*, 1689–1697.

Masoumian, A., Marei, D., Abdulwahab, S., Cristiano, J., Puig, D., Rashwan, H. (2021). Absolute distance prediction based on deep learning object detection and monocular depth estimation models.

Mertan, A., Duff, D.J. (2019). Analysis of Neural Networks for Depth Estimation, *27th Signal Processing and Communications Applications (SIU),* DOI:10.13140/RG.2.2.22062.56643

Mertan, A., Duff, D.J., Unal, G. (2022). Single Image Depth Estimation: An Overview. *Digital Signal Processing*, *123*. DOI: 10.1016/j.dsp.2022.103441

Ming, Y., Meng, X., Fan, C., Yu, H. (2021). Deep learning for monocular depth estimation: a review. *Neurocomputing*, *438,* 14-33. DOI: 10.1016/j.neucom.2020.12.089

Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J. (2017). 3D Bounding Box Estimation Using Deep Learning and Geometry. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 5632-5640. DOI:10.48550/ARXIV.1612.00496

Mur-Artal, R., Montiel, J.M., Tardos, J.D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics, 31* (5), 1147–1163.

Pinto, A., Cordova, M., Decker, L.G.L., Campana, J.L.F., Souza, M., dos Santos, A.A., Conceicao, J.S., Gagliardi, H.F., Luvizon, D.C., Torres, R.S., Pedrini, H. (2020). Parallax Motion Effect Generation Through Instance Segmentation And Depth

Estimation. *2020 IEEE International Conference on Image Processing (ICIP)*. DOI:10.1109/ICIP40778.2020.9191168

Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V. (2022). Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer. *IEEE Transactions on Pattern Analysis & Machine Intelligence, 44* (03), 1623-1637. DOI:10.1109/TPAMI.2020.3019967

Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Lecture Notes in Computer Science*, 9351. DOI: 10.1007/978-3-319-24574-4_28

Sanz, P. R., Mezcua, B. R., Pena, J. M. S. (2012). *Depth Estimation – An Introduction*. IntechOpen. DOI:10.5772/45904

Saxena, A., Chung S.H., Ng A.Y. (2006). Learning Depth from Single Monocular Images. *Advances in Neural Information Processing Systems, 18*, 1161–1168.

Schulter, S., Zhai, M., Jacobs, N., Chandraker, M. (2018). Learning to look around objects for topview representations of outdoor scenes. *European Conference on Computer Vision (ECCV)*, 787–802.

Schuon, S., Theobalt, Ch., Davis, J., Thrun, S. (2008). High-quality scanning using time-offlight depth superresolution. *IEEE CVPR Workshop on Time-Of-Flight Computer Vision*, 1-7.

Şentas, A., Kul, S., Sayar, A. (2019). Real-Time Traffic Rules Infringing Determination Over the Video Stream: Wrong Way and Clearway Violation Detection. *2019 International Artificial Intelligence and Data Processing Symposium (IDAP)*, Malatya, Türkiye, Eylül 2019. DOI:10.1109/IDAP.2019.8875889

Şentaş, A., Tashiev, I., Küçükayvaz, F., Kul, S., Eken, S., Sayar, A., Becerikli, Y. (2020). Performance evaluation of support vector machine and convolutional neural network algorithms in real-time vehicle type and color classification. *Evolutionary Intelligence, 13* (1), 83-91. DOI:10.1007/s12065-018-0167-z

Sun, Y., Liu, W., Bao, Q., Fu, Y., Mei, T., Black, M. (2022). Putting People in their Place: Monocular Regression of 3D People in Depth. DOI: 10.1109/CVPR52688.2022.01289

Szeliski, R. (2011). *Computer Vision: Algorithms and Applications*. Springer.

Town, C. (2011). *Computer Science Tripos Part II*. University Of Cambridge, https://www.cl.cam.ac.uk/teaching/1011/CompVision/Town%20_%20ComputerVision%20-%20L1.pdf, (Ziyaret Tarihi:10 Nisan 2022).

Ullman, S. (1979). The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences, 203* (1153), 405–426.

URL-1: https://en.wikipedia.org/wiki/Bilinear_interpolation, (Ziyaret Tarihi:10 Nisan 2022).

URL-2: http://www.cvlibs.net/datasets/kitti, (Ziyaret Tarihi:12 Nisan 2022).

URL-3: https://towardsdatascience.com/inverse-projection-transformation-c866ccedef1c , (Ziyaret Tarihi: 22 Nisan 2022).

URL-4: https://www.mathworks.com/discovery/edge-detection.html, (Ziyaret Tarihi:15 Mayıs 2022).

URL-5: https://medium.com/swlh/image-segmentation-using-k-means-clustering-46a60 488ae71, (Ziyaret Tarihi:15 Mayıs 2022).

URL-6: https://www.v7labs.com/blog/panoptic-segmentation-guide, (Ziyaret Tarihi:15 Mayıs 2022).

URL-7: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html, (Ziyaret Tarihi:22 Mayıs 2022).

URL-8: https://en.wikipedia.org/wiki/Lidar#cite_note-18, (Ziyaret Tarihi: 20.12.2022).

URL-9: https://en.wikipedia.org/wiki/Kinect, (Ziyaret Tarihi: 20.12.2022).

Valdez-Rodriguez, J. E., Calvo, H., Felipe-Riveron, E., Moreno-Armendariz, M. A. (2022). Improving Depth Estimation by Embedding Semantic Segmentation: A Hybrid CNN Model. *Sensors, 22* (4), 1669. DOI: 10.3390/s22041669

Van Dijk, T., De Croon, G. (2019). How Do Neural Networks See Depth in Single Images. *2019 IEEE/CVF International Conference on Computer Vision (ICCV),* 2183-2191. DOI:10.1109/ICCV.2019.00227

Ye, L., Liu, Z., Wang, Y. (2017). Depth-aware object instance segmentation. *2017 IEEE International Conference on Image Processing (ICIP),* 325-329. DOI:10.1109/ICIP.2017.8296296

Zhao, C., Sun, Q., Zhang, C., Tang, Y., Qian, F. (2020). Monocular depth estimation based on deep learning: An overview. *Sci. China Technol. Sci. 63*, 1612–1627. DOI:10.1007/s11431-020-1582-8

Zhou, T., Brown, M., Snavely, N., Lowe, D. (2017). Unsupervised Learning of Depth and Ego-Motion from Video. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 6612-6619. DOI:10.1109/CVPR.2017.700

Xiaogang, R., Wenjing, Y., Jing, H., Peiyuan, G., Wei, G. (2020). Monocular Depth Estimation Based on Deep Learning:A Survey. *2020 Chinese Automation Congress (CAC),* 2436-2440. DOI:10.1109/CAC51589.2020.9327548

Xie, J., Girshick, R., Farhadi, A. (2016). Deep3D: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks. *In European Conf. Computer Vision.*

Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J. (2018). Frustum PointNets for 3D Object Detection from RGB-D Data. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* DOI:10.1109/CVPR.2018.00102

Qin, Z., Wang, J., Lu, Y. (2019). MonoGRNet: A Geometric Reasoning Network for Monocular 3D Object Localization. *Proceedings of the AAAI Conference on Artificial Intelligence, 33*(01), 8851-8858.

Wu, Y., Ying, S., Zheng, L. (2018). Size-to-depth: A New Perspective for Single Image Depth Estimation.

EKLER

Ek-A

Üçüncü bölümdeki uygulamada bahsi geçen kod parçacıkları aşağıda yer almaktadır.

1- Aşağıdaki kod parçası uygulamadaki Aruco tanımlanmasına aittir.

```
parameters = cv2.aruco.DetectorParameters_create()
aruco_dict = cv2.aruco.Dictionary_get(cv2.aruco.DICT_5X5_1000)
corners,ids,rejected = cv2.aruco.detectMarkers(img, aruco_dict, parameters=parameters)
```

2- Aşağıdaki kod parçası Aruco Marker üzerinde piksel/cm oranı ile genişlik, yükseklik tespitini cm cinsinden sağlamaktadır.

```
aruco_perimeter = cv2.arcLength(corners[0], True) # Aruco Çevre Ölçüsü
pixel_cm_ratio = aruco_perimeter / 20
w = int(math.sqrt((topRight[1] - topLeft[1]) ** 2 + (topRight[0] - topLeft[0]) ** 2))
h = int(math.sqrt((topLeft[1] - bottomLeft[1]) ** 2 + (topLeft[0] - bottomLeft[0]) ** 2))
object_width = w / pixel_cm_ratio
object_height = h / pixel_cm_ratio
```

3- Aşağıdaki kod parçası görüntüde diğer nesneleri devre dışı bırakarak yalnızca kupa nesnesinin tespitini ve Aruco işaretçisinden elde piksel/cm oranı ile kupanın genişlik, uzunluklarını bulmayı sağlamaktadır.

```
(class_ids, scores, bboxes) = model.detect(img, confThreshold=0.3, nmsThreshold=.4)
for class_id, score, bbox in zip(class_ids, scores, bboxes):
      (x, y, w, h) = bbox
      class_name = classes[class_id]
      if class_name == "cup":
         cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 200), 3)
         cup_width = w / pixel_cm_ratio
         cup_height = h / pixel_cm_ratio
```

4- Aşağıdaki kod parçası Aruco işaretçisinin ekrana cm cinsinden uzaklığını bulmayı sağlamaktadır.

```
x = [227, 180, 151, 133, 116, 103, 95, 88, 79, 74, 69, 63, 60, 56, 53, 50,48]
y = [20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]
coff = np.polyfit(x, y, 2)
distance = w
A, B, C = coff
distanceCM = A * distance ** 2 + B * distance + C
```

Ek-B

Dördüncü bölümdeki uygulamada bahsi geçen kod parçacıkları aşağıda yer almaktadır.

1- Aşağıdaki kod parçası uygulamanın ilk aşaması olan görüntünün perspektif dönüşümüne aittir.

```
# 4 nokta alarak perspektif dönüşüm işlemi

import numpy as np
import cv2

# Koordinatların bir listesi, ilk giriş sol üst, ikinci giriş sağ üst, üçüncü giriş sağ alt ve dördüncüsü sol alt olacak şekilde oluştur.

def order_points(pts):
   rect = np.zeros((4, 2), dtype = "float32")

# Sol üst nokta en küçük toplamı alacak, sağ alt nokta en büyük toplamı alacak

   s = pts.sum(axis = 1)
   rect[0] = pts[np.argmin(s)]
   rect[2] = pts[np.argmax(s)]

# Noktalar arasındaki fark hesaplanır, sağ üst nokta en küçük farka sahip olacak, sol altta en büyük fark olacak

   diff = np.diff(pts, axis = 1)
   rect[1] = pts[np.argmin(diff)]
   rect[3] = pts[np.argmax(diff)]

 # Sıralı koordinatları döndürür

   return rect

#Görüntü ve noktalar sıralı olarak alınır.

def four_point_transform(image, pts):
   rect = order_points(pts)
   (tl, tr, br, bl) = rect

#Oluşacak yeni görüntünün genişliğini hesaplar, sağ alt ve sol alt arasındaki maksimum mesafe, x koordinatları veya sağ üst ve sol üst x koordinatları

   widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
   widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
   maxWidth = max(int(widthA), int(widthB))
```

#Oluşacak yeni görüntünün yüksekliğini hesaplar, sağ üst ve sağ alt arasındaki maksimum mesafe, y koordinatları veya sol üst ve sol alt y koordinatları

```
heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
maxHeight = max(int(heightA), int(heightB))
```

Yeni görüntünün boyutlarına sahip olduğumuza göre, görüntünün bir kuş bakışı görünümü (yani yukarıdan aşağıya görünümü) elde etmek için varış noktaları kümesini oluşturulur, yine sol üst, sağ üstteki noktaları belirtilir, sağ alt ve sol alt sıra

```
dst = np.array([
    [0, 0],
    [maxWidth - 1, 0],
    [maxWidth - 1, maxHeight - 1],
    [0, maxHeight - 1]], dtype = "float32")
```

Perspektif dönüşüm matrisini hesaplanıır ve ardından uygulanır

```
M = cv2.getPerspectiveTransform(rect, dst)
warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))
```

Dönüştürülmüş görüntü döndürülür

```
return warped
```

2- Aşağıdaki kod parçası uygulamanın ilk aşaması olan görüntünün perspektif dönüşümünü gerçekleştirmek için Madde1’ deki fonksiyonlara argüman gönderimini sağlamaktadır.

#İlgili kütüphanelerin çağrımı

```
from search.transform import four_point_transform
import numpy as np
import argparse
import cv2
import imageio
```

#Argümanları oluşturma ve ayrıştıma

```
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", help = "path to the image file")
ap.add_argument("-c", "--coords", help = "comma seperated list of source points")
args = vars(ap.parse_args())
```

#Görüntüyü yükleme ve kaynak koordinatlarını alma ((x, y) noktalarının listesi)

```
image = cv2.imread(args["image"])
pts = np.array(eval(args["coords"]), dtype = "float32")
```

Görüntünün kuş bakışı görünümünü elde etmek için four_point_transform fonksiyonu uygulama

```
warped = four_point_transform(image, pts)

#Orjinal görüntü ve dönüştürülmüş görüntüyü gösterme ve images klasörüne kaydetme

cv2.imshow("Original", image)
cv2.imshow("Warped", warped)
imageio.imwrite('./images/transform.png', warped)
cv2.waitKey(0)
```

3- Aşağıdaki kod parçası uygulamanın ikinci aşaması olan görüntüdeki araçlar arası mesafe ve derinlik tespitini sağlamaktadır.

```
#Gerekli kütüphanelerin çağırımı

import cv2
import numpy as np
import math

# Mask RCNN yükleme

net = cv2.dnn.readNetFromTensorflow("dnn/frozen_inference_graph_coco.pb",
"dnn/mask_rcnn_inception_v2_coco_2018_01_28.pbtxt")

# Görüntü okuma

img = cv2.imread("Images/bursa.png")
height, width, _ = img.shape

# Sonuçlar için siyah ekran oluşumu ve değişken tanımlama

black_image = np.zeros((height, width, 3), np.uint8)
black_image[:] = (10, 10, 0)

#Değişkenleri tanımlama

aracuzunluk=450
car_front=[]
car_back=[]
carpoints=[]
sortcarpoints=[]

# Nesneleri bulma

blob = cv2.dnn.blobFromImage(img, swapRB=True)
net.setInput(blob)
```

```
boxes, masks = net.forward(["detection_out_final", "detection_masks"])
detection_count = boxes.shape[2]

#Bulunan her bir nesne için for döngüsünde maskeleme işlemi yapılır, yeni koordinatları ve harris metodu ile uç köşeleri hesaplanır.

for i in range(detection_count):
        box = boxes[0, 0, i]
        class_id = box[1]
        score = box[2]
        if score < 0.5:
          continue

        x = int(box[3] * width)
        y = int(box[4] * height)
        x2 = int(box[5] * width)
        y2 = int(box[6] * height)
        car_front.append([x,y,i])
        car_back.append([x2,y2,i])
        roi = black_image[y: y2, x: x2]
        roi_height, roi_width, _ = roi.shape

        # Maskeleme işleme

        mask = masks[i, int(class_id)]
        mask = cv2.resize(mask, (roi_width, roi_height))
        _, mask = cv2.threshold(mask, 0.5, 255, cv2.THRESH_BINARY)
        cv2.rectangle(img, (x, y), (x2, y2), (255, 0, 0), 3)

        # Get mask coordinates

        contours, _ = cv2.findContours(np.array(mask, np.uint8),
        cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        contours = sorted(contours, key=cv2.contourArea, reverse=True)
        for cnt in contours:
                cv2.fillPoly(roi, [cnt], (200,100,100))
                cv2.drawContours(roi,[cnt], -1, (255, 255, 255), thickness=2)

        gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)

        # Harris corners bulma

        gray = np.float32(gray)
        dst = cv2.cornerHarris(gray, 2, 3, 0.04)
        dst = cv2.dilate(dst, None)
        ret, dst = cv2.threshold(dst, 0.01 * dst.max(), 255, 0)
        dst = np.uint8(dst)
```

```
            # Centroids bulma

            ret, labels, stats, centroids = cv2.connectedComponentsWithStats(dst)
            criteria = (cv2.TERM_CRITERIA_EPS +
            cv2.TERM_CRITERIA_MAX_ITER, 100, 0.001)
            corners = cv2.cornerSubPix(gray, np.float32(centroids), (5,5), (-1, -1),
            criteria)
            koseler=corners
            koordinatlar=[]
            second_sort=[]
            ilkson=[]
            for koor in koseler:
                    for xy in koor:
                        index=list(koor).index(xy)
                        xy=int(np.round(xy))
                        koor[index]=xy
                    koordinatlar.append(list(koor))
            koordinatlar.sort()
            ilkson.append(koordinatlar[0])
            ilkson.append(koordinatlar[len(koordinatlar)-1])
            ilkson.append(i)
            carpoints.append(ilkson)
            second_sort=sorted(koordinatlar,key=lambda x: int(x[1]))
            cv2.line(roi, (int(ilkson[0][0]), int((int(second_sort[0][1])+
            int(second_sort[len(second_sort)-1][1])) / 2)),  (int(ilkson[1][0]),
            int((int(second_sort[0][1]) + int(second_sort[len(second_sort)-1][1])) /
            2)),(255, 0, 255), 2)
            cv2.putText(black_image, "{}".format(i+1), (int(car_front[i][0]),
            int(car_front[i][1])), cv2.FONT_HERSHEY_SIMPLEX, .7, (255,0,0), 2)
            res = np.hstack((centroids, corners))
            res = np.int64(res)
            roi[res[:, 1], res[:, 0]] = [255, 0, 255]

#Araçların orijinal koordinatlarının sıralanması

car_front.sort()
car_back.sort()

#Araçların maskelenmiş ve orijinal görüntülere göre uzunluklarının hesaplanması

for i in range(len(car_front) ):
        for j in range(len(carpoints)):
            if (car_front[i][2]==carpoints[j][2]):
                    sortcarpoints.append(carpoints[j])
                    maracuzunluk = int(math.sqrt((0)**2 +(int(sortcarpoints[i][1][0])-
                    int(sortcarpoints[i][0][0]))**2))
                    print(i + 1, ". Maskelenmiş Arac Uzunluğu : ", maracuzunluk)
        garacuzunluk = int(math.sqrt(0+(int(car_front[i][0]) - int(car_back[i][0])) ** 2))
```

```
        print("", i+1, ". Arac Uzunluğu ", garacuzunluk)

#Araçlar arası mesafenin hesaplanması

for i in range(len(car_front)-1):
        aramesafe = int(math.sqrt(0+ (int(car_front[i + 1][0]) - int(car_back[i][0])) ** 2))
                    print(i,"-",i+1,"Araç Arası Mesafe :", aramesafe)
        m1aracuzunluk = int(math.sqrt((0)**2 +(int(sortcarpoints[i][1][0])-
        int(sortcarpoints[i][0][0]))**2))
        m2aracuzunluk = int(math.sqrt((0) ** 2 + (int(sortcarpoints[i+1][1][0]) -
        int(sortcarpoints[i+1][0][0])) ** 2))

        cv2.line(black_image,(int(int(sortcarpoints[i][1][0])+int(car_front[i][0])),
        int((int(car_front[i + 1][1])+int(car_back[i][1]))/2)), (int(car_front[i +
        1][0]+int(sortcarpoints[i+1][0][0])), int((int(car_front[i +
        1][1])+int(car_back[i][1]))/2)),(255, 0, 0), 2)

        cv2.line(img, (int(car_front[i + 1][0]), int((int(car_front[i +
        1][1])+int(car_back[i][1]))/2)), (int(car_back[i][0]), int((int(car_front[i +
        1][1])+int(car_back[i][1]))/2)),(255, 255, 255), 2)

            if(car_back[i][0]>car_front[i+1][0]):
                m_aramesafe=int(math.sqrt((int(car_front[i+1][0]+int(sortcarpoints[i+1][
                0][0]))-int(int(sortcarpoints[i][1][0])+int(car_front[i][0])))**2)
                +aramesafe)
            else:
                m_aramesafe = int(math.sqrt((int(car_front[i + 1][0] + int(sortcarpoints[i
                + 1][0][0])) - int(int(sortcarpoints[i][1][0]) + int(car_front[i][0]))) ** 2))

            print(i,"-",i+1,"Maskelenmiş Ara mesafe: ",m_aramesafe)
            mesafe1=int(np.round(m_aramesafe*aracuzunluk)/m1aracuzunluk)
            mesafe2 = int(np.round(m_aramesafe * aracuzunluk) / m2aracuzunluk)
            mesafeort=int((mesafe1+mesafe2)/2)

        cv2.putText(black_image,"{}cm".format(mesafeort), (int(sortcarpoints[i][1][0]+
        car_front[i][0]),int((int(car_front[i+1][1])+int(car_back[i][1]))/2)+25),
        cv2.FONT_HERSHEY_SIMPLEX,.4, (255, 0, 0), 1)

#Ekranda sonuçları gösterme

print("sortCarPoints:",sortcarpoints)
print("car_front: ",car_front)
print("car_back: ",car_back)
cv2.imshow("", img)
cv2.imshow("image", black_image)
cv2.waitKey(0)
```

KİŞİSEL YAYIN VE ESERLER

Akın, Z., Sayar, A. (2022). Challenges in Determining the Depth in 2-D Images. *The 16th International Conference on Innovations in Intelligent Systems and Applications (INISTA'22),* Biarritz,France, 8-10 August 2022.

ÖZGEÇMİŞ

İlk, orta ve lise öğrenimini Kocaeli'nde tamamladı. 2013 yılında girdiği Sakarya Üniversitesi, Bilgisayar ve Bilişim Bilimleri Fakültesi, Bilgisayar Mühendisliği Bölümü'nden 2017 yılında Bilgisayar Mühendisi olarak mezun oldu. 2018-2022 yılları arasında, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı'nda Yüksek Lisans öğrenimini tamamladı. 2018-2020 yılları arasında Senkron Bilgisayar Elek. İlet. Sist. Ltd. Şti.'de yazılım mühendisi olarak görev yaptı. 2021 yılında Verikar Yazılım firmasında yazılım mühendisi olarak görev yaptı.