T.C.
İSTANBUL GEDİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

GÖRÜNTÜ İŞLEME TEKNİKLERİ KULLANILARAK GERÇEK ZAMANLI PARK YERİ TESPİT SİSTEMİ

YÜKSEK LİSANS TEZİ

Berkay ŞENER

Yapay Zeka Mühendisliği Anabilim Dalı

Yapay Zeka Mühendisliği (Tezli) Yüksek Lisans Programı

HAZİRAN 2025
İSTANBUL

T.C.
İSTANBUL GEDİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

GÖRÜNTÜ İŞLEME TEKNİKLERİ KULLANILARAK GERÇEK ZAMANLI PARK YERİ TESPİT SİSTEMİ

YÜKSEK LİSANS TEZİ

Berkay ŞENER
(221239003)

Yapay Zeka Mühendisliği Anabilim Dalı

Yapay Zeka Mühendisliği (Tezli) Yüksek Lisans Programı

Tez Danışmanı: Dr. Öğr. Üyesi Şerife Esra DİNÇER

İstanbul 2025

T.C.
İSTANBUL GEDİK ÜNİVERSİTESİ
Lisansüstü Eğitim Enstitüsü Müdürlüğü

Jüri Tez Onay Formu

26.06.2025

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ MÜDÜRLÜĞÜ

Bu çalışma 26.06.2025 tarihinde aşağıdaki jüri tarafından Yapay Zeka Mühendisliği Anabilim Dalı, Yapay Zeka Mühendisliği (Tezli Yüksek Lisans) Programı Yüksek Lisans Tezi olarak kabul edilmiştir.

TEZ JÜRİSİ

Dr. Öğr. Üyesi Şerife Esra DİNÇER
Danışman
İstanbul Gedik Üniversitesi

Dr. Öğr. Üyesi Aytaç Uğur YERDEN
Üye (İmza)
İstanbul Gedik Üniversitesi

Dr. Öğr. Üyesi Enver AKBACAK
Üye (İmza)
İstanbul Fenerbahçe Üniversitesi

YEMİN METNİ

Yüksek Lisans tezi olarak sunduğum "Görüntü İşleme Teknikleri Kullanılarak Gerçek Zamanlı Park Yeri Tespit Sistemi" adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya'da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (26/06/2025)

Berkay ŞENER

ÖNSÖZ

Günümüzde artan araç sayısıyla birlikte, şehir içi otopark sorunu da giderek büyüyen bir problem haline gelmiştir. Özellikle yoğun nüfuslu bölgelerde sürücüler, park yeri bulmak için uzun süreler harcamakta ve bu durum hem zaman kaybına hem de çevresel olumsuzluklara yol açmaktadır. Bu çalışmada, görüntü işleme tekniklerinden faydalanarak gerçek zamanlı bir park yeri tespit sistemi geliştirilmesi hedeflenmiş ve bu sistemin pratikte uygulanabilirliği araştırılmıştır.

Tez süreci boyunca edindiğim bilgi ve deneyimler, sadece akademik anlamda değil, aynı zamanda problem çözme ve sistem geliştirme yeteneklerimi de büyük ölçüde geliştirmiştir. Bu süreçte beni destekleyen ve yönlendiren değerli danışmanım Şerife Esra DİNÇER'e, akademik bilgisi ve sabrıyla yolumu aydınlattığı için teşekkür ederim.

Ayrıca, aldığım her kararda beni desteklemeyi asla bırakmayan aileme teşekkür ederim. Şu ana kadar başardığım her şey, onların sevgisi ve desteği sayesinde oldu. Onlara sahip olduğum için minnettarım ve her zaman minnettar kalacağım.

Son olarak, nişanlıma sonsuz desteği ve bana olan inancı için teşekkür ederim. Birlikte geçirdiğimiz günlerdeki güzel anılar, bu süreçte bana çok yardımcı oldu. Onun sevgisi ve desteği için her zaman minnettar olacağım.

Bu tezin, görüntü işleme ve akıllı ulaşım sistemleri alanında yapılacak yeni çalışmalara katkı sağlamasını temenni ederim.

Haziran 2025 Berkay ŞENER

Bilgisayar Mühendisi

İÇİNDEKİLER

KISALTMALAR

CLAHE : Kontrast Sınırlı Adaptif Histogram Eşitleme- Contrast Limited Adaptive Histogram Equalization

CNN : Evrişimsel Sinir Ağı- Convolutional Neural Network

COCO : Bağlamdaki Yaygın Nesneler- Common Objects in Context

COINS : Otopark Doluluk Bilgi Sistemi- Car Park Occupancy Information System

CSV : Virgülle Ayrılan Değerler- Comma Separated Values

FCM : Bulanık C-Ortalama Kümeleme- Fuzzy C-Means

FLOPS : Saniye Başına Kayan Nokta İşlemleri- Floating Point Operations Per Second

FPS : Saniye Başına Kare- Frames Per Second

IoT : Nesnelerin İnterneti- Internet of Thing

IoU : Birleşim Üzerindeki Kesişim- Intersection Over Union

KB : Kilobayt

KHz : Kilohertz

K-NN : K-En Yakın Komşu Algoritması- K-Nearest Neighbors

LBP : Yerel İkili Desen- Local Binary Pattern

LCD : Sıvı Kristal Ekran- Liquid Crystal Display

LDA : Doğrusal Diskriminant Analizi- Linear Discriminant Analysis

MAE : Ortalama Mutlak Hata- Mean Absolute Error

mAP : Ortalama Hassasiyet- Mean Average Precision

MB : Megabayt

OCR : Optik Karakter Tanıma- Optical Character Recognition

RCNN : Bölge Tabanlı Evrişimsel Sinir Ağları- Region BasedConvolutional Neural Networks

RGB : Kırmızı Yeşil Mavi- Red Green Blue

ROI : İlgi Bölgesi- Region of Interest

SSD : Tek Atış Dedektörü- Single Shot Detector

SVM : Destek Vektör Makinesi- Support Vector Machine

TF : TensorFlow

XML : Genişletilebilir İşaretleme Dili- Extensible Markup Language

ÇİZELGE LİSTESİ

Sayfa No.

ŞEKİL LİSTESİ

Sayfa No.

GÖRÜNTÜ İŞLEME TEKNİKLERİ KULLANILARAK GERÇEK ZAMANLI PARK YERİ TESPİT SİSTEMİ

ÖZET

Teknoloji, günlük yaşamımızda olumlu bir etki yaratmış olup, çevremizi kontrol etmemizi ve bilgiye daha hızlı erişmemizi sağlamıştır. Park yeri tespit sistemleri, otoparkların doluluk durumunu izlemek için kullanılan ve otomatik olarak çalışan kontrol sistemlerindendir. Bu sistemler, otoparkları optimize etmeye, karar verme süreçlerine bilgi sağlamaya ve sürücüleri boş park yerlerine yönlendirmeye yardımcı olur.

Bu çalışmada, otoparklara kuş bakışı yerleştirilmiş kameralardan gelen gerçek zamanlı video akışlarını analiz etmek için görüntü işleme tekniklerini içeren bir sistem ortaya konmaktadır. Görüntü işleme adımları, gürültü azaltma için Gauss bulanıklığı, medyan filtreleme ve ilgili özellikleri çıkarıp, giriş görüntülerinin kalitesini artırmak için eşikleme içermektedir. Daha sonra, OpenCV teknolojisi kullanılarak tespit edilen konturların iyileştirilmesi için genişletme gibi morfolojik teknikler uygulanmaktadır.

Geliştirilen sistem, Google'ın TensorFlow teknolojisini ve otopark görüntüleri üzerinde eğitilmiş bir CNN makine öğrenimi sınıflandırıcısını kullanan bir araç tespit sistemi ile karşılaştırılmıştır. Sistemler, otoparklardan alınmış 400 ve 679 görüntü karesi içeren iki video akışı içerisinden seçilen bir park yeri üzerinde karşılaştırılmıştır.

Geliştirilen sistemin başarılı araç tespit oranı, Google'ın TensorFlow sınıflandırma sistemine göre, 400 kare görüntü içeren birinci video akışı için %25,50, 679 kare görüntü içeren ikinci video akışı içinse %10,61 oranında daha yüksek bulunmuştur. Ek olarak, geliştirilen sistem, Google'ın TensorFlow sınıflandırma sistemine kıyasla daha hızlı çalışmakta ve daha az disk alanı kullanımı gerektirmektedir.

Ayrıca geliştirilen sistem, CNN algoritması tabanlı YOLOv8 teknolojisini kullanan başka bir sınıflandırma sistemiyle de karşılaştırılmıştır. Geliştirilen sistemin başarılı araç tespit oranı, YOLOv8 sınıflandırma sistemine göre, 400 kare görüntü içeren birinci video akışı için %2,75, 679 kare görüntü içeren ikinci video akışı içinse %2,66 oranında daha yüksek bulunmuştur. Ek olarak, geliştirilen sistem, YOLOv8 sınıflandırma sistemine kıyasla daha hızlı çalışmakta ve daha az disk alanı kullanımı gerektirmektedir.

Genel olarak, bu çalışma, gerçek zamanlı park yeri tespiti için ölçeklenebilir ve güvenilir bir çözüm sunarak akıllı park sistemlerinin gelişimine katkıda bulunmaktadır.

Anahtar Kelimeler: *Otoparklar, Boş park yeri, Araç tespiti, Görüntü işleme, Görüntü filtreleme.*

REAL TIME PARKING SPACE DETECTION SYSTEM USING IMAGE PROCESSING TECHNIQUES

ABSTRACT

Technology has had a positive impact on our daily lives, enabling us to control our environment and access information faster. Parking detection systems are one of the automated control systems used to monitor the occupancy of parking lots. They help to optimize parking areas, provide information for decision-making and guide drivers to empty parking spots.

This study presents a system includes image processing to analyze real-time video feeds coming from tha cameras located on parking lots. Image processing steps include noise reduction with Gaussian blurring, median filtering and thresholding to extract relevant features and enhance the quality of input images. Morphological operations such as dilation are then applied to refine the detected contours using OpenCV technology.

The developed system is compared with a car detection system that uses Google's TensorFlow technology and a CNN machine learning classifier, which involves a trained model on parking lot images. The systems have been compared with two real-world parking lot video streams containing 400 and 679 frames on a selected parking space in the parking lot.

The successful car detection rate of the developed system is found 25,50% higher than the Google's TensorFlow classification system for the first video stream containing 400 frames. Then, the successful car detection rate of the developed system is found 10,61% higher than the Google's TensorFlow classification system for the second video stream containing 679 frames. In addition, the developed system runs faster and requires less disc space usage than the Google's TensorFlow classification system.

Moreover, the successful care detection rate of the developed system is compared with another classification system using YOLOv8 technology based on CNN machine learning classifier. The successful car detection rate of the developed system is found 2,75% higher than YOLOv8 classification system for the first video stream containing 400 frames. Then, the successful car detection rate of the developed system is found 2,66% higher than YOLOv8 classification system for the second video stream containing 679 frames. In addition, the developed system runs faster and requires less disc space usage than the YOLOv8 classification system.

Overall, this study contributes to the advancement of intelligent parking systems by providing a scalable and reliable solution for real-time parking space detection.

Keywords: *Parking lots, Empty park space, Car detection, İmage processing, İmage filtering.*

1. GİRİŞ

Günümüzde kentlerin hızla artan nüfusuyla birlikte park yeri bulma sorunu giderek daha önemli bir konu haline gelmektedir. Özellikle yoğun şehirlerde, park yeri bulmak sürücüler için zaman alıcı ve stresli bir deneyim olabilmektedir. Bu sorunun çözümüne yönelik olarak, park yerlerinin boş veya dolu olduğunun gerçek zamanlı olarak tespit edilmesi ve sürücülere bilgi sağlanması, trafik yönetimini ve sürüş deneyimini iyileştirmek için önemli bir adım olabilir.

Son yıllarda yapılan güncel araştırmalar, kentleşmenin hızla artmasıyla birlikte özellikle büyük şehirlerde park yeri bulmanın ciddi bir sorun haline geldiğini ortaya koymaktadır. Bu durum yalnızca bireysel düzeyde zaman kaybı ve stres yaratmakla kalmayıp, aynı zamanda kent içi trafik yoğunluğunu artırmakta ve çevresel etkileri de beraberinde getirmektedir. INRIX tarafından 2024 yılında yayımlanan küresel trafik raporuna göre, sürücüler yılda ortalama 43 saati trafik sıkışıklığı nedeniyle kaybetmekte ve bu durum kişi başına yaklaşık 771 dolarlık bir maliyete neden olmaktadır [1]. 2025 yılına yönelik projeksiyonlar ise, akıllı park yeri yönetim sistemlerinin şehir içi trafik yoğunluğunu önemli ölçüde azaltabileceğine işaret etmektedir [2].

Bu bağlamda, kent yaşamında giderek önem kazanan bu sorun hem toplumsal hem de akademik düzeyde çözüm arayışlarını önemli kılmaktadır. Bu çalışmanın temel motivasyonu; kent içi ulaşımı kolaylaştırmak, sürücülerin zaman ve enerji kayıplarını minimize etmek ve şehir planlamasına veri temelli yaklaşımlar sunmaktır. Böylece, bu çalışma yalnızca akademik bir araştırma olmanın ötesinde, şehir yaşamının kalitesini artırmaya yönelik somut ve uygulanabilir çözümler geliştirmeyi hedeflemektedir.

Görüntü işleme teknikleri, son yıllarda çeşitli endüstrilerde birçok uygulama alanı bulmuştur. Özellikle, trafik yönetimi, güvenlik sistemleri ve otomotiv endüstrisi gibi alanlarda, görüntü işleme tekniklerinin kullanımı artmaktadır. Bu teknikler, video kameralar aracılığıyla elde edilen görüntüler üzerinde çeşitli işlemler yapılarak

bilgi çıkarmayı sağlarlar. Bu bağlamda, park yeri tespiti gibi uygulamalar, görüntü işleme tekniklerinin kullanımıyla gerçek zamanlı olarak yapılabilmektedir.

Bu çalışma, görüntü işleme teknikleri kullanılarak gerçek zamanlı park yeri tespiti üzerine odaklanmaktadır. Çalışma kapsamında, park yerlerinin dolu veya boş olduğunun tespiti için kullanılabilecek çeşitli görüntü işleme algoritmaları incelenmiş ve bu algoritmaların gerçek zamanlı bir sistemde nasıl uygulanabileceği araştırılmıştır. Ayrıca, önerilen yöntemlerin verimliliği ve uygulanabilirliği çeşitli deneyler ve karşılaştırmalar yoluyla değerlendirilmiştir.

Bu çalışmanın temel amacı, gerçek zamanlı çalışarak park yerlerinin durumunu tespit edip ve sürücülere park yerlerinin durumu hakkında anlık bilgi sunacak bir sistem geliştirmektir. Bu amaca ulaşmak için, çeşitli görüntü işleme algoritmaları ve teknikleri kullanılarak park yerlerinin boş veya dolu olduğunun tespit edilmesi hedeflenmiştir. Bu tespit işlemi, kameralar aracılığıyla elde edilen video akışlarından elde edilen görüntüler üzerinde gerçek zamanlı olarak gerçekleştirilmiştir.

Ayrıca bu çalışma, park yeri tespiti üzerine yapılan araştırmalara bir katkı sağlamak ve görüntü işleme tekniklerinin pratik uygulamaları için bir temel oluşturmak amacıyla da yapılmıştır. Bu çalışmanın önemi, aşağıdaki maddeler altında değerlendirilebilir:

Trafik Yönetimi: Gerçek zamanlı olarak park yerlerinin durum tespitinin yapılması, otoparklarda bulunan park yerlerinin durumu hakkında sürücülere bilgi sağlayarak trafik akışını düzenlemeye ve trafik yoğunluğu azaltmaya yardımcı olabilir.

Sürücü Konforunu Artırma: Sürücülerin park yeri arama sürecini ve stresini azaltarak sürüş deneyimlerini iyileştirebilir.

Şehir Planlaması ve Ulaşım Politikaları: Elde edilen veriler, şehir planlaması ve ulaşım politikalarının geliştirilmesinde kullanılabilir. Park yeri talepleri ve kullanımı hakkında daha fazla bilgi sağlayarak, şehirlerin altyapısının daha etkili bir şekilde yönetilmesine katkıda bulunabilir.

Çevresel Etkilerin Azaltılması: Park yeri arama sürecinin azaltılması, otoparklardaki trafik sirkülasyonunu ve buna bağlı olarak hava kirliliğini azaltabilir, böylece çevresel etkileri en aza indirebilir.

Sonuç olarak, bu çalışma, şehir yaşamını iyileştirmeye yönelik pratik çözümler sunma potansiyeline sahiptir. Bu nedenle, trafik yönetiminin iyileştirilmesi konusuna önemli bir katkı sağlayabilir.

Bu çalışmanın ikinci bölümünde, konuya ilişkin daha önce yapılmış çalışmalar, araştırmalar ve yaklaşımlar incelenerek literatür taraması sunulmuştur. Üçüncü bölümde ise çalışmada kullanılan materyaller tanıtılmış, veri toplama yöntemleri ile analiz süreçleri detaylandırılmıştır. Dördüncü bölümde elde edilen bulgular sunulmuş ve bu bulgular ilgili literatür çerçevesinde tartışılmıştır. Son olarak beşinci bölümde, çalışmanın genel sonuçları özetlenmiş, elde edilen bulgular doğrultusunda çeşitli önerilerde bulunulmuştur.

2. LİTERATÜR ARAŞTIRMASI

Bilimsel çalışmaların sağlam temellere dayanabilmesi için, öncelikle ilgili alanda daha önce yapılmış çalışmaların kapsamlı bir biçimde incelenmesi gerekmektedir. Literatür araştırması, mevcut bilgi birikimini ortaya koyarak çalışmanın ana çerçevesini oluşturur ve araştırmanın özgün yönlerini belirlemeye yardımcı olur. Bu bölümde, çalışma konusu ile doğrudan ya da dolaylı olarak ilişkili olan kaynaklar incelenmiş, önceki çalışmaların bulguları, kullanılan yöntemler ve ulaşılan sonuçlar ele alınmıştır. Böylece, bu çalışmanın mevcut literatindeki yeri ve katkısı daha açık bir şekilde ortaya konulmuştur.

Görüntü işleme, otomatik sistemlerin geliştirilmesi ve çeşitli uygulamalarda kullanılabilmesi açısından önemli bir alanı temsil etmektedir. Park yeri tespiti, bu tekniklerin uygulama alanlarından biridir ve son yıllarda araştırmacılar tarafından yoğun bir ilgiyle ele alınmıştır. Bu bölümde, mevcut literatürdeki önemli çalışmalara ve yöntemlere odaklanılarak, boş park yeri tespiti konusundaki gelişmeler ve çalışmalar özetlenmiştir.

Görüntü işleme tekniklerinin, bu alanda sağladığı yenilikçi çözümler, akıllı park sistemlerinin geliştirilmesi açısından büyük bir potansiyele sahiptir. Bu çalışmada, ortaya konan literatürden yola çıkarak, daha etkin bir park yeri tespit sistemi geliştirilmesi hedeflenmiştir.

Park yerlerinin dolu veya boş olduğunun tespiti için ağırlıklı olarak iki tür yaklaşım kullanılmaktadır. Bu yaklaşımlardan biri, temelinde donanım olarak sensörlerin kullanıldığı ve sensörlerden gelen bilgiler ile karar veren sistemlerin kullanılmasıdır. Diğer bir yaklaşım ise donanım olarak kameraları kullanan ve bu kameralardan elde edilen video akışlarındaki görüntüleri arkaplanda bilgisayarların ve algoritmaların yardımı ile işleyip karar veren sistemlerin kullanılmasıdır.

2.1 Sensör Kullanılarak Kurulan Sistemler

Park yerlerinin dolu veya boş olduğunu tespit etmede kullanılan yöntemlerden biri, sensörler vasıtasıyla bu tespitin yapılmasıdır [3]. Park yerlerinin durumunu tespit eden bu sistemler, kendi iç mimarisinde bir veya birden fazla sensör kullanılarak geliştirilmiştir.

Bu sistemlerin birçoğu, mimarilerinde kullanılan sensörlerin nitelikleri sebebiyle, çok sayıda donanımın park yerlerine kurulumunu ve bakımını zorunlu kılmaktadır. Örnek verilecek olursa, bu tarz sistemler, her bir park yerine en az bir sensör kurmayı zorunlu kılar. Bu durum, çok kapsamlı otoparklar için maliyeti ciddi ölçüde artırır. Çizelge 2.1'de yaygın sistemlerde donanım olarak kullanılan sensörler ve bu sensörlerin çalışma prensibinde kullandığı yöntemler gösterilmiştir.

Çizelge 2.1: Park Yerlerinin Durum Tespitinde Kullanılan Sensörler

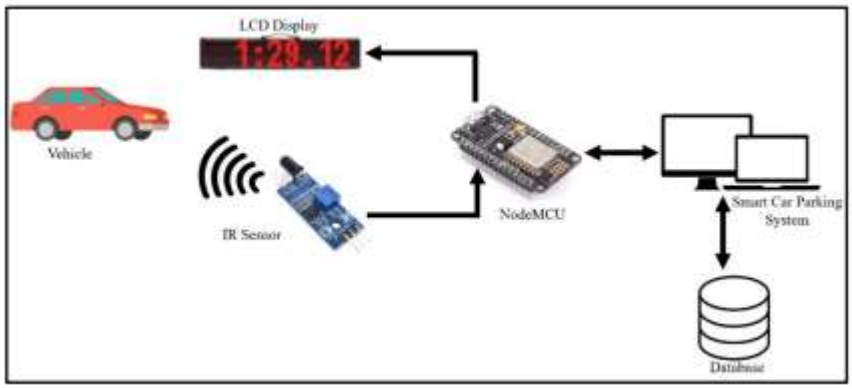
Yöntem	Sensör
Işık	Kızılötesi Sensörler
Mekanik	Pnömatik Yol Tüpü
Mekanik	Araç Kantarı
Mikrodalga	Mikrodalga Radar Sensörü
Manyetik	Endüktif Döngü Dedektörleri
Ultrasonik	Ultrasonik Sensörler

Kaynak: Revathi ve Dhulipala. (2012)

Kızılötesi Sensörler

İlk olarak, sürücü boş bir park yerine girdiğinde kızılötesi sensör aracın varlığını tespit eder. Kızılötesi sensör aracın varlığını tespit ettikten sonra, mikroişlemci LCD ekranda kırmızı ışık yakar. Bir akıllı sistem aracılığı ile veritabanına park yerinin dolu olduğu bilgisi gönderilir.

Araç park yerinden ayrıldıktan sonra ise, mikroişlemci kızılötesi sensörden aldığı yeni bilgiyle birlikte LCD ekranda yeşil ışık yakar ve park yerinin boş olduğu bilgisi veritabanına gönderilir. Bu sayede kontrol odasındaki personel veritabanından bilgiyi okuyarak, otoparktaki sürücüler ise LCD ekranların rengine bakarak hangi park yerlerinin boş veya dolu olduğu bilgisini öğrenebilir [4].



Şekil 2.1: Kızılötesi Sensör Tabanlı Otopark Yönetim Sistemi Mimarisi

Kaynak: Kadir, Osman, Othman ve Sedek. (2020)

Pnömatik Yol Tüpü

Araçlar, hava basıncıyla algılanır. Araçlar tüpün üzerinden geçtiğinde bir switch kapatılır ve sinyal üretilir [5]. Uygulaması ve bakımı kolaydır ancak sıcaklığa karşı duyarlıdır. Araç ilk kez tüpün üzerinden geçtiğinde park yerinin dolu olduğu bilgisi sunucuya ve veritabanına gönderilir; araç tekrar üzerinden geçtiğinde ise park yerinin boş olduğu bilgisi gönderilir. Doğruluk oranı düşüktür.

Şekil 2.2: Pnömatik Yol Tüpü

Kaynak: Shaheen, Rodier ve Eaken. (2005)

Araç Kantarı

Araç kantarı ile donatılmış park yerine giren cismin ağırlığı, kantar tarafından ölçülür. Eğer bu değer bir aracın ağırlığı kadar ise park yerinin dolu olduğu bilgisi sunucuya ve veritabanına gönderilir, aksi durumda ağırlık sıfır veya bir aracın olabilecek ağırlığından düşük bir değer ise park yerinin boş olduğu bilgisi gönderilir.

Her bir park yerine araç kantarı yerleştirilmesi, büyük ölçekli otoparklar için aşırı maliyetli bir durumdur. Bununla beraber ticari ve binek otomobillerin ağırlık

ayrımının yapılması zor bir durum olduğundan tek tip araç girişine izin verilen otoparklar için kullanılabilir.

Şekil 2.3: Araç Kantarı

Kaynak: Shaheen, Rodier ve Eaken. (2005)

Mikrodalga Radar Sensörü

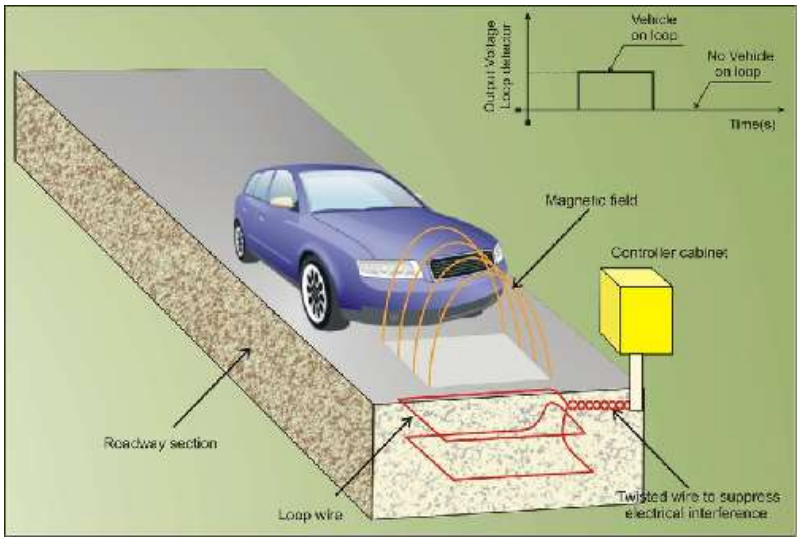
Mikrodalga radar sensörleri, enerjiyi bir anten aracılığıyla iletir ve antene geri yansıyan enerji sayesinde bir aracı tespit eder. İki tip mikrodalga radar sensörü vardır. Bunlar sürekli mikrodalga radarı ve frekans modülasyonlu sürekli mikrodalga radarlarıdır. Duyarsızdırlar ve duran aracı tespit etmek için yardımcı sensörler ile donatılmaları gerekir [5]. Bu durum ekstra bir maliyet oluşturur.

Şekil 2.4: Mikrodalga Radar Sensörü

Kaynak: Shaheen, Rodier ve Eaken. (2005)

Endüktif Döngü Sensörleri

Endüktif döngü sensörleri, frekansları 10 ila 50 kHz arasında değişen sinyallere çıkan çeşitli boyutlarda tel döngülerdir [6]. Manyetik alan, park yerine denk gelecek şekilde konumlandırılır. Endüktif döngünün salınım frekansı, aracın varlığıyla değişir ve bu sayede park yerinin dolu veya boş olduğu bilgisi sunucuya ve veritabanına gönderilir.

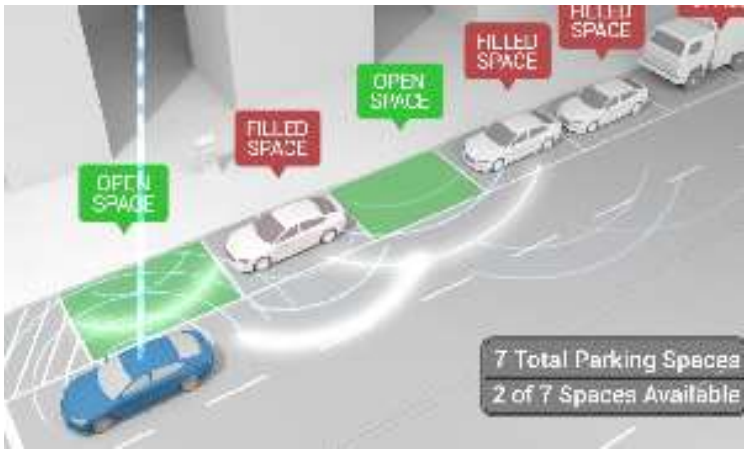


Şekil 2.5: Endüktif Döngü Sensörü Çalışma Prensibi

Kaynak: Mircea ve Nicolae. (2014)

Ultrasonik Sensörler

Ultrasonik sensörler, frekansı 25 ila 50 kHz arasındaki ultrasonik dalga biçimlerini park yerine iletir ve sensöre geri yansıyan enerjiler aracılığıyla park yerinde araç olup olmadığını tespit eder. Kurulumu kolaydır, ancak çevreye karşı duyarlıdırlar. Sensörlerin kurulumu genelde kaldırımların yola bakan kısımlarına yapıldığı için çevresinde kaldırım veya sensörün yerleştirilebileceği bir alan olmayan park yerleri için uygulanması zordur.



Şekil 2.6: Ultrasonik Sensörler ile Park Yeri Durum Tespit Sistemi

Kaynak: Braibanti (2017)

2.2 Algoritma Kullanılarak Kurulan Sistemler

Bu bölümde, algoritmalar ve kamera sistemleri kullanılarak gerçek zamanlı boş park yeri tespiti üzerine yapılan önceki çalışmaların incelenmesi ve analizi yapılmıştır.

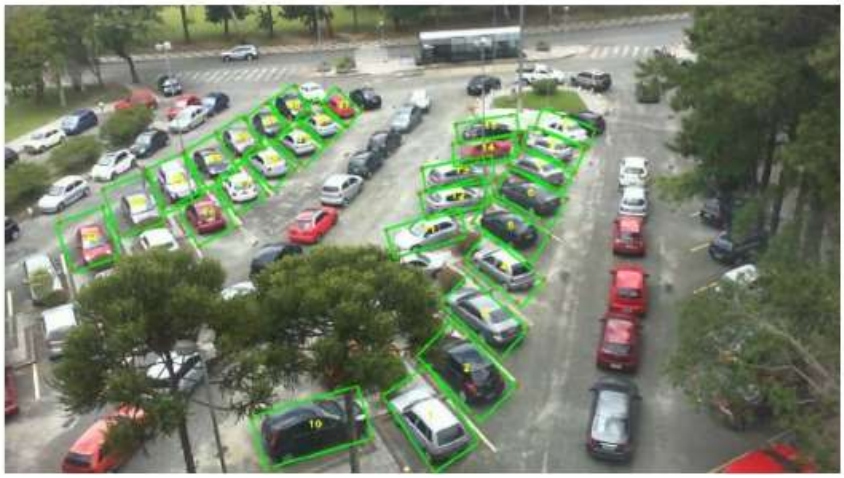
Sensörlere kıyasla, ideal pozisyon ve yükseklikte kuş bakışı olarak konumlandırılmış kameralar birçok park yerini aynı anda gerçek zamanlı bir şekilde gözetleyebilmektedirler. Kameralı sistemler, kurulumu ve bakımı sensörlü sistemlere kıyasla daha az maliyetlidir. Düşük maliyetleri sebebiyle, bu sistemler günümüzde otoparklarda yaygınlaşmaya ve sensörlerin yerini almaya başlamışlardır. Park yerlerinin doluluğunun tespit edilmesinde çeşitli makine öğrenmesi ve görüntü işleme metodları kullanılmaktadır.

Şekil 2.7: Otoparka Kuş Bakışı Konumlandırılmış Kamera Görüntüsü

Kaynak: Pngtree (2023)

Kameralı sistemlerde, park yerlerini gözeten çok sayıda veya tek bir adet kamera kullanılabilir. Bu sayede, kameralardan elde edilen görüntülerdeki park yerlerinin dolu veya boş olarak sınıflandırılması yapılabilir.

Bunun için ilk olarak otoparktaki durum tespiti yapılması istenen bütün park yerlerini içeren bir görüntüdeki park yerleri, kullanıcı veya sistem yöneticisi tarafından işaretlenir [9]. İşaretlenen park yerlerinin konumu, sonrasında tespit algoritması tarafından kullanılır.



Şekil 2.8: Park Yerlerinin İşaretlenmesi

Kaynak: Almeida, Oliveira, Britto Jr., Silva Jr., ve Koerich. (2016)

a) Sol Otopark Alt Kümesi b) Sağ Otopark Alt Kümesi

Şekil 2.9: Park Yerlerinin İşaretlenmesi

Kaynak: Amato ve diğerleri. (2016)

İşaretleme dikdörtgen veya kare şeklinde yapılır. İşaretli alanın içerisindeki görüntünün bir araç içerip içermediği, algoritmanın yapacağı sınıflandırma ile belirlenir. İşaretlenen park yerlerindeki araç varlığını tespit etmek için kullanılan algoritmada, bir takım makine öğrenmesi ve görüntü işleme tekniklerinden faydalanılır [10].

Park yeri durum tespiti, görüntü işleme tekniklerinin kullanıldığı karmaşık bir süreçtir. Bu süreç, park yeri sınırlarının belirlenmesi, park yerlerinin bölütlenmesi ve boş olup olmadığının sınıflandırılması gibi adımları içerir. Bu bölümün devamında, park yeri durum tespiti için bahsedilmiş önceki çalışmalarda yaygın olarak kullanılan bazı görüntü işleme teknikleri detaylı olarak açıklanmıştır:

Renk Bölütleme

Renk bölütleme, bir görüntüdeki farklı renk bölgelerini belirlemek için kullanılan bir tekniktir. Park yeri durum tespitinde, genellikle zeminin rengini belirlemek için renk bölütleme kullanılır. Örneğin, bir park yerinde zeminin genellikle gri veya siyah renkte olması beklenir. Bu nedenle, bu renklere yakın pikselleri belirleyerek görüntüdeki zemin alanları belirlenebilir.

Kenar Tespiti

Kenar tespiti, görüntüdeki keskin geçişleri veya farklılıkları tespit etmek için kullanılan bir tekniktir. Park yeri çizgileri genellikle bir araç ve zemin arasındaki net geçişlerdir, bu nedenle kenar tespiti bu çizgilerin tespitinde kullanılabilir.

Hough Dönüşümleri

Hough dönüşümleri, düz çizgilerin tespiti için yaygın olarak kullanılan bir tekniktir. Park yerlerinin sınırlarının genellikle düz çizgilerle tanımlanabileceği düşünüldüğünde, Hough dönüşümleri park yeri sınırlarını belirlemede kullanışlı olabilir.

Nesne Tanıma

Nesne tanıma teknikleri, belirli nesnelerin tanınması ve sınıflandırılması için kullanılır. Araç varlığının tespit edilmesi için nesne tanıma teknikleri kullanılabilir. Böylece park yerlerinin boş veya dolu olduğu belirlenebilir.

Bölgesel İşleme

Bölgesel işleme teknikleri, bir görüntüde belirli bölgelerin veya alanların işlenmesini sağlar. Park yeri durum tespiti için, öncelikle görüntüdeki park yeri alanlarının belirlenmesi ve işaretlenmesi gerekir. Bu alanlarda daha sonra ayrıntılı işlemlerin yapılabilmesi için bölgesel işleme teknikleri kullanılabilir.

Park yeri durum tespiti için kullanılan bu görüntü işleme teknikleri, genellikle bir araya getirilerek daha etkili ve güvenilir bir sistem oluşturulur. Ancak, her bir tekniğin kendi zorlukları ve sınırlamaları bulunur. Bu nedenle, kullanılacak teknikler uygulama bağlamına göre ideal bir şekilde seçilmelidir.

2.2.1 Statik görüntülerde boş park yeri tespiti

Statik görüntülerde boş park yeri tespiti çalışmasında, renk histogram sınıflandırıcısı ve Harris köşe tespiti teknikleri bir arada kullanılmıştır. Renk histogram sınıflandırıcısı her bir park yerinin ROI'sini alır ve renk alanını RGB'den, CIELAB renk uzayına dönüştürür. Parlaklık kanalını devre dışı bırakır ve yalnızca renklilik kanallarını korur. Bu sayede görüntüler genellikle ışığa tolare hale gelir. Daha sonra her bir park yerinin alt görüntüsü için bir renk histogramı oluşturulur.

Bu renk histogramı daha sonra ya bir destek vektör makinesi ya da k-en yakın komşu sınıflandırıcısı kullanılarak sınıflandırılır [11]. Bu teknik ile araçların dış özellikleriyle uyumlu şekiller görüntüde bulunmaya çalışılır. Bakılacak olan alt görüntünün renk kanalları değiştirilir ve bu sayede görüntüye vuran ışığın kötü etkisi ortadan kaldırılmış olur.

Renk histogram sınıflandırmasına paralel olarak bir araç özellik noktası tespit algoritması çalışır. Bu algoritma her bir park yerinin alt görüntüsünü alır ve o görüntüde bulunan özelliklerin türüne ve sayısına göre park yerini boş veya dolu olarak sınıflandırır.

İlk adım, park yerlerinin olduğu alt görüntülerde Harris köşe tespiti kullanarak ilgi noktalarını tespit etmektir. Algoritma, Forstner köşe tespiti yerine Harris köşe tesptini kullanmaktadır. Çünkü Forstner köşe tespiti, eğitim görüntülerinde çok fazla ilgi noktası tespit edememektedir. Eğitim sırasında algoritma, araç özelliklerine ve diğer araç olmayan cisimlerin özelliklerine ilişkin bir kelime dağarcığı oluşturur [11].

Şekil 2.10: Harris Köşe Tespiti ile İlgi Noktası Tespiti

Kaynak: True (2007)

Şekil 2.11: Dolu Bir Park Yerinden Alınan Örnek Özellik Blokları

Kaynak: True (2007)

Sonraki aşamada, algoritma her özellik noktasında ortalanmış 25x25 boyutunda küçük görüntüler oluşturur. Bu küçük görüntüler, mevcut görüntü karesindeki araçları tespit etmek için algoritmanın kullandığı özelliklerdir. Bu özellik sınıflarının varlığı veya yokluğu daha sonra test görüntüsünde hedef objenin yani araçların tespit edilmesinde ve konumunun belirlenmesinde kullanılır [11].

Bireysel özelliğin sınıf tahmini daha sonra hedef park yerinin nihai sınıflandırılmasında boş veya dolu olarak oy verir [11]. Bir araç, bir park yerinin çoğunu kaplayabilir ve bu durumda sınıflandırma oylaması yapmak gereksiz gibi görülebilir.

Ancak bardak gibi çeşitli nesneler, yağ lekeleri sıklıkla park yerlerinde görülebilir ve oylama yapılmaması durumunda bu sistemin bozulmasına neden olabilir. Şekil 2.12'de alt ve orta kısımdaki iki görüntüde bir karton bardak bulunmaktadır. Park yerinin ROI'si içinde kaldığı için bir araba özelliği olarak yanlış sınıflandırılmıştır.

Şekil 2.12: Yanlış Sınıflandırma Örneği

Kaynak: True (2007)

Algoritma kullanılarak yapılan bu çalışmada, açık alanlardaki park yeri doluluğunu belirlemek için önceden eğitilmiş bir CNN tabanlı model tarafından çıkarılan özelliklerin, bir SVM sınıflayıcı içinde kullanılmasıyla görüntü tabanlı bir yazılım çerçevesi geliştirilmiştir.

Çerçeve, eğitim veri setinde %99,7 gibi yüksek bir doğruluk oranı elde etmiş ve bağımsız bir test veri setinde öğrenme doğruluğunun oranı %96,6 olarak belirlenmiştir.

Ancak, öğrenme performansını sınırlayan bazı zorluklar bulunmaktadır. Bunlar arasında, binaların ve cisimlerin park yerleri üzerindeki gölgeleri, araçlardan kaynaklanan güçlü güneş yansımaları, kullanıcı veya sistem yöneticisi tarafından belirlenen park alanların dışına veya aralarına park edilmiş araçlar ve kullanılan eğitim verisinin kalitesi yer almaktadır [12].

Şekil 2.13: 30 Adet Park Yerinin İşaretlenmesi

Kaynak: Acharya, Yan ve Khoshelham. (2018)

a) Doluluk Tespiti Yapılmadan Öncesi

b) Doluluk Tespiti Yapıldıktan Sonrası

Şekil 2.14: 30 Adet Park Yerinin Doluluk Tespiti

Kaynak: Acharya, Yan ve Khoshelham. (2018)

2.2.2 Geliştirilmiş MobileNetV3 ile otopark doluluk algılama

Geliştirilmiş MobileNetV3 ile otopark doluluk algılama çalışmasında, mimarisinde yapılan çeşitli değişikliklerle güçlendirilen ve doğruluğunu artıran bir CNN sınıflandırma modeli olan MobileNetV3 kullanılarak bir park yeri durum tespit sistemi geliştirilmiştir. Geliştirilen sistem, PKLot ve CNRPark-EXT gibi tanınmış iki adet otopark veri seti üzerinde eğitilmiştir [13].

Sisteme girdi olarak verilen video akışı, kare kare işlenmekte ve her bir kare, parçalara bölünmektedir. Modifiye edilmiş MobileNetV3 modeli, her bir parçada

bulunan park yerini dolu ya da boş park yeri olarak sınıflandırmaktadır. Sınıflandırma sonuçları, her park yerinin etrafına çizilen sınır kutularıyla birlikte mevcut görüntü karesine çizilmiştir [13].

Geliştirilen sistemin performansı, diğer yerleşik sınıflandırma modelleri ile deneysel olarak karşılaştırılmıştır. Değerlendirme ve deneysel sonuçlar, geliştirilmiş MobileNetV3 modelinin yüksek doğruluk oranı elde ettiğini ve hem doğruluk hem de hız açısından diğer sınıflandırma modellerini geride bıraktığını ortaya koymuştur [13].

Şekil 2.15: Modifikasyonlu MobileNetV3 ile Geliştirilen Sistem

Kaynak: Yuldashev, Mukhiddinov, Abdusalomov, Nasimov ve Cho. (2023)

2.2.3 Maskeli bölge tabanlı evrişimsel sinir ağı kullanılarak etkili görüntü tabanlı park yeri doluluk tespiti

Maskeli bölge tabanlı evrişimsel sinir ağı kullanılarak etkili görüntü tabanlı park yeri doluluk tespiti çalışmasında, derin öğrenme tekniklerini ve ağlarını kullanarak iç ve dış mekân park yeri durum tespiti problemleri için dinamik bir algoritma geliştirilmiştir.

Çalışmada bulunan algoritma, park yerindeki aracın tespiti için örnek bir segmentasyon kullanan, Mask-RCNN algoritması ile başarılı bir şekilde uygulanmıştır. IoU tekniğini kullanarak, algoritma boş ve dolu park yerlerini tespit edebilmiştir [14].

Nesne tespiti yapan Mask-RCNN, 2000 görüntüden oluşan bir otopark veri setinde hiperparametre optimizasyonu gerçekleştirdikten sonra %91 doğruluk oranı ile araç tespiti yapabilmiştir [14].

Algoritma, Mask-RCNN modeli tarafından tespit edilen araç üzerinden IoU hesaplaması yaparak dolu ve boş park yerlerini sınıflandırabilmiştir. Algoritmanın sınıflandırma sonuçları, dolu ve boş olarak etiketlenmiş park yeri görüntüleri ile karşılaştırılmıştır. Başarı oranına dayanarak algoritmanın iyi bir performans sergilediği gösterilmiştir.

Ancak, sonuçlardaki tespit uyumsuzluğuna neden olabilecek bazı sınırlamalar bulunmaktadır. Bunlar arasında, görüntüde bir nesnenin görünürlüğünü engelleyen örtüşme ve bir nesnenin diğerinin üzerinde bulunması gibi durumlar yer almıştır.

Ayrıca sonuçlar, hiperparametre optimizasyonuna rağmen, algoritmanın ortalama bir performans sergilediğini ve bu uygulamanın gerçek dünya senaryolarına tam olarak entegre edilebilmesi için daha fazla keşif ve değişiklik gerektirdiğini göstermektedir [14].

a) Sol Otopark Alt Görüntüsü b) Sağ Otopark Alt Görüntüsü

Şekil 2.16: Mask-RCNN Algoritması Kullanan Sistemin Tespitleri

Kaynak: Skariah (2022)

2.2.4 Otopark doluluk bilgi sistemi

Bir başka çalışmada, COINS (Otopark Doluluk Bilgi Sistemi) adında gelişmiş bir park yeri tespit sistemi sunulmuştur. Çalışmada, sürücülerin boş park yerlerini verimli bir şekilde bulmalarına yardımcı olmak amacıyla entegre bir görüntü işleme yaklaşımı kullanılmıştır. Çalışmada, özellikle park yerlerinin sınırlı olduğu yoğun

saatlerde veya otoparkların neredeyse dolu olduğu durumlarda, sürücülerin park yeri arama süresini azaltmalarına yardımcı olmak amaçlanmıştır [22].

COINS, görüntü işleme teknikleri kullanarak park yerlerinin doluluk durumlarını tespit eder ve izler. Sistem aşağıdaki birkaç ana bileşenden ve süreçten oluşur:

Başlatma Süreci

Sistem, başlatma işlemi için manuel işaretleme prosedürü gerektirir. Bu, kullanıcının veya sistem yöneticisinin otoparktaki park yerlerini analiz etmesi ve işaretlmesi için sisteme başlangıç girdisi sağlanması adımıdır.

Sınır Arama

Başlatma sürecinden sonra, COINS, park yerlerinin kesin konumlarını belirlemek için görüntülerde bir sınır araması yapar. Bu arama, park yerlerini görsel bağlamda net bir şekilde tanımlamaya yardımcı olur. Sistem, park yerlerinin dolu olup olmadığını belirlemek için çoğunlukla görüntü farkı, nesne tespiti, kenar tespiti ve oylama algoritması kullanır.

Görüntü Farkı

Bu teknik, zaman içinde farklı görüntü karelerini karşılaştırarak değişiklikleri tespit eder ve park yerinin dolup dolmadığını gösterir.

Nesne Tespiti

Sistem, park yerlerindeki araçları tespit edebilir ve alanın dolu olup olmadığını belirleyebilir.

Kenar Tespiti

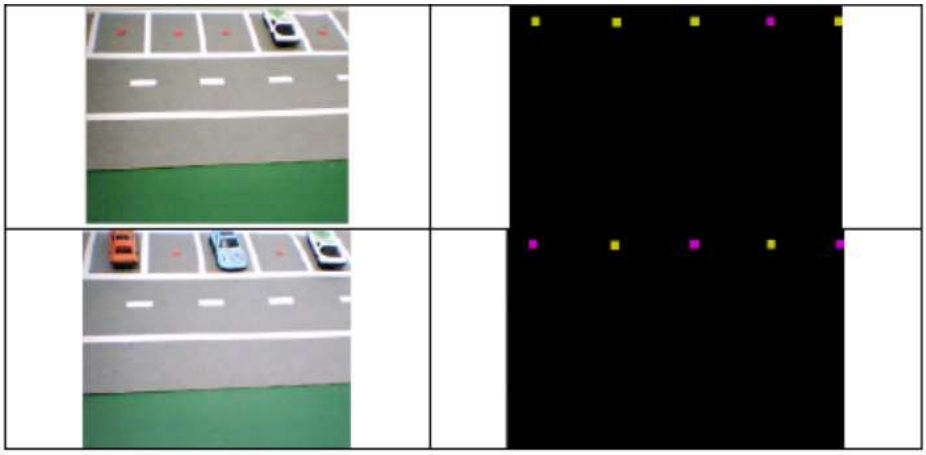
Kenar tespiti, park yerlerinin sınırlarını görüntülerde tanımlamaya yardımcı olur.

Oylama Algoritması

Oylama algoritması, sistemin çeşitli özellikleri ve oyları birleştirerek park yeri doluluğu hakkında nihai bir karar almasına yardımcı olur.

Yapılan deneyler ve testler sonucunda, sistemin farklı hava ve aydınlatma koşullarında performansı değerlendirilmiştir. Çünkü bu koşullar görüntü kalitesini ve tespit doğruluğunu etkileyebilir.

Sonuçlar, COINS'in çeşitli koşullar altında %93 oranında doğru tespit sağladığını ve park yerlerinin doluluk durumlarını başarılı bir şekilde tespit ettiğini göstermiştir.



Şekil 2.17: COINS Park Yeri Durum Tespiti

Kaynak: Bong, Ting ve Lai. (2008)

2.2.5 Görüntü işleme teknikleri ve kablosuz sensör ağı kullanılan akıllı park sistemi

Bu çalışmada, görüntü işleme tekniklerini ve kablosuz sensör ağlarını birleştirerek otoparklardaki park yerlerinin durumunu verimli bir şekilde izleyen akıllı bir park sistemi ortaya konmuştur [23].

Sistem, park yerlerinin görüntülerini yakalamak için kameraları kullanır. Bu kameralar, park yerlerinin her birinin durumunu izlemek için otoparklara stratejik olarak yerleştirilir.

Bir görüntü yakalandığında, sistem, bu görüntüyü analiz etmek ve park yerinde bir aracın olup olmadığını belirlemek için nesne tespit algoritmalarını kullanır. Arka plan çıkarımı ve hareket tespiti gibi yaygın teknikler, park yerlerindeki araçları tanımlamak için kullanılır [23].

Gelişmiş kenar tespiti teknikleri, park edilmiş araçların sınırlarını belirlemeye yardımcı olur. Sistem, ardından görüntü verilerini sınıflandırarak park yerinin boş mu yoksa dolu mu olduğunu belirler. Sistem, araçları ve boş park yerlerini ayırt ederek, park yerlerinin durumunu gerçek zamanlı olarak günceller.

Otoparktaki her park yerine, aracın varlığını tespit etmek için ultrasonik sensörler veya kızılötesi sensörler gibi kablosuz sensörler yerleştirilir. Bu sensörler,

park yerinin dolu olup olmadığını, sensör ile araç arasındaki mesafeyi ölçerek ve çevredeki ortam değişikliklerini tespit ederek belirler [23].

Bu sensörler, Wi-Fi, Zigbee veya Bluetooth gibi kablosuz protokoller kullanarak, merkezi bir sunucuya park yerlerinin doluluk durumuna dair verileri iletir. Bu kablosuz iletişim, sistemin ölçeklenebilir olmasını sağlamış ve genişlemeyi kolaylaştırmıştır.

Sistem hem görüntü işleme sisteminden hem de kablosuz sensörlerden gelen verileri birleştirir. Sensör verileri, park yerinin doluluk durumuna dair gerçek zamanlı bilgi sağlar, görüntü işleme sistemi ise her bir park yerinin durumunun daha doğru ve ayrıntılı bir şekilde doğrulanmasını sağlar.

Çalışmada, veri füzyon teknikleri, iki kaynaklı verilerin birleştirilmesi için uygulanmıştır. Böylece, sistem tutarsızlıkları ele alır ve park yerlerinin doluluk tespitinin doğruluğunu artırır. Bir sensör bir aracı tespit ettiğinde, sistem kamera görüntüleri ile park yerinin gerçekten dolu olup olmadığını doğrulamak için çapraz kontrol yapar [23].

Kameralar ve sensörlerden gelen tüm veriler, merkezi bir sunucuya gönderilir. Sonrasında, sunucu üzerinde park yerlerinin durumu işlenir ve analiz edilir.

Sunucu, sensörler ve kameralar arasındaki verileri toplar ve gerçek zamanlı bir gösterge paneli veya mobil uygulama üzerinden park yerlerinin doluluk durumu görüntülenir. Sürücüler bu bilgiyi dijital tabelalar veya mobil uygulamalar aracılığıyla kullanarak, boş park yerlerini hızlıca bulabilirler.

Sistem, sürücüler için kullanıcı dostu bir arayüz sağlar, böylece gerçek zamanlı park yeri durumu görüntülenebilir. Bu arayüz, sürücülere mevcut park yerlerini gösterir ve onları en yakın boş alana yönlendirir, böylece park yeri arama süresi azaltılmış olur.

Dijital tabelalar veya LED göstergeler, otoparklarda mevcut park yerlerinin durumunu gösterir ve mobil uygulamalar en yakın boş park yerine yönlendirme sağlar [23].

Yapılan deneyler ve testler sonucunda, sistemin park yerlerinin doluluk durum tespitinde %90'ın üzerinde doğruluk oranı elde ettiği görülmüştür.

Şekil 2.18: Görüntü İşleme Teknikleri ve Kablosuz Sensör Ağı Kullanılan Akıllı Park Sistemi

Kaynak: Idris, Tamil, Razak, Noor ve Kin. (2009)

2.2.6 Otoparkta görüntü işleme ile araçların tespiti

Bu çalışmada, otoparklarda bulunan kameralar tarafından çekilen görüntülerden bilgi çıkarmaya odaklanılmıştır [24].

Ham görüntüler, araçların şekillerini ve otopark düzenini vurgulayan özelliklerin belirginleşmesi için ön işlemeye tabii tutulmuştur. Bu ön işleme kontrast ayarı, görüntü düzleştirme ya da kenarları vurgulama gibi teknikler ile yapılmıştır.

Görüntü ön işlendikten sonra, sistem otoparktaki araçları tespit eder. Bu tespit, aracın sınırlarını belirlemek için kenar tespiti, araçlara ait bölgeleri tespit etmek için eşikleme ve araçların arka plandan ayırt edilmesi için renk segmentasyonu gibi tekniklerle yapılmıştır [24].

Çalışmada, şablon eşleştirme ve makine öğrenmesi tabanlı yöntemler kullanılarak araçların, ağaçlar, direkler ya da gölgeler gibi diğer nesnelerden ayırt edilmesi önerilmiştir. Araçlar, genellikle şekilleri, boyutları ve konumlarına göre tanınır.

Tespit edilen bölgelerin konturları ve sınırları analiz edilerek, bu bölgelerin içerisinde araç olma olasılığı belirlenir. Algoritma, görüntüdeki şekilleri, bilinen araç şablonları ve desenleriyle karşılaştırarak sınıflandırma yapar.

Araç tespit edildikten sonra, sistem aracın otoparktaki konumunu tam olarak belirler. Bu işlem, tespit edilen aracın görüntü koordinatlarını, otoparktaki belirli bir park yerine yerleştirmek için yapılır.

Ayrıca çalışmada, kameranın görüş açısındaki perspektif bozulmaları dikkate alınmıştır. Çünkü bu bozulma, araçların gerçek boyutlarını ve konumlarını etkileyebilir. Bu etkileri düzeltmek için homografi ve geometrik dönüşümler gibi teknikler uygulanmıştır.

Otoparklar, aydınlatma, hava koşulları ve engellenmeler gibi değişkenlere tabi olabilir. Çalışma, algoritmanın bu tür sorunlarla nasıl başa çıktığını açıklamaktadır. Algoritma, yalnızca tek bir görsel işarete değil, otoparkın genel yapısına ve araçların şekillerine odaklanan etkili görüntü işleme teknikleri kullanarak bu zorlukları aşmaktadır [24].

Sistem, arka plan çıkarımı kullanarak, araçları otopark zemini gibi statik arka plan elemanlarından ayırmayı ve alakasız nesneleri filtrelemeyi amaçlamıştır.

Çalışmada, araçları konumlandırırken karşılaşılan bazı zorlukları vurgulanmış ve bu sorunları aşmak için önerilen çözümler sunulmuştur.

Otoparklar, günün saati, gölgeler ve hava koşulları nedeniyle farklı aydınlatma koşullarına sahip olabilir. Sistemde, görüntü işleme algoritması, adaptif eşikleme ve dinamik renk modelleri gibi teknikler kullanılarak, değişen koşullarda doğruluğun korunması sağlanmıştır.

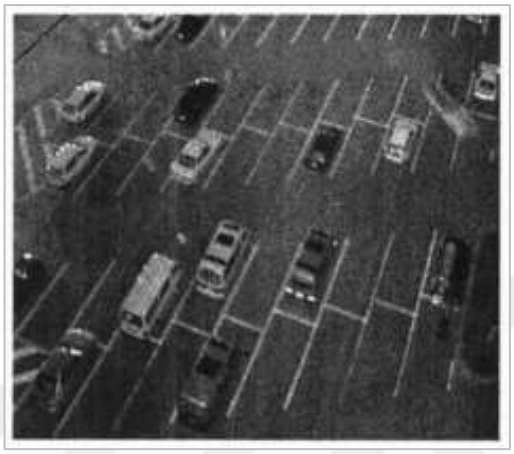
Araçlar bazen diğer araçları engelleyebilir, bu da engellenen araçları tespit etmeyi zorlaştırabilir. Sistem, engellenmiş araçların kısmi şekillerini tanıyarak bu durumu aşmaya çalışır. Ayrıca, istatistiksel modeller kullanarak, engellenen araçların varlığını, çevre boşluğu ve önceki verilerden tahmin etmeye çalışır.

Çalışmada, özellikle yoğun otoparklarda görüntülerin gerçek zamanlı işlenmesi gerektiği belirtilmiştir. Algoritmaların, gecikme olmadan araçları hızlı bir şekilde tespit etme ve konumlandırma yeteneğine sahip olması önemli olmuştur, bu da gerçek zamanlı otopark yönetimi sistemi için kritik bir gereklilik olmuştur [24].

Çalışmada, önerilen yöntemin etkinliğini doğrulamak için deneysel sonuçlar sunulmuştur. Bu deneyler, farklı ışık seviyeleri, farklı araç türleri ve kalabalık ya da boş otoparklar gibi farklı koşullarda algoritmanın test edilmesini içermektedir.

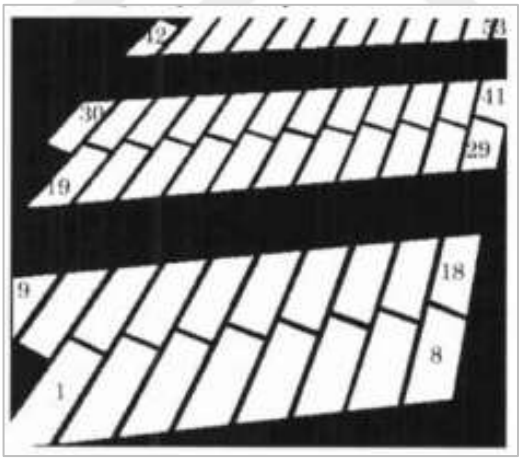
Sonuçlar, tespit doğruluğu, işlem hızı, engellenme ve çevresel faktörlere karşı dayanıklılık gibi metrikleri içermektedir.

Sistemin farklı saatlerde ve aydınlatma koşullarında performansı değerlendirilmiştir. Test ve deney sonuçları, sistemin logaritmik dönüşümsüz ortalama %96,2; logaritmik dönüşümlü ortalama %97,4 oranında doğru tespit sağladığını göstermiştir [24].



Şekil 2.19: Otoparkta Görüntü İşleme ile Araçların Tespiti Verisi

Kaynak: Hsu, Tanaka, Sugie ve Ueda. (2002)



Şekil 2.20: Otoparkta Görüntü İşleme ile Araçların Tespiti Sonucu

Kaynak: Hsu, Tanaka, Sugie ve Ueda. (2002)

2.2.7 Gözetim kamerası ve fcm sınıflandırıcısı kullanarak otoparkta boş park yeri tespiti

Bu çalışmada, dış mekân otoparklarında boş park yerlerini gözetim kameraları ve Fuzzy C-Means (FCM) sınıflandırıcısı kullanılarak verimli bir şekilde tespit etmeyi amaçlayan bir sistem ele alınmıştır [25].

Sistem, bir gözetim kamerası aracılığıyla otoparktan elde edilen video görüntü karelerini toplar. Bu kameralar, park yerlerinin üstten ve yan taraftan görüntülerini çeker. Görüntüler, park yerlerini tespit edebilmek için sonradan işlenir.

Çekilen video akışları, park yerlerinin diğer görüntü öğelerinden ayrılabilmesi için çeşitli görüntü işleme tekniklerine tabi tutulur. Bu adım önemlidir, çünkü kamera görüşünde, arka plan gürültüsü, ağaçlar, geçiş yapan araçlar ve diğer park yeri dışı alanları bulunabilir.

Arka plan çıkarma ve kenar tespiti gibi yöntemlerle park yerleri, çevrelerinden ayrılır. Sistem, aydınlatma, hava koşulları, gölgeler ve yansımalar gibi diğer çevresel faktörlerin görüntü kalitesini etkilemesini önlemiştir.

Sonraki aşamada, park yerlerinin çeşitli özellikleri çıkartılmıştır. Bunlar, park yerlerinin şekli ve boyutu, dolu bir park yeri ile boş bir park yeri arasındaki renk farkı gibi renk özellikleri, görüntüdeki park yerlerinin koordinatlarıdır. Bu özellikler, sistemin park yerlerini boş ve dolu olarak ayırt edebilmesi için gerekli olmuştur. Sistem, park yerlerindeki görsel ipuçlarını analiz eder ve bir aracın varlığını tanır.

Sonrasında FCM kümeleme işlemi yapılmıştır. Sistemin sınıflandırma yönteminin merkezinde baz olarak FCM algoritması kullanılmıştır. FCM, özellikle görüntü segmentasyonu için yaygın olarak kullanılan bir kümeleme tekniğidir.

FCM, denetimsiz bir makine öğrenimi algoritmasıdır ve verileri bulanık kümelere sınıflandırır. Geleneksel katı kümeleme yöntemlerinden farklı olarak, her veri noktası yalnızca bir kümeye ait olmak yerine, birden fazla kümeye farklı derecelerde ait olabilir. Bu sistemde, FCM, renk, şekil, pozisyon gibi çıkarılan özelliklere dayanarak park yerlerini iki kümeye ayırır. Bunlar, boş park yeri ve dolu park yeri kümeleridir.

FCM'in bulanık üyelikler kullanması, sistemin görüntü verilerindeki belirsizlikleri yönetmesini sağlamıştır. Örneğin, kısmen kapalı park yerleri, gölgeler veya geçici engeller gibi zorluklar altında doğru şekilde sınıflandırma yapabilir.

FCM kümeleme çıktısına dayanarak, her park yeri sınıflandırılır. Eğer bir park yeri boş park yeri kümesi ile yüksek üyelik gösteriyorsa, boş olarak sınıflandırılır ve işaretlenir. Eğer park yeri dolu park yeri kümesi ile yüksek üyelik gösteriyorsa, dolu olarak sınıflandırılır ve işaretlenir. Sistem, bu bilgiyi alır ve her bir park yerinin durumunu gerçek zamanlı olarak günceller.

Şekil 2.21: Gözetim Kamerası ve FCM Sınıflandırıcısı Kullanarak Otoparkta Boş Park Yeri Tespiti

Kaynak: Ichihashi, Notsu, Honda, Katada ve Fujiyoshi. (2009)

Çalışmada, karşılaşılan bazı zorluklar ve bunlarla nasıl başa çıkıldığı tartışılmıştır. Dış mekân otoparkları, gün boyunca farklı ışık koşullarına sahip olabilir. Sistem, görüntü normalizasyonu ve adaptif eşikleme gibi tekniklerle bu farklılıklarla başa çıkmıştır.

Otoparklarda, yayalar veya geçen araçlar gibi hareketli nesneler olabilir. Bunu yönetmek için, sistem hareket tespiti algoritmaları kullanarak alakasız nesneleri filtreler ve yalnızca park yerlerine odaklanır. Yağmur, kar veya sis, park yerlerini gizleyebilir. Sistem, zamanla hava koşullarına uyum sağlamak için makine öğrenimi modelleri kullanır ve eşikleri dinamik olarak ayarlar.

Çalışmada sistemin, gerçek dünyadan alınmış otopark görüntüleri üzerinde gerçekleştirilen deneyler ile performansı değerlendirilmiştir. Ayrıca, FCM tabanlı yaklaşım, geleneksel eşikleme veya şablon eşleştirme gibi yöntemlerle karşılaştırılmıştır. Sonuçlar, FCM sınıflandırıcısının, düşük ışık, kısmi engellemeler gibi zorlu koşullarda bile boş ve dolu park yerlerini yüksek doğruluk ve verimlikle tespit ettiğini göstermiştir.

2.2.8 Ölçeklenebilir gerçek zamanlı otopark sınıflandırması

Bu çalışmada, görüntü özellikleri ve denetimli öğrenme algoritmaları kullanılarak park yerlerinin boş veya dolu sınıflandırılması otomatik olarak yapılmıştır. Çalışmanın temel amacı, park yerlerinin durumunu doğru bir şekilde sınıflandırmak için farklı görüntü özelliklerinin ve makine öğrenme tekniklerinin değerlendirilmesi olmuştur [26].

Çoğu mevcut park yeri sınıflandırma sistemi küçük otoparklar üzerinde yoğunlaşırken, bu çalışma büyük ölçekli otoparklara uygulanabilir bir çözüm geliştirmeyi hedeflemiştir. Çalışmada, hangi görüntü özelliklerinin park yerlerini sınıflandırmak için en uygun olduğunu belirlemeye çalışılmıştır.

Yüksek çözünürlüklü görüntüler, otoparklara yerleştirilen kameralar aracılığıyla toplanmıştır. Bu görüntülerde hem boş hem de dolu park yerleri yer almıştır. Ayrıca görüntüler, sistemi eğitmek ve test etmek için de kullanılmıştır.

Görüntüler, ilk olarak normalizasyon, gürültü giderme ve segmantasyon gibi ön işleme adımlarından geçirilmiştir. Bu adım, sistemin daha net ve tutarlı verilerle çalışmasını sağlamıştır. Sonrasında, çeşitli görüntü özellikleri çıkartılmış ve bu özellikler park yerlerinin boş veya dolu olduğunu sınıflandırmak için kullanılmıştır.

Gabor filtreleri veya LBP gibi özellikler kullanılarak, görüntüdeki araç yüzeyi ile boş park yeri yüzeyi arasındaki farklar gibi dokusal farklar tespit edilmiştir. Renk histogramları hesaplanmıştır, böylece araçlar ile boş park yerleri arasındaki renk farkları belirlenmiştir. Park yerlerinin geometrik şekilleri tespit edilmiştir.

Bu işlem, kenar tespiti gibi yöntemler kullanılarak yapılmıştır. Özellikler çıkarıldıktan sonra, makine öğrenme modelleri kullanılarak park yerlerinin boş veya dolu olduğu sınıflandırılmıştır. Çalışmada aşağıdaki denetimli öğrenme algoritmaları karşılaştırılmıştır:

Destek Vektör Makineleri

SVM, çalışmada park yeri verilerini boş ve dolu gibi farklı sınıflara ayıran optimum hiperdüzlemi bulmaya çalışan güçlü bir sınıflandırma algoritması olmuştur.

Doğrusal Diskriminant Analizi

LDA, çalışmada özellikle sınıflandırma ve boyut indirgeme problemleri için kullanılan güçlü bir denetimli öğrenme algoritması olmuştur. Temelde, LDA'nın amacı farklı sınıfları en iyi şekilde ayırmak ve veriyi daha düşük boyutlara indirgemektir. Bu, özellikle çok sayıda özelliğe sahip veri setlerinde faydalıdır.

K-En Yakın Komşu

K-NN, çalışmada daha basit bir algoritma olmuştur ve bir veri noktasını, özellik uzayında en yakın K komşusunun çoğunluk sınıfına göre sınıflandırır.

Bu algoritmalar, etiketli veri setleri ile eğitilmiş ve daha sonra test veri seti ile test edilmiştir.

Şekil 2.22: Görüntü Özelliklerinin ve Denetimli Öğrenme Algoritmalarının Değerlendirilmesi

Kayank: Tschentscher, Koch, König, Salmen ve Schlipsing. (2015)

Çalışmada, algoritmaların çapraz doğrulama yöntemleriyle performansları değerlendirilmiştir. Performans, doğruluk, kesinlik, geri çağırma ve F1 skoru gibi metriklerle ölçülmüştür. Bu metrikler, her modelin boş ve dolu park yerlerini ne kadar doğru sınıflandırdığını göstermiştir.

Yapılan deneyler ve testler sonrasında, SVM algoritmasının en fazla %94,13, LDA algoritmasının en fazla %83,31 ve K-NN algoritmasının en fazla %93,58 doğruluk oranına sahip olduğu görülmüştür. Sistem, gerçek zamanlı görüntü verisi ile çalışacak şekilde tasarlanmıştır.

Yazarlar, çok sayıda park yeri için aynı anda işlem yapabilen ve her birini gerçek zamanlı olarak sınıflandırabilen bir çözüm sunduklarını göstermişlerdir.

Ayrıca çalışma, görüntü özellikleri ve denetimli öğrenme algoritmaları kombinasyonunun, gerçek zamanlı park yeri sınıflandırması için etkili bir çözüm sunduğunu vurgulamıştır.

2.2.9 Derin evrişimsel sinir ağları tabanlı park yeri boşluk göstergesi sistemi

Bu çalışmada, görüntü işleme ve CNN derin öğrenme kullanılarak, park yerlerinin boş veya dolu durumunu gerçek zamanlı video görüntüleriyle belirleyebilen bir sistem ortaya konmuştur. Sistemde, video kameralardan gerçek

zamanlı olarak gelen görüntüler kullanılarak park yerlerinin dolu veya boş olduğunu otomatik olarak tespit etmek amaçlanmıştır [27].

Sistem, otoparklarda kurulu kameralar tarafından elde edilen video akışlarını kullanır. Bu kameralar, belirli aralıklarla görüntüler sağlar ve sistem, her bir park yerinin boş mu yoksa dolu mu olduğunu belirlemek için bu görüntüleri işler.

Bu çalışmada kullanılan mimari, birkaç evrişim katmanından, havuzlama katmanlarından ve tam bağlı katmanlardan oluşarak, görüntülerden özellikler çıkarır ve nihai sınıflandırmayı yapar.

CNN ile kameralardan alınan ham görüntüler işlenir. Bu işlemde, görüntülerin yeniden boyutlandırılması, piksel değerlerinin normalize edilmesi ve veri artırma teknikleri kullanılmıştır.

Evrişim katmanları, görüntülerden kenar, doku ve şekil gibi hiyerarşik özellikler çıkarır. Bu özellikler, park yerlerinin boş mu yoksa dolu mu olduğunu belirlemede kullanılır. Ağ, her bir park yerini "boş" veya "dolu" olarak sınıflandırır. Nihai katman, bu sınıflandırmayı verir. Sistem, gerçek zamanlı olarak çalışacak şekilde tasarlanmıştır. Böylece, park yerlerinin durumu hızlıca tespit edilip, sürücülere anında bilgi sağlanabilir.

Şekil 2.23: Derin Evrişimsel Sinir Ağları Tabanlı Park Yeri Boşluk Göstergesi Sistemi

Kaynak: Valipour, Siam, Stroulia ve Jagersand. (2016)

Sistem, özellik mühendisliği yapmadan çalışır. Çünkü CNN, park yerlerinin durumunu belirlemek için ham görüntü verilerinden hangi özelliklerin önemli

olduğunu öğrenir. Bu durum, "sonuçtan sonuca öğrenme" tekniği uygulanabildiği şartlarda, sistemin daha esnek ve geleneksel yöntemlere göre daha kolay dağıtılabilir olmasını sağlamıştır.

Çalışmada, her bir park yerinin boş veya dolu olarak etiketlendiği bir otopark veri seti üzerinde CNN modeli eğitilmiştir. Modeli daha güçlü hale getirebilmek için görüntüler üzerinde çevirme, kırpma ve döndürme gibi çeşitli veri artırma teknikleri kullanılmıştır. Bu durum, modelin farklı koşullara karşı daha dayanıklı hale gelmesine yardımcı olmuştur.

Çalışmada, transfer öğrenmesi adı verilen bir yöntem kullanılmıştır. Bu yöntemde, ImageNet gibi büyük bir veri setinde önceden eğitilmiş bir model, daha küçük ve spesifik bir otopark veri seti üzerinde ince ayar yapılarak kullanılmıştır. Bu durum, eğitim süresini kısaltmış ve modelin doğruluğunu artırmıştır.

Yapılan testler ve deneyler sonucu sistemin, en fazla %5 oranında yanlış sınıflandırma yaptığı ve %95 oranında başarılı sınıflandırma yaptığı gözlemlenmiştir. Ayrıca, farklı ışık ve hava koşullarında yapılan testlerde, sistem yüksek bir başarı oranı göstermiştir.

2.2.10 Merkeziyetsiz park yeri doluluk tespiti için derin öğrenme

Bu çalışmada, merkeziyetsiz bir sistem tasarlanmıştır. CNN derin öğrenme algoritması kullanılarak, park yerlerinin doluluk durumunu doğrudan kamera seviyesinde tespit edilmesi önerilmiştir. Bu sayede, veri iletimi ve bant genişliği gereksinimleri azaltılmış, aynı zamanda gerçek zamanlı olarak yüksek performans sağlanmıştır [28].

Sistemde, otoparklara stratejik olarak yerleştirilen birden fazla kamera kullanılmıştır. Her kamera, kendi video akışını bağımsız olarak işler. Böylece, veriler merkezi bir işlem birimine gönderilmeden yerel olarak işlenir.

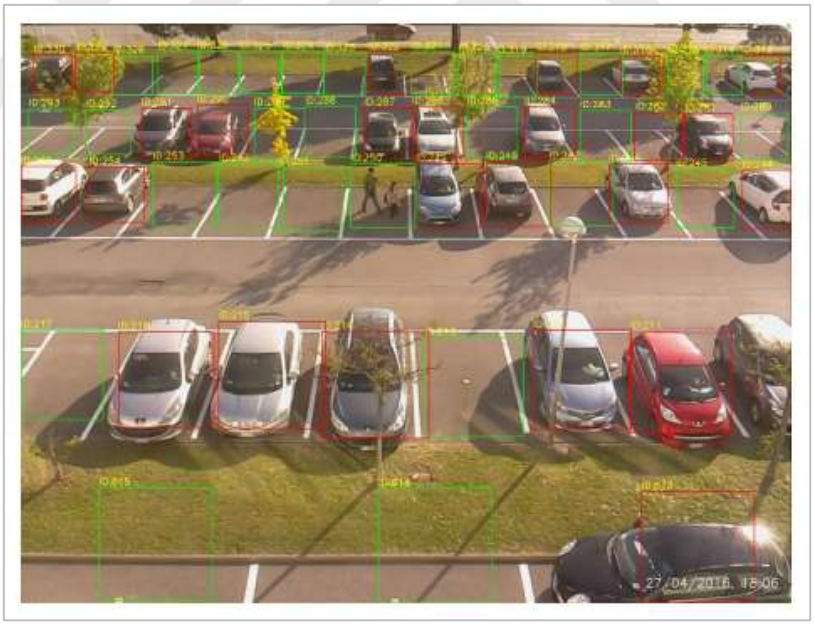
Her kamera, yerel bir işleme birimiyle donatılmıştır ve video akışını analiz ederek park yerlerinin doluluk durumunu gerçek zamanlı olarak sınıflandırır. Bu merkeziyetsiz mimari, veri iletimini önemli ölçüde azaltmıştır. Bu durum, veri aktarım gereksinimlerini ve ağ trafiğini minimuma indirmiştir. Ayrıca, büyük ölçekli otoparklarda sistem için ölçeklenebilirlik sağlamıştır. Çünkü kameralar bağımsız

olarak çalışır ve merkezi bir sunucuyu aşırı yükleme yapılmadan veriler işlenebilir hale gelmiştir.

Sistemde, görüntü sınıflandırma için etkili bir CNN derin öğrenme modeli kullanılmıştır. Çalışmada kullanılan CNN, özellikle görüntü işleme ve video akışlarını analiz etme görevlerinde oldukça başarılı olmuştur. Çünkü bu model, görüntülerden kenarlar, dokular ve şekiller gibi karmaşık özellikleri öğrenebilmiştir.

Model, park yerlerinin boş veya dolu olarak etiketlendiği veri setleriyle eğitilmiştir. CNN modeli, park yerlerinin durumunu belirleyecek şekilde, görsellerdeki örüntüleri tanır ve hangi yerlerin boş, hangi yerlerin dolu olduğunu öğrenir.

Her bir kamera, gerçek zamanlı şekilde video akışını işleyerek, her bir park yerinin doluluk durumunu sürekli günceller. Kamera, park yerlerinin boş veya dolu olduğunu anlık olarak belirler ve bu bilgiyi hızlıca sunucuya veya LED donanıma iletir.



Şekil 2.24: Merkeziyetsiz Park Yeri Doluluk Tespiti için Derin Öğrenme

Kaynak: Amato ve diğerleri. (2016)

Ön işleme adımları arasında, görüntülerin yeniden boyutlandırılması, piksel değerlerinin normalize edilmesi, döndürme, çevirme ve parlaklık artırma adımları yer almıştır. Bu işlemler, modelin çevresel değişkenliklere karşı daha dayanıklı olmasını sağlamıştır.

CNN modeli, bir dizi evrişim katmanı ve ardından tam bağlı katmanlar kullanarak görüntülerden yüksek seviyeli özellikler öğrenir. Ayrıca sistemde, transfer öğrenmesi yöntemi de kullanılmıştır. Bu aşamada, park yeri görüntüleri üzerinde ince ayar yapılmıştır. Böylece, modelin doğruluk oranı artırılmıştır.

Eğitim tamamlandıktan sonraki aşamada, her bir kamera kendi video akışını yerel olarak işler ve park yerlerinin doluluk durumunu gerçek zamanlı olarak belirler. Veriler yerel olarak işlendiği için, kameradan merkezi bir sunucuya veri iletilmesine gerek duyulmaz. Yalnızca her park yeri için boş veya dolu sınıflandırma sonucu merkezi sunucuya iletilir, bu da bant genişliği kullanımını minimuma indirmiş olur.

Dokuz adet kamera ile yapılan test ve deneyler sonucunda, sistemin ortalama başarılı tespit oranı %95,88 olarak gözlemlenmiştir.

3. MATERYAL VE YÖNTEM

Bu bölümde, çalışmanın amacına ulaşabilmesi için kullanılan materyaller ve uygulanan adımlar ayrıntılı bir şekilde açıklanmıştır. Çalışmanın güvenilirliğini ve geçerliliğini sağlamak amacıyla, veri toplama süreci, kullanılan araçlar, araştırma deseni, örneklem özellikleri ve analiz teknikleri sistematik bir biçimde sunulmuştur. Uygulanan yöntemlerin seçilme gerekçeleri belirtilmiş ve çalışma sürecinin her aşaması şeffaf biçimde aktarılmıştır. Böylece, çalışmanın bilimsel niteliğinin ve tekrarlanabilirliğinin temelleri ortaya konmuştur.

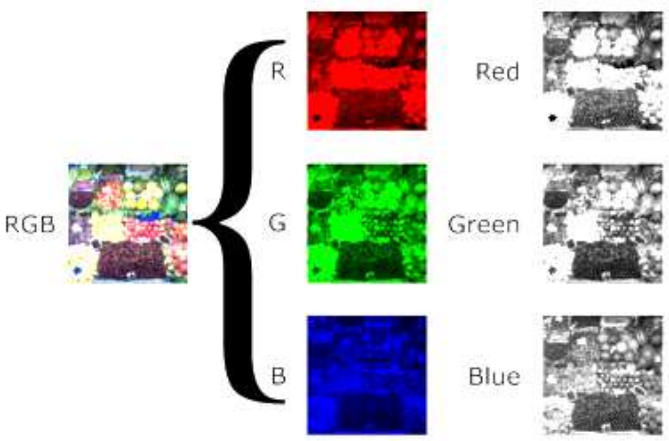
3.1 Park Yeri Durum Tespiti İçin Kullanılan Görüntü İşleme Teknikleri

Bu çalışmada geliştirilen gerçek zamanlı park yeri durum tespit sisteminde kullanılan görüntü işleme tekniklerinin açıklanması bu bölümde yapılmıştır. Bu teknikler, geliştirilen sistemde kullanınan algoritmaların temelini oluşturmaktadır. Ayrıca, daha ileri düzeydeki tekniklerin anlaşılmasına yardımcı olur.

3.1.1 Gri tonlamalı görüntü

Gri tonlamalı görüntü, her pikselin yalnızca bir gri tonuyla temsil edildiği bir görüntü türüdür. Normal renkli bir görüntüde, her piksel genellikle kırmızı, yeşil ve mavi renk bileşenlerinin bir kombinasyonu ile temsil edilir. Ancak gri tonlamalı bir görüntüde, her piksel, genellikle değeri 0 ile 255 arasında değişen sadece tek bir gri tonu ile temsil edilir. Bu durum, görüntünün renkli ayrıntılarının kaldırıldığı ve sadece parlaklık bilgisinin korunduğu anlamına gelir [16].

Gri tonlamalı görüntüler, belirli uygulamalarda kullanışlıdır. Örneğin, yüz tanıma, nesne tanıma, kenar tespiti ve benzeri görevlerde sıklıkla kullanılırlar. Çünkü renk bilgisine ihtiyaç duymadan ana görüntü yapısını korurlar. Ayrıca, daha düşük bellek ve işlemci gücü gerektirirler. Bu durum, onları bazı uygulamalarda tercih edilir hale getirir.



Şekil 3.1: Gri Tonlamalı Görüntü Örneği

Kaynak: Dilmen (2012)

3.1.2 Gauss bulanıklığı

Gauss bulanıklığı, görüntü işleme alanında sıkça kullanılan bir bulanıklık tekniğidir. Bu teknik, bir görüntüdeki detayları yumuşatmak ve gürültüyü azaltmak için kullanılır.

Gauss bulanıklığı, bir Gauss normal dağılım fonksiyonunu kullanarak pikselleri bulanıklaştırır. Her piksel, çevresindeki diğer piksellerin ağırlıklı ortalamasıyla değiştirilir [16].

Bu ağırlıklı ortalama, pikselin etrafındaki bir Gauss dağılımı tarafından belirlenir. Bu durum daha fazla ağırlığın merkezdeki piksellere ve daha az ağırlığın merkeze uzak piksellere verildiği anlamına gelir [16].

Gauss bulanıklığı, bir görüntüyü yumuşatmanın yanı sıra kenar tespiti ve diğer görüntü işleme tekniklerinde de kullanılır. Bu yöntem, görüntüdeki keskin geçişlerin yumuşatılmasına ve gürültünün azaltılmasına yardımcı olur. Böylece daha düzgün ve işlenebilir bir görüntü elde edilir.

Gauss bulanıklığı, her bir piksele uygulanacak dönüşümü hesaplamak için istatistikte normal dağılımını da ifade eden bir Gauss fonksiyonunu kullanan görüntü bulanıklama filtresidir. Bir boyutta Gauss fonksiyonunun formülü aşağıda gösterilmiştir [16].

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{3.1}$$

Formülde bulunan x değeri yatay eksenin orijinden uzaklığı, y değeri dikey eksenin orijinden uzaklığı ve σ değeri Gauss dağılımının standart sapmasıdır.

a) Orjinal Görüntü b) Uyarlanabilir Eşikleme

Şekil 3.2: Gauss Bulanıklığı Örneği

Kaynak: doxygen (2025)

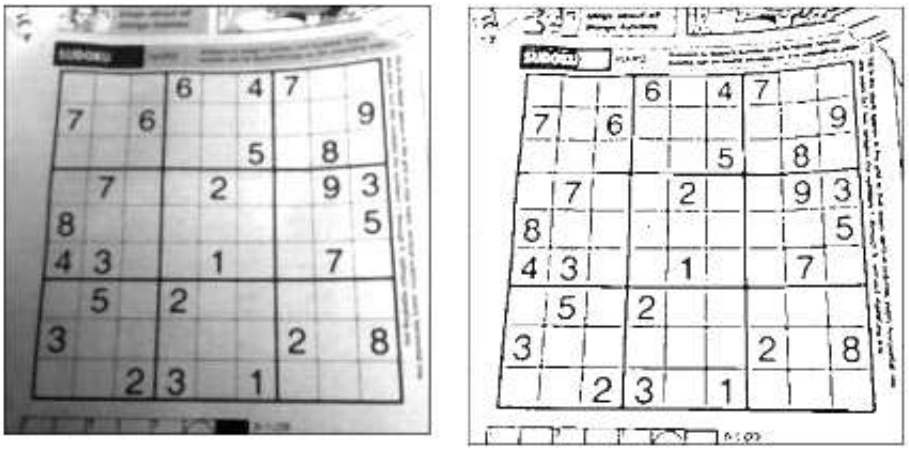
3.1.3 Uyarlanabilir eşikleme

Uyarlanabilir eşikleme, görüntü işlemede sıkça kullanılan bir tekniktir. Bu teknik, bir görüntüyü siyah beyaz formata dönüştürmek için kullanılır. Özellikle farklı aydınlatma koşullarında değişen görüntülerle çalışırken yararlıdır.

Geleneksel eşikleme tekniklerinde, bir eşik değeri seçilir ve bu eşik değerinin üzerindeki pikseller beyaz, altındakiler ise siyah olarak işaretlenir. Ancak, bu yöntem aydınlatma koşullarındaki değişikliklerden etkilenir ve sabit bir eşik değeri seçmek zor olabilir.

Uyarlanabilir eşikleme ise, görüntünün farklı bölgelerindeki aydınlatma değişimlerine uyum sağlayabilen dinamik bir eşik değeri belirler. Algoritma, görüntüyü küçük parçalara böler ve her bir parça için o parçanın kendi eşik değerini hesaplar. Bu hesaplama ise genellikle piksellerin ortalaması veya ağırlıklı ortalaması alınarak yapılır [16].

Sonuç olarak, uyarlanabilir eşikleme tekniği, görüntülerin çeşitli aydınlatma koşullarında bile etkili bir şekilde işlenmesine olanak tanır ve özellikle OCR, nesne tespiti ve kenar tespiti gibi uygulamalarda yaygın olarak kullanılır. Bu yöntem, daha doğru sonuçlar elde etmek için geleneksel eşikleme yöntemlerinden daha verimlidir.



a) Orjinal Görüntü b) Uyarlanabilir Eşikleme

Şekil 3.3: Uyarlanabilir Eşikleme Örneği

Kaynak: doxygen (2025)

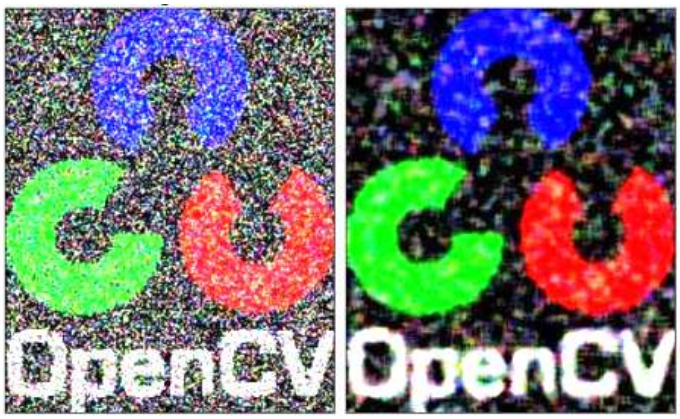
3.1.4 Medyan bulanıklığı

Medyan bulanıklığı, bir görüntüyü yumuşatmak ve gürültüyü azaltmak için kullanılan bir bulanıklık tekniğidir. Bu teknikte, her bir pikselin, kendisi ve çevresindeki piksellerin değerlerinin sıralandığı bir liste oluşturulur. Daha sonra listenin medyan değeri pikselin yeni değeri olarak seçilir [16].

Medyan bulanıklığı, özellikle tuz ve biber gürültüsü gibi gürültü türlerinde etkilidir. Bu tür gürültüler, rastgele piksellerin aşırı parlak veya aşırı karanlık olmasına neden olur ve bu rastgele pikseller görüntüde istenmeyen noktacıklar olarak belirir.

Medyan bulanıklığı bu tür gürültüyü azaltırken, görüntünün kenarlarını korumaya yardımcı olur. Çünkü bu teknikte, bir pikselin değeri, sadece kendisi ve çevresindeki piksellerin değerlerinin sıralandığı listedeki medyan değerdir [16].

Medyan bulanıklığı, görüntü işlemede önemli bir araçtır ve özellikle diğer bulanıklık teknikleri, kenar tespiti veya nesne tanıma gibi tekniklerle birlikte kullanıldığında oldukça yararlıdır.



a) Orjinal Görünütü b) Median Bulanıklığı

Şekil 3.4: Medyan Bulanıklığı Örneği

Kaynak: doxygen (2025)

3.1.5 Genişletme

Görüntü işlemede genişletme işlemi, morfolojik bir işlemdir. Bu işlem, bir görüntü üzerindeki beyaz bölgeleri genişletirken ve büyütürken, siyah bölgeleri ise daraltır ve küçültür.

Genişletme işlemi, genellikle nesneleri birleştirme, boşlukları kapatma, nesneleri büyütme ve kontur genişletme gibi amaçlarla kullanılır. Temelde, görüntü karesindeki her bir pikselin, bir bölgesindeki en yüksek değere sahip piksel değerinin karşıt bir piksel değeri ile değiştirildiği görüntü işleme tekniğidir [16].

Genişletme işlemi, yapı elemanı adı verilen bir çekirdek kullanılarak gerçekleştirilir. Bu yapı elemanı, genellikle kare, dikdörtgen veya daire şeklinde olabilir ve genişletme işleminin nasıl gerçekleşeceğini belirler. Yapı elemanı, her bir pikselin etrafındaki komşu pikselleri kontrol etmek için kullanılır. Genişletme işlemi, bu yapı elemanının piksel üzerindeki konumuna ve içeriğine bağlı olarak gerçekleştirilir [16].

Sonuç olarak, genişletme işlemi, nesnelerin boyutunu artırmak veya bağlantılarını güçlendirmek için kullanılırken, görüntü üzerindeki küçük delikleri veya boşlukları doldurmak için de kullanılabilir.

a) Orjinal Görünütü b) Genişletme

Şekil 3.5: Genişletme Örneği

Kaynak: doxygen (2025)

3.2 Park Yeri Durum Tespit Sistemin Geliştirilmesi

Python programlama dili, temiz ve okunabilir sözdizimine sahip yüksek seviyeli bir programlama dilidir. Bu nedenle, görüntü işleme tabanlı uygulamaları geliştirmek için tercih edilir. Ayrıca Python, kullanıcıya geniş bir kütüphane ekosistemi sunar, hızlı prototiplemeyi ve geliştirme süreçlerini destekler [17].

OpenCV (Open Source Computer Vision Library), görüntü işleme ve makine görüşü uygulamalarını geliştirmek için kullanılan açık kaynaklı bir kütüphanedir. OpenCV, birçok temel ve gelişmiş görüntü işleme fonksiyonu ve algoritması sunar.

Ek olarak OpenCV, kenar tespiti, nesne tanıma, yüz tanıma, stereo görüş, kamera kullanılarak konum belirleme gibi birçok alanda kullanılabilen geniş bir araç seti sunar. Bu nedenlerden dolayı bu çalışmada, Python programlama dili ve OpenCV kütüphanesi birlikte kullanılmıştır.

Çalışmada geliştirilen sistemde, ilk olarak OpenCV'nin "cv2.VideoCapture" ve "read" fonksiyonları kullanılarak video akışından görüntüler elde edilmiş ve ekranda gösterilmiştir. Daha sonrasında video akışı her bittiğinde programı tekrar başlatmamak için video akışı döngüye sokulmuştur.

Bu işlem, video akışında son görüntü karesine gelindiğinde mevcut görüntü karesinin ilk görüntü karesi olarak atanmasıyla yapılmıştır.

```
116    #Görüntü Okuma
117    vf=cv2.VideoCapture(dosyaadi)
118    while True:
119        #Video Tekrar
120        #Mevcut frame, Toplam frame sayisina eşit ise sıfırıncı frame e git
121        if vf.get(cv2.CAP_PROP_POS_FRAMES) == vf.get(cv2.CAP_PROP_FRAME_COUNT):
122            vf.set(cv2.CAP_PROP_POS_FRAMES,  value: 0)
123        success, frame = vf.read()
124        cv2.imshow( winname: "Goruntu",frame)
```

Şekil 3.6: Video'dan Görüntü Karesi Alma Kodu

Şekil 3.7: Orjinal Görüntü

Kaynak: jhorrocks (2022)

Görüntünün gri tonlamalı hale dönüştürülmesi, her bir pikselin renk bileşenlerinin birleştirilmesiyle yapılır. Normalde renkli bir görüntü, her piksel için kırmızı (R), yeşil (G) ve mavi (B) renk bileşenlerinden oluşur. Gri tonlamalı bir görüntü ise, her pikselin sadece tek bir gri tonuyla temsil edildiği bir görüntü formatıdır.

Gri tonlamalı bir görüntü oluşturmak için, görüntüdeki her pikselin RGB değerinin ortalaması alınır. Bu ortalamaya dayalı olarak, her pikselin yeni değeri tek bir gri tonunu temsil eder. Matematiksel olarak, bu işlem aşağıdaki şekilde ifade edilir:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad (2.2)$$

Burada, R, G ve B sırasıyla kırmızı, yeşil ve mavi bileşenlerinin değerini temsil eder. Katsayılar olan 0.299, 0.587 ve 0.114 değerleri ise insan gözünün farklı renklere olan duyarlılığını ifade eder.

Formül 2.2, renkli bir görüntüyü gri tonlamalı hale dönüştürmek için park yeri durum tespit sisteminde bulunan algoritma tarafından doğrudan kullanılabilir. Ancak, OpenCV, bu dönüşümü daha hızlı ve etkili bir şekilde gerçekleştirebilmek için optimize edilmiş birtakım fonksiyonlar bulundurur ve bunları kullanıcıya sunar. Bu fonksiyonlar, piksel dönüşümlerini vektörleştirerek ve daha optimize edilmiş algoritmalar kullanarak renkli görüntüyü gri tonlamalı görüntüye dönüştürme işlemini gerçekleştirir. Bu durum, görüntü işleme uygulamalarında performansı artırır ve daha hızlı işleme gücü sunar. Geliştirilen sistemde, OpenCV kütüphanesinde bulunan renk değiştirme fonksiyonu olan "cvtColor()" fonksiyonuna mevcut görüntü karesiyle birlikte "cv2.BGR2GRAY" parametresi verilerek renkli görüntüyü gri tonlamalı görüntüye dönüştürme işlemi yapılmıştır.

```
129     #Gri Tonlamali Goruntuye (Grayscale) Donusturme
130     frameGray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
131     cv2.imshow( winname: 'Gri Tonlamali Goruntu', frameGray)
```

Şekil 3.8: Gri Tonlamalı Görüntüye Dönüştürme Kodu

Şekil 3.9: Gri Tonlamalı Görüntüye Dönüşmüş Görüntü

Görüntüye Gauss bulanıklığı uygulama işlemi, her bir pikselin çevresindeki piksellerin ağırlıklı ortalamasının hesaplanmasıyla ve bu ağırlıklı ortalamanın pikselin yeni değeri olarak kullanılması ile yapılır.

Gauss bulanıklığında, görüntüdeki her pikselin etrafındaki piksellerin değerlerine dayalı olarak bir Gauss normal dağılımı kullanıp bulanıklık işlemi uygulanır.

Gauss bulanıklığı uygulanırken, bir filtre çekirdeği veya maske kullanılır. Bu maske, Gauss fonksiyonunun bir yaklaşımını temsil eder ve görüntüdeki her bir pikselin etrafındaki piksellerin ağırlıklı ortalamasını hesaplamak için kullanılır.

Matematiksel olarak, bir Gauss bulanıklığı filtresi, genellikle Gauss fonksiyonunun bir yaklaşımı olan bir çekirdek kullanılarak uygulanır. Bu çekirdek, Gauss normal dağılımını temsil eden bir matris olarak düşünülebilir. Görüntüdeki her bir pikselin yeni değeri, çekirdek matrisi içindeki komşu piksellerin değerleri ile ağırlıklı bir ortalama alınarak hesaplanır.

Gauss bulanıklığı işlemi, görüntünün pürüzsüzleştirilmesine, görüntüdeki gürültünün azaltılmasına ve kenarların yumuşatılmasına yardımcı olur.

Geliştirilen sistemde, Gauss bulanıklığı işlemi uygulamak için OpenCV kütüphanesinde bulunan “cv2.GaussianBlur()” fonksiyonu kullanılmıştır.

Bu fonksiyonda, kutu filtresi yerine bir Gauss çekirdeği kullanılır. Görüntü karesini, çekirdek boyutunu ve Gauss fonksiyonunun x ve y yönlerindeki standart sapmalarını parametre olarak alır. Bu parametreler, Gauss bulanıklığı filtresinin boyutunu ve etkisini belirler.

Çekirdeğin genişliği ve yüksekliği, pozitif ve tek sayı olmalıdır. Ayrıca, X ve Y yönlerindeki standart sapmanın, fonksiyonu çağırırken parametrede sigmaX ve sigmaY olarak verilmesi gerekir. Eğer sadece sigmaX belirtilirse, sigmaY ve sigmaX eşit kabul edilir. Gauss bulanıklığı, bir görüntüden Gauss gürültüsünü çok etkili bir şekilde kaldırmak için kullanılır.

```
135     #Gaussian Blur Ekleme kernel boyutu=3,3 sigmax,sigmay=1
136     frameBlur = cv2.GaussianBlur(frameGray, ksize: (3,3), sigmaX: 1)
137     cv2.imshow( winname: 'Gauss Bulanikligi', frameBlur)
```

Şekil 3.10: Gauss Bulanıklığı Uygulama Kodu

Şekil 3.11: Gauss Bulanıklığı Uygulanmış Görüntü

Uyarlanabilir eşikleme, görüntünün farklı bölgelerindeki aydınlatma değişikliklerine uyum sağlayabilen bir eşikleme tekniğidir. Bu teknikte, görüntüdeki her pikselin etrafında bir eşik değeri belirlenir ve bu değere göre piksel siyah veya beyaz olarak işaretlenir.

Teorik olarak, uyarlanabilir eşikleme işleminde aşağıdaki adımlar izlenir:

İlk olarak görüntü, gri tonlamalı görüntüye dönüştürülür.

İkinci olarak, görüntü, örneğin 3x3 veya 5x5 gibi belirli bir boyut içindeki her piksel için ayrı bir eşik değeri hesaplamak için kullanılır.

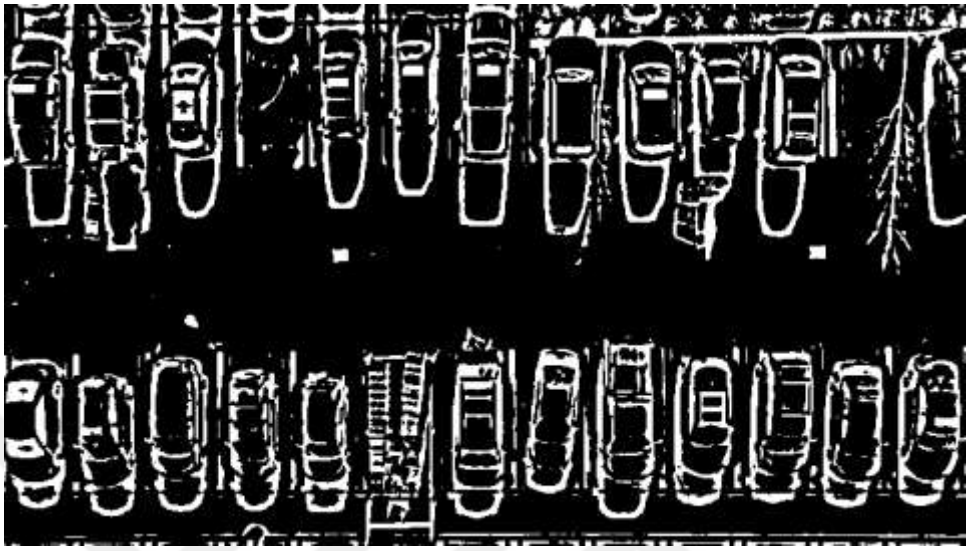
Üçüncü olarak, görüntüdeki her piksel için lokal eşik değeri, pikselin etrafındaki bir bölgeye dayalı olarak hesaplanır. Bu hesaplama, genellikle o pikselin etrafındaki bir bölgenin ortalama veya medyan değerine dayanır.

Son olarak, her piksel, kendi lokal eşik değeriyle karşılaştırılır. Eğer piksel değeri eşik değerinden büyükse, piksel beyaz olarak işaretlenir; aksi halde siyah olarak işaretlenir.

Geliştirilen sistemde, uyarlanabilir eşikleme işlemi uygulamak için OpenCV kütüphanesinde bulunan “cv2.adaptiveThreshold()” fonksiyonu kullanılmıştır. Bu fonksiyon, görüntü karesi, maksimum piksel değeri, eşikleme türü, eşikleme yöntemi, blok boyutu ve eşikleme için kullanılan C sabiti parametrelerini alır. Bu parametreler, uyarlanabilir eşikleme işleminin nasıl gerçekleştirileceğini belirler.

```
141     #Adaptive Threshold blockSize:25 C:16
142     frameThresh = cv2.adaptiveThreshold(frameBlur, maxValue: 255,
143     cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV, blockSize: 25, C: 16)
144     cv2.imshow( winname: 'Uyarlanabilir Esikleme', frameThresh)
```

Şekil 3.12: Uyarlanabilir Eşikleme Uygulama Kodu



Şekil 3.13: Uyarlanabilir Eşikleme Uygulanmış Görüntü

Görüntüye medyan bulanıklığı uygulamak için, görüntüdeki her bir pikselin etrafındaki bir bölgede bulunan piksellerin değerleri sıralanır. Sıralanan bu değerlerin medyan değeri pikselin yeni değeri olarak seçilir.

Medyan bulanıklığı işlemi, görüntüdeki her pikselin değerini etkileyen bir filtre çekirdeği veya maske kullanılarak gerçekleştirilir.

Medyan bulanıklığı işleminde aşağıdaki adımlar izlenir:

İlk olarak görüntü, gri tonlamalı görüntüye dönüştürülür.

İkinci olarak, her bir pikselin etrafında bir bölge belirlenir. Bu bölge genellikle bir kare veya dikdörtgen şeklinde olabilir ve pikselin çevresindeki komşu pikselleri içerir.

Üçüncü olarak, her pikselin etrafındaki piksellerin değerleri sıralanır.

Son olarak, sıralanmış piksel değerleri arasındaki medyan değer, pikselin yeni değeri olarak seçilir. Bu işlem, görüntünün her pikseli için tekrarlanır ve medyan bulanıklığı uygulanmış yeni bir görüntü elde edilir.

Geliştirilen sistemde, medyan bulanıklığı işlemi uygulamak için OpenCV kütüphanesinde bulunan “cv2.medianBlur()” fonksiyonu kullanılmıştır.

Bu fonksiyon, görüntü karesini ve medyan bulanıklığı işlemi için kullanılacak filtre çekirdeğinin boyutunu parametre olarak alır. Bu parametreler, medyan bulanıklığı filtresinin boyutunu ve etkisini belirler. Çekirdek boyutu pozitif ve tek tam sayı olmalıdır.

```
148     #Median Blur kernel boyutu=5
149     frameMBlur = cv2.medianBlur(frameThresh, ksize: 5)
150     cv2.imshow( winname: 'Medyan Bulanikligi', frameMBlur)
```

Şekil 3.14: Medyan Bulanıklığı Uygulama Kodu

Şekil 3.15: Medyan Bulanıklığı Uygulanmış Görüntü

Görüntüde genişletme işlemi, bir yapı elemanı kullanılarak gerçekleştirilir. Bu işlem, beyaz piksellerin yani nesnelerin alanlarını genişletirken, siyah piksellerin yani arka planın alanını küçültür.

Genişletme işleminde, ilk olarak, görüntü üzerindeki her bir piksel, yapı elemanının merkeziyle hizalanır. Sonrasında, yapı elemanı, pikselin etrafındaki komşu pikselleri kontrol etmek için kullanılır. Eğer yapı elemanı ve pikselin çakıştığı herhangi bir piksel beyaz ise, komşu piksel de beyaz yapılır. Bu işlem, görüntünün her pikseli için tekrarlanır ve genişletilmiş yeni bir görüntü elde edilir.

Teorik olarak, bir genişletme işlemi, her pikselin etrafındaki bir bölgeye dayalı olarak gerçekleştirilir. Bu bölge genellikle bir kare veya dikdörtgen şeklinde olabilir. Piksel ve bu bölge çakıştığında, pikselin değeri beyaz yapılır. Genişletme işleminin teorik formülü aşağıdaki gibidir:

$$dst(x,y) = \max_{(x',y'):element(x',y')\neq 0} src\,(x+x',y+y') \tag{2.3}$$

Geliştirilen sistemde, genişletme işlemi uygulamak için OpenCV kütüphanesinde bulunan "cv2.dilate()" fonksiyonu kullanılmıştır. Bu fonksiyon, görüntü karesini, bir yapı elemanını ve genişletme işlemi için kullanılan iterasyon sayısını parametre olarak alır. Bu parametreler, genişletme işleminin boyutunu ve etkisini belirler. Çalışmada, yapı elemanı, kare olarak kullanılmıştır.

```
154     #Dilation
155     frameDil = cv2.dilate(frameMBlur,np.ones( shape: (3,3),np.uint8),iterations=1)
156     cv2.imshow( winname: 'Genisletme', frameDil)
```

Şekil 3.16: Genişletme Uygulama Kodu

Şekil 3.17: Genişletme Uygulanmış Görüntü

Bölgesel işleme algoritmaları, bir görüntü karesinde belirli bölgelerin veya alanların işlenmesini sağlarlar. Park yeri tespiti uygulamalarında, ilk olarak otoparktan alınmış bir görüntüde, kullanıcı tarafından boş veya dolu olarak sınıflandırılması istenen park yerleri belirlenir ve ardından bu bölgelerde daha ayrıntılı analizler yapılır.

Birbirinden farklı otoparklardaki park yerleri, farklı şekillerde ve yapılarda olabileceğinden dolayı, geliştirilen sistemde kullanıcının veya sistem yöneticisinin öncesinde durum tespiti yapılması istenen park yerlerini seçerek sistemi kalibre etmesi gerekmektedir. Bunun için geliştirilen sistemde bir arayüz oluşturulmuştur.

Bu arayüz içerisinde video akışının ilk görüntü karesi veya otoparktan alınmış kuş bakışı bir görüntü ekrana getirilmektedir. Kullanıcı farenin sol tıkını kullanarak park yerlerini işaretleyebilir veya farenin sağ tıkını işaretlediği alanlar içerisinde kullanarak, öncesinde yapmış olduğu işaretlemeleri kaldırabilir.

Şekil 3.18: Kullanıcı Tarafından Seçilmiş Park Yerleri

Kullanıcının, Şekil 3.18'deki gibi arayüzde işaretlediği park yerlerinin konumları "pickle" kütüphanesi kullanılarak bir pickle dosyası içerisinde saklanır. Sonrasında, gerçek zamanlı olarak park yerlerinin dolu veya boş olduğunun tespitini yapan algoritma tarafından, bu park yerlerinin konumu kullanılır. Geliştirilen algoritma, her bir park yerinin işaretli alanı içerisinde kalan görüntüde bölgesel işleme yapar.

```
34    try:
35        #dosyadaki konumları konum listesine at
36        with open(yerlerfile, 'rb') as dosya:
37            konumListesi = pickle.load(dosya)
38    except:
39        konumListesi = []
```

Şekil 3.19: Pickle Dosyası Yüklenme Adımı

Şekil 3.20: Bölgesel İşlemeden Sonra İşaretlenmiş Park Yerlerinin Görüntüsü

OpenCV'de “countNonZero()” fonksiyonu, bir görüntü karesindeki piksel değeri 0 olmayan piksellerin sayısını, yani beyaz piksellerin sayısını hesaplamak için kullanılır.

İlk olarak, video akışından alınan görüntü karesi, gri tonlamalı görüntüye dönüştürülür. Bu adım, beyaz piksellerin değerinin 0'dan farklı olduğu bir görüntü formatı elde etmek için gereklidir.

Sonraki aşamada, başlangıç değeri 0 olan bir sayaç tanımlanır. Görüntü karesinde bulunan ve kullanıcı tarafından işaretlenmiş her park yerinin işaretli alanı içerisinde bulunan piksellerin, sırasıyla değerlerine bakılır. Pikselin değeri 0'dan farklı ise, yani piksel beyaz piksel ise, sayacın değeri bir artırılır. İşaretli alan içerisindeki her piksel için bu işlem tekrarlanır ve üzerinde çalışılan park yeri için beyaz piksellerin sayısı hesaplanmış olur.

Bu işlem, görüntü karesindeki park yerleri bazında bir döngü kurularak gerçekleştirilir ve mevcut görüntü karesindeki işaretlenmiş her park yerinin içerisinde bulunan beyaz piksel sayısı hesaplanmış olur.

OpenCV’de bu işlem, “countNonZero()” fonksiyonu ile daha verimli bir şekilde yapılır. Fonksiyon, parametre olarak görüntü karesini alır ve içerisinde bulunan beyaz piksellerin sayısını daha az kaynak kullanarak, performanslı ve hızlı bir şekilde hesaplar.

```
53          if counter<=esik:
54              renk=(0,255,0)
55              kalinlik=5
56              bosYerCounter+=1
57          else:
58              renk=(0,0,255)
59              kalinlik = 2
```

Şekil 3.21: countNonZero() Fonksiyonu

İşaretlenmiş park yeri görüntüsündeki beyaz piksel sayısını tutan sayacın değeri, program çalışma süresi boyunca her görüntü karesi için adaptif olarak belirlenen bir X eşik değerine eşit veya daha küçükse, park yeri boş olarak sınıflandırılır. Park yeri sınırları, yeşil renkte ve daha kalın bir şekilde işaretlenir. Ayrıca, toplam boş park yeri sayısı bir artırılır.

Eğer sayacın değeri X eşik değerinden büyükse, park yeri dolu olarak sınıflandırılır. Park yeri sınırları, kırmızı renkte ve daha ince bir şekilde işaretlenir. Ayrıca, toplam boş park yeri sayısı bir azaltılır.

İşaretlenmiş her bir park yerinin sol alt köşesine, işaretli alan içinde kalan görüntünün beyaz piksel sayısı yazdırılır. Boş ve toplam park yeri sayısı mevcut görüntü karesinin sol üst köşesine yazdırılır.

Video akışından farklı zamanlarda elde edilen görüntülerde ışık koşullarının ve çevresel etkilerin değişkenlik göstermesi nedeniyle, sabit eşik kullanımı yerine her kareye özgü dinamik bir eşik değeri kullanılması tercih edilmiştir. Bu kapsamda, mevcut görüntü karesinde tüm park yerlerine ait beyaz piksel sayılarının aritmetik ortalaması alınarak mevcut karedeki tüm park yerleri için ortak olan X eşik değeri hesaplanmıştır:

$$X(t) = \frac{1}{N} \sum_{i=1}^{N} B_i(t) \tag{3.2}$$

Formül 3.2'deki değişkenler aşağıda açıklanmıştır:

X(t): t anındaki görüntü için dinamik olarak hesaplanan ortak eşik değeri.

Bi(t): i park yerindeki beyaz piksel sayısı

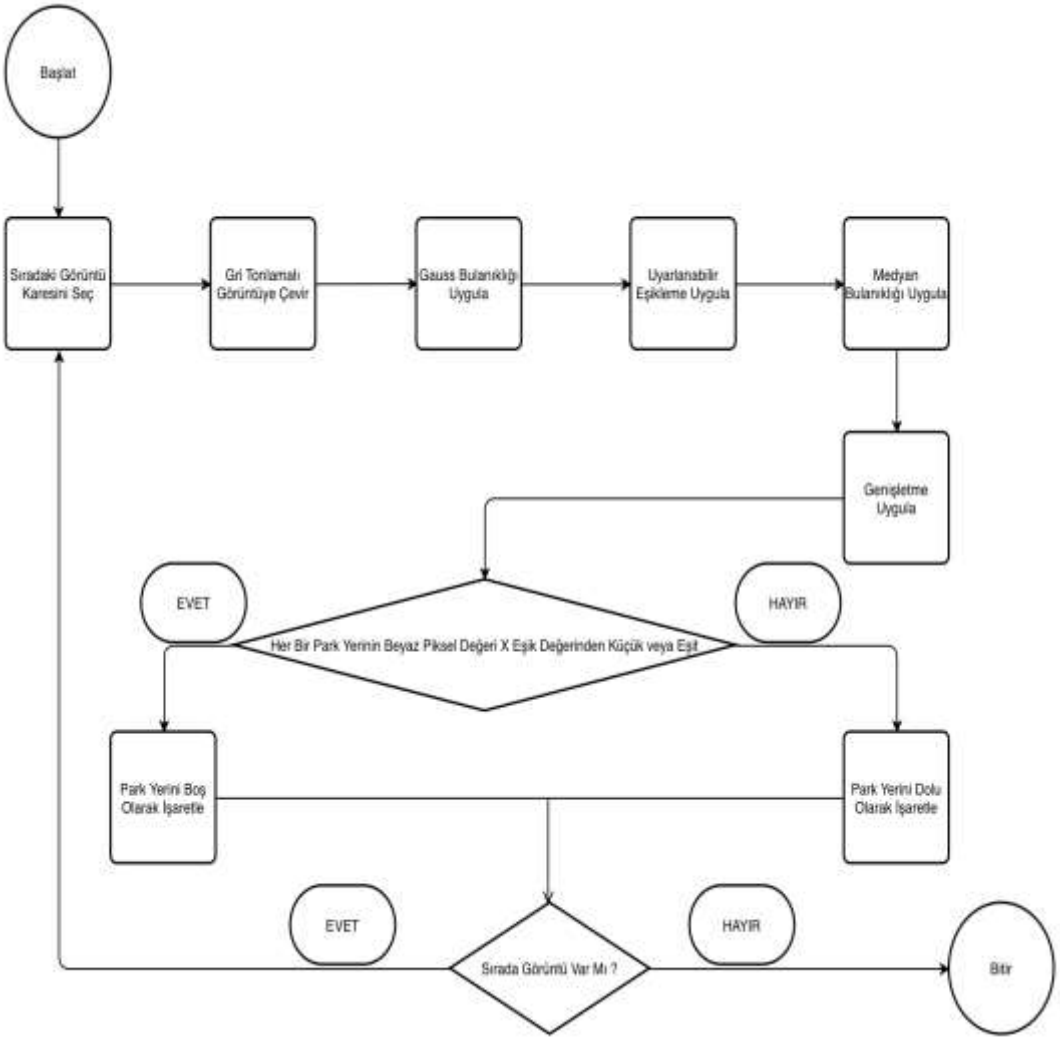
N: Otoparktaki toplam park yeri sayısı

Her park yerinin durumu, o park yerine ait beyaz piksel sayısının X(t) ile karşılaştırılması yoluyla belirlenmiştir. Bu çalışmada yapılan eşikleme işlemine bağlı olarak, park yerinde bir araç bulunuyorsa o park yerinde beyaz piksel sayısı artmakta; boş park yerlerinde ise beyazlık piksel sayısı görece daha düşük kalmaktadır. Bu nedenle park yerinin sınıflandırması şu şekilde yapılır:

$$\begin{aligned} &\text{Eğer } B_i(t) \leq X(t) \Rightarrow \text{Park yeri } i \text{ boştur} \\ &\text{Aksi takdirde } \Rightarrow \text{Park yeri } i \text{ doludur} \end{aligned} \tag{3.3}$$

Bu yöntem sayesinde geliştirilen sistem, ortam ışığındaki değişikliklere karşı daha tolare hale getirilmiş ve farklı görüntü koşullarında tutarlı sonuçlar verebilecek şekilde yapılandırılmıştır.

Geliştirilen sistemin, program çalışma süresi boyuncaki akış diyagramı, Şekil 3.22'de açıklanmıştır.



Şekil 3.22: Geliştirilen Sistemin Akış Diyagramı

4. TARTIŞMA VE BULGULAR

Bu bölümde, çalışmanın sonucunda elde edilen veriler analiz edilerek bulgular ortaya konmuş ve ilgili literatür ışığında tartışılmıştır. Elde edilen sonuçlar, çalışmanın amacı doğrultusunda değerlendirilmiş, benzer ve farklı çalışmalarla karşılaştırılarak yorumlanmıştır.

Bulguların hangi ölçüde mevcut bilgi birikimiyle örtüştüğü veya ondan ayrıldığı tartışılmış, bunun olası nedenleri açıklanmaya çalışılmıştır. Böylece, bu çalışmanın literatüre katkısı ve uygulamadaki yansımaları daha net bir şekilde ortaya konmuştur.

4.1 Geliştirilen Sistem ile Sınıflandırma

Geliştirilen sistem, gerçek dünyadan alınmış ve ücretsiz kaynak olan bir otopark video akışı üzerinde test edilmiş ve değerlendirilmiştir. Video akışı, 400 kare görüntü içermektedir ve otoparkta 25 adet park yeri bulunmaktadır. Başlangıçta, otoparkta 3 boş park yeri vardır. Otoparktan çıkış yapan araç sayısı 9 ve otoparka giriş yapan araç sayısı ise 8'dir.

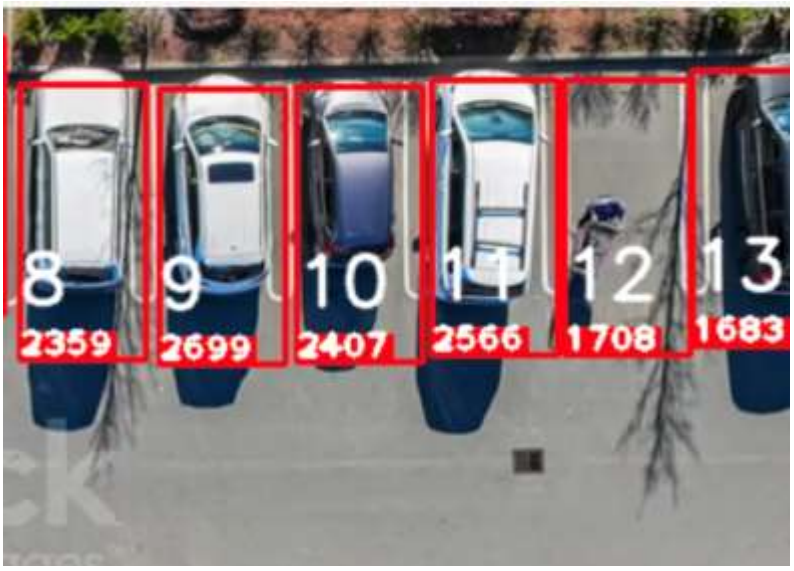
Geliştirilen sistemde harici bir özellik olarak, hareket eden araçlar mor renkli bir çerçeve ile işaretlenir. Bu işlem sadece bir önceki görüntü karesine kıyasla mevcut görüntü karesinde yeri değişmiş olan araçlar için yapılır. Böylece, iki görüntü karesi boyunca konumu değişmemiş, kısaca sabit duran araçlar işaretlenmez.

Şekil 4.1: Araç Park Yerine Giriş Yapmadan Önce Sınıflandırma

Şekil 4.2 Araç Park Yerine Giriş Yaptıktan Sonra Sınıflandırma

Geliştirilen sistem, park yerlerinin durumunu belirlemek için piksel tabanlı bir görüntü işleme tekniği kullandığından, ışık ve gölge değişiklikleri, sınıflandırma başarısını etkileyebilir. Bunun yanı sıra, işaretlenen park yerleri içerisinnde bulunan ve araç olmayan bazı nesneler araç olarak sınıflandırılabilir.

Video akışında, ilk olarak, 12 numaralı boş park yerinden geçen bir yaya, 5 kare görüntü boyunca, işaretli park yerine ait beyaz piksel sayısını artırmış ve bu sayıyı X eşik değerinin üzerine çıkarmıştır. Bu nedenle, 5 görüntü karesi boyunca 12 numaralı park yeri dolu olarak yanlış sınıflandırılmıştır.



Şekil 4.3: Geliştirilen Sistem Tarafından Hatalı Sınıflandırma

İkinci olarak, 21 numaralı park yerinden çıkış yapan aracın yarısından fazlası, park yerinden çıktığı için park yerine ait beyaz piksel sayısı, X eşik değerinin altına düşmüştür. Bu nedenle, 4 görüntü karesi boyunca, araç daha tam olarak park yerinden çıkmamasına rağmen 21 numaralı park yeri boş olarak yanlış sınıflandırılmıştır.

Şekil 4.4: Geliştirilen Sistem Tarafından Hatalı Sınıflandırma

Sonuç olarak, video akışında bulunan 400 kare görüntü boyunca, toplamda 9 kare görüntüde yanlış sınıflandırma yapıldığı ve otoparktaki toplam 25 park yerinden 23'ünün başarılı bir şekilde sınıflandırıldığı görülmüştür.

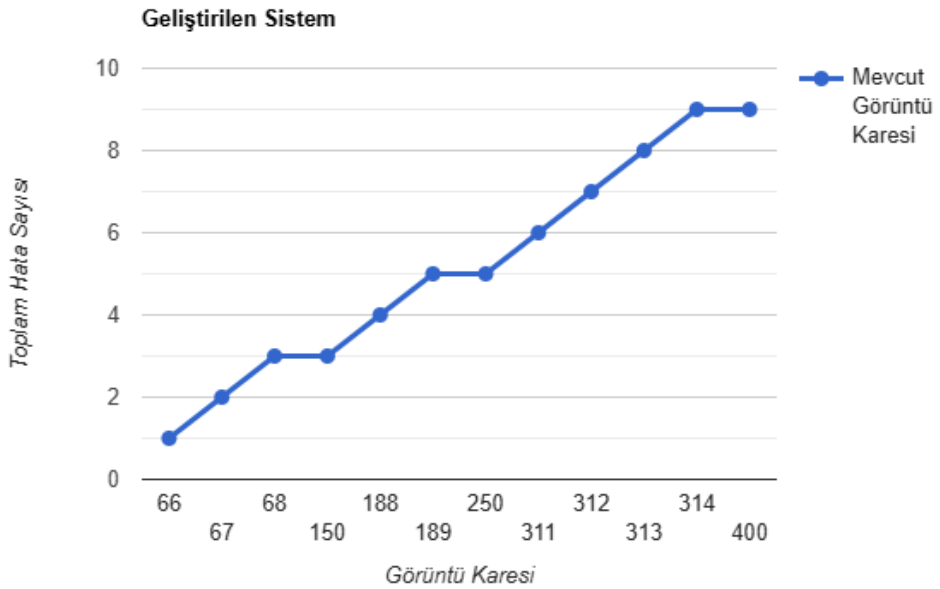
Geliştirilen sistemin, bu video akışı için hatalı sınıflandırma yüzdesi 9/400 kare görüntü üzerinden %2,25 olarak hesaplanmıştır.

Çizelge 4.1: Geliştirilen Sistemin Hatalı Sınıflandırmaları

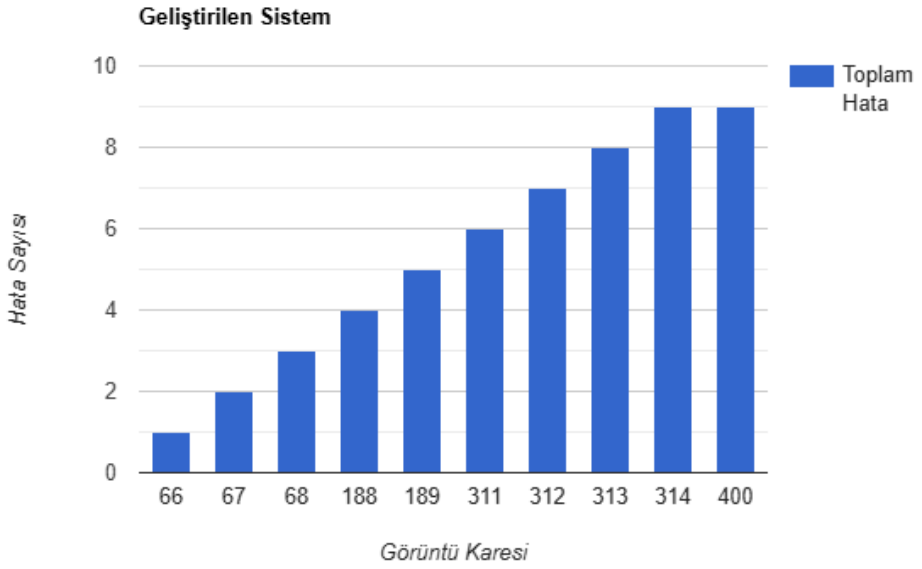
Görüntü Karesi	Park Yeri	Sınıflandırma	Gerçek Durum
66	12 Numara	Dolu	Yaya Var
67	12 Numara	Dolu	Yaya Var
68	12 Numara	Dolu	Yaya Var
188	12 Numara	Dolu	Yaya Var
189	12 Numara	Dolu	Yaya Var
311	21 Numara	Boş	Dolu
312	21 Numara	Boş	Dolu
313	21 Numara	Boş	Dolu
314	21 Numara	Boş	Dolu

Çizelge 4.2: Geliştirilen Sistemin Başarısı

Toplam Hata	9 Kare Görüntü
Başarı Yüzdesi	391/400 = %97,75
Hata Yüzdesi	9/400 = %2,25



Şekil 4.5: Geliştirilen Sistemin Hata Çizgi Grafiği



Şekil 4.6: Geliştirilen Sistemin Hata Bar Grafiği

İkinci bir deney olarak, geliştirilen sistem, başka bir gerçek dünyadan alınmış ve ücretsiz kaynak olan otopark video akışı üzerinde test edilmiş ve değerlendirilmiştir. Video akışı, 679 kare görüntü içermektedir ve otoparkta 69 adet park yeri bulunmaktadır. Başlangıçta, otoparkta 12 adet boş park yeri vardır. Otoparktan çıkış yapan araç sayısı 3'tür.

a) Araçların Çıkışı Öncesi b) Araçların Çıkışı Sonrası

Şekil 4.7: İki Araba Çıkış Yaptıktan Sonra Park Yerlerinin Boş Sınıflandırılması

Sonuç olarak, geliştirilen sistemin video akışında bulunan otoparktaki toplam 69 park yerinden 67'sini başarılı bir şekilde sınıflandırıldığı görülmüştür. Video akışı boyunca, yalnızca 13 ve 52 numaralı park yerlerinde yanlış sınıflandırma yapılmıştır.

Ek olarak, 13 ve 52 numaralı park yerlerinin, video akışında bulunan 679 kare görüntü içerisinde toplam 6 kare görüntüde yanlış şekilde sınıflandırdığı

görülmüştür. 13 numaralı park yeri için 3 kare görüntüde, 52 numaralı park yeri için de 3 kare görüntüde yanlış sınıflandırma gözlemlenmiştir.

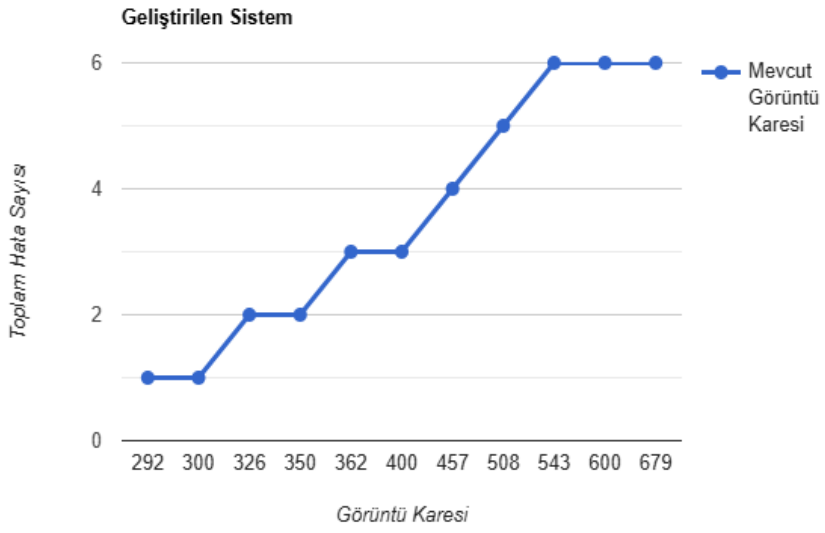
Böylece, geliştirilen sistemin, bu yeni video akışı özelinde hatalı sınıflandırma yüzdesi 6/679 görüntü kare üzerinden %0,88 olarak hesaplanmıştır.

Çizelge 4.3: Geliştirilen Sistemin Hatalı Sınıflandırmaları (2. Video)

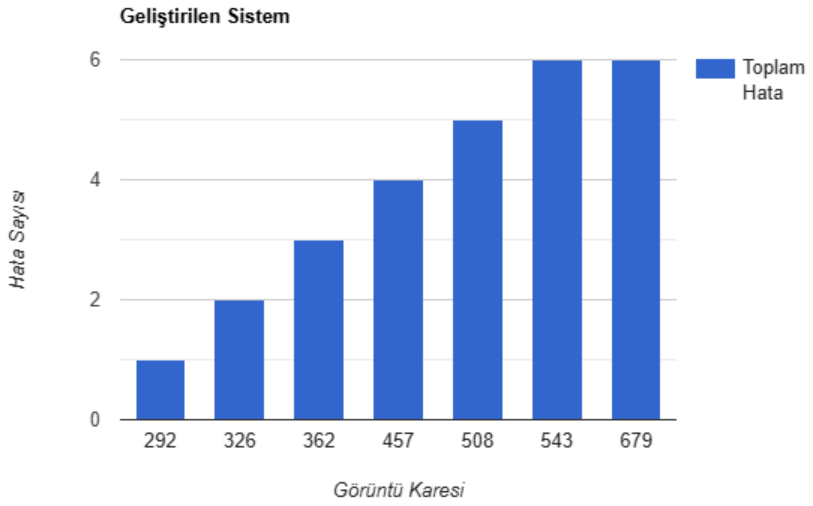
Görüntü Karesi	Park Yeri	Sınıflandırma	Gerçek Durum
292	13 Numara	Boş	Dolu
326	13 Numara	Boş	Dolu
362	13 Numara	Boş	Dolu
457	52 Numara	Dolu	Boş
508	52 Numara	Dolu	Boş
543	52 Numara	Dolu	Boş

Çizelge 4.4: Geliştirilen Sistemin Başarısı (2. Video)

Toplam Hata	6 Kare Görüntü
Başarı Yüzdesi	673/679 = %99,12
Hata Yüzdesi	6/679 = %0,88



Şekil 4.8: Geliştirilen Sistemin Hata Çizgi Grafiği (2. Video)



Şekil 4.9: Geliştirilen Sistemin Hata Bar Grafiği (2. Video)

4.2 Google TensorFlow ile Sınıflandırma

TensorFlow, Google tarafından geliştirilen açık kaynaklı bir makine öğrenimi yazılım sistemidir. Derin öğrenme modelleri başta olmak üzere, makine öğrenimi modelleri oluşturmak, eğitmek ve dağıtmak için kapsamlı bir ekosistem sunar [19].

TensorFlow, doğal dil işleme ve görüntüden nesne tanıma gibi çeşitli alanlardaki uygulamalarda, öğrenmeyi ve model üretmeyi desteklemek için kullanılır. Hem araştırma hem de endüstriyel alanda, akıllı sistemler oluşturmak için yaygın olarak tercih edilmektedir [19].

Bir modelin eğitilmesi için, sınırlayıcı kutulara sahip ve sınıf etiketleri ile etiketlenmiş görüntüler toplanmalıdır. Ardından, önceden eğitilmiş bir model, sisteme yüklenip, oluşturulan veri seti üzerinde ince ayarlar yapılmalıdır. Yapılan ince ayarlamalar, genellikle modelin son katmanlarının, oluşturulan veri setindeki özel sınıflara uyacak şekilde ayarlanması işlemidir.

TensorFlow, sınıflandırma için çapraz entropi kaybı ve sınırlayıcı kutu regresyonu için MAE kaybı gibi çeşitli kayıp fonksiyonları kullanır. Eğitim tamamlandıktan sonra, model, veri setindeki performansını değerlendirmek için ortalama doğruluk gibi metriklerle değerlendirilir.

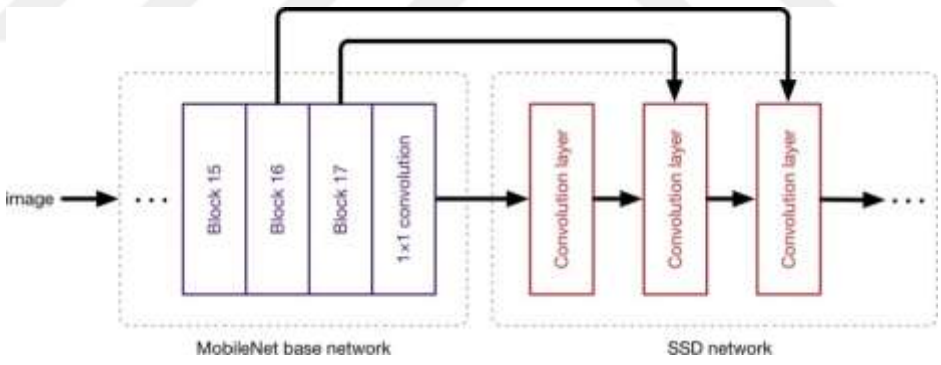
Eğitilen model, uygulamalarda gerçek zamanlı nesne tespiti için kullanılabilir. Resimleri veya video akışı içerisindeki görüntü karelerini işleyerek, nesneleri sınırlayıcı kutular ve sınıf etiketleri ile tanımlayabilir, ve mevcut görüntü karesi üzerinde gösterebilir.

Özet olarak, TensorFlow'un güçlü nesne tespit teknolojisi ve eğitilmiş modelleri, otopark izleme sistemlerinden, otonom araçlara kadar birçok uygulama için doğruluk oranı yüksek olan nesne tespiti sistemlerinin geliştirilmesini ve kullanılmasını sağlar.

Bu çalışmada, TensorFlow ile birlikte SSD MobileNet teknolojisi kullanılmıştır. SSD kullanarak bir model eğitmek, doğruluk ve hız arasında çok iyi bir denge sağlar. SSD, verilen giriş görüntüsü üzerinde yalnızca bir kez evrişimsel sinir ağı çalıştırır ve bir özellik haritası hesaplar.

Daha sonra, SSD bu özellik haritası üzerinde küçük 3x3 boyutlarında bir evrişimsel çekirdek çalıştırarak sınırlayıcı kutuları ve sınıflandırma olasılığını tahmin eder. Ayrıca, çeşitli en boy oranlarında çapa kutuları kullanır ve kutuyu öğrenmek yerine, ofseti öğrenir.

SSD, birden fazla evrişimsel katmandan sonra sınırlayıcı kutuların içerisindeki nesneleri tahmin eder. Her evrişimsel katman farklı bir ölçekte çalıştığından, farklı boyutlardaki nesneleri tespit edebilir. SSD MobileNet V2 modelinin blok diyagramı, Şekil 4.10'da gösterilmiştir.



Şekil 4.10: SSD MobilNet Akış Diyagramı

Kaynak: Pokhrel (2020)

Bu çalışmada TensorFlow ve SSD MobileNet'in birlikte kullanılması aşağıdaki nedenlere dayanmaktadır:

Hızlı Çalışma

SSD mimarisi, nesneleri tek bir işlemde tanıyıp sınıflandırabilmesi sayesinde hızlıdır. Ek olarak, MobileNet hafif bir modeldir, bu da mobil ve gömülü cihazlarda bile hızlı çalışmasını sağlar.

Verimli Kaynak Kullanımı

MobileNet, düşük bellek ve işlem gücü gerektirdiğinden, kaynakları sınırlı olan cihazlarda iyi bir performans sunar. Bu nedenle, gerçek zamanlı çalışan uygulamalar için idealdir.

Yüksek Doğruluk

SSD MobileNet, yüksek doğruluk oranlarında nesne tanıma sonuçları sağlayan etkili bir mimari sunar. Özellikle, hafif modeller arasında iyi bir doğruluk oranı yakalamak mümkündür.

Esneklik ve Uyumluluk

TensorFlow, modelin kolayca eğitilmesi, ince ayar yapılması ve farklı veri setlerine uygulanması için güçlü araçlar sunar. Ek olarak, TensorFlow Lite ile mobil cihazlarda kullanım için sistem optimize edilebilir.

Özelleştirilebilirlik

Kullanıcı tarafından istenilen herhangi bir veri seti ile model eğitilebilir ve belirlenen ihtiyaçlara göre özelleştirilebilir.

Özetle hızlı, verimli ve doğruluk oranı yüksek bir nesne tanıma çözümü sağladığı için SSD MobileNet ve TensorFlow bu çalışmada birlikte kullanılmıştır.

SSD MobileNet V2 için veri toplama ve ön işleme adımları, sistemin doğruluğu ve güvenilirliği açısından kritik öneme sahiptir. Aşağıda, kullanılan veri setinin tanıtımı, veri toplama süreci ve ön işleme adımları ayrıntılı olarak açıklanmıştır:

Veri Seti Tanıtımı

Kullanılan veri seti, farklı ortamlarda ve farklı koşullarda çeşitli otoparklardan toplanmıştır. Her bir görüntü, otopark içerisinde konumlandırılmış birden fazla park yerini içerir. Görüntüler, farklı saatlerde, farklı hava koşullarında ve farklı ışıklandırma seviyeleri altında çekilmiştir. Kullanılan veri seti, otoparkta bulunan hem boş hem de dolu park yerlerini içerir. Ayrıca, görüntüler farklı renkteki ve tipteki araçları da içerir.

Veri Toplama Süreci

Veri toplama sürecinde, görüntüler farklı otoparklardaki park yerlerine kuş bakışı bakan bir kamera kullanılarak çekilmiştir. Kameralar, park yerlerini farklı

açılardan ve farklı yüksekliklerden gözlemlemek için stratejik konumlara yerleştirilmiştir. Veri toplama süreci sırasında, otoparklardaki araç trafiği de dikkate alınmış olup, hem dolu hem de boş park yerlerini içeren çeşitli senaryolarda görüntüler kaydedilmiştir.

Ön İşleme Adımları

Veri toplandıktan sonra, veri setinin kalitesini artırmak ve veri setini analize hazırlamak için ön işleme adımları uygulanmıştır. Ön işleme adımları, aşağıdaki maddeleri içerir:

Boyut Azaltma

Görüntülerin boyutu azaltılarak, işlem süresi ve bellek kullanımı gereksinimleri azaltılmıştır. Bu adımla, gereksiz piksellerin kaldırılması ve görüntünün boyutunun azaltılması sağlanmıştır.

Gürültü Temizleme

Görüntülerdeki gürültüyü azaltmak için filtreleme ve pürüzsüzleştirme teknikleri kullanılmıştır. Bu adımla, görüntülerdeki istenmeyen gürültüleri temizleyerek daha net ve temiz görüntüler elde etmeyi amaçlanmıştır.

Kontrast Düzenleme

Görüntülerin kontrastı ayarlanarak, daha iyi aydınlatılmış ve daha belirgin konturlara sahip görüntüler elde edilmiştir. Bu adım, park yerlerinin sınırlarını daha iyi belirlemeyi ve onları tespit etmeyi kolaylaştırmıştır.

Renk Dönüşümü

Renk dönüşümü yapılarak, görüntülerin renk dengesi ayarlanmış ve renk uyumsuzlukları giderilmiştir. Bu adım, görüntülerin farklı aydınlatma koşullarında daha tutarlı görünmesini sağlamıştır.

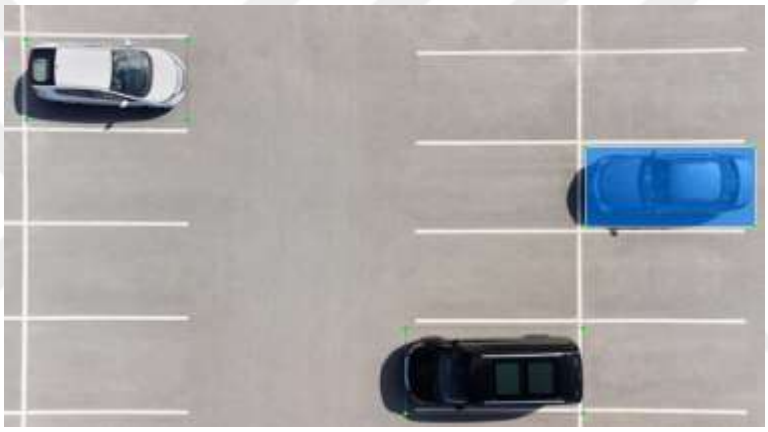
Ön işleme adımlarının tamamlanmasının ardından veri seti, park yeri durum tespit sisteminin doğruluğunu ve güvenirliğini test etmek ve değerlendirmek için kullanıma hazır hale getirilmiştir. Bu adımlar, sistemin performansını artırmak ve güvenilir sonuçlar elde etmek için gerekli olmuştur.

Bu çalışmada geliştirilen ve SSD MobilNet V2 ile Google'ın TensorFlow teknolojisini beraber kullanan park yeri durum tespit sisteminin veri toplama

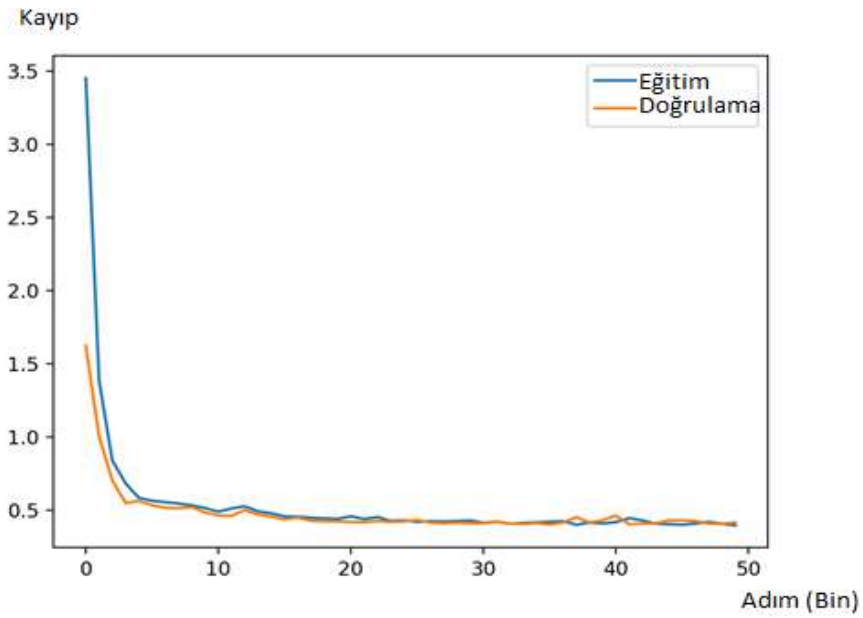
adımında, görüntüler, kare şeklinde kırpıldıktan sonra 800x600 piksel olarak yeniden boyutlandırılmıştır.

Görüntülerde bulunan araçlar etiketlenmiş ve XML formatında kaydedilmiştir. SSD MobileNet V2 mimarisi için oluşturulan XML dosyası, CSV formatına dönüştürülmüştür. Dönüştürülen CSV dosyası, derin öğrenme algoritmasına girdi olarak verilerek TF kayıtlarına dönüştürülmüştür.

TF kayıtları oluşturulduktan sonra, modelin eğitim aşamasına geçilmiştir. Eğitim aşaması, kayıp değeri yaklaşık 0.5'e veya altına düşene kadar devam ettirilmiştir. Böylece, oluşturulan model üzerinde makul bir performans elde edilmişti. Modelin eğitimi amacıyla işaretlenmiş ve etiketlenmiş bazı görüntüler, Şekil 4.11'de gösterilmiştir.



Şekil 4.11: Veri Setinden Örnek Görüntü



Şekil 4.12: İdeal Eğitim ve Doğrulama Öğrenim Eğrileri

```
INFO:tensorflow:global step 9: loss 8.8730 (3.448 sec/step)
I0719 16:54:16.180121 15184 learning.py:512] global step 9: loss = 8.8730 (3.448 sec/step)
INFO: tensorflow:global step 10: loss 8.9420 (3.459 sec/step)
I0719 16:54:19.639897 15184 learning.py:512] global step 10: loss = 8.9420 (3.459 sec/step)
INFO: tensorflow: global step 11: loss 8.2890 (3.489 sec/step)
I0719 16:54:23.129682 15184 learning.py:512] global step 11: loss = 8.2890 (3.489 sec/step)
```

Şekil 4.13: Modelin Eğitilmesi

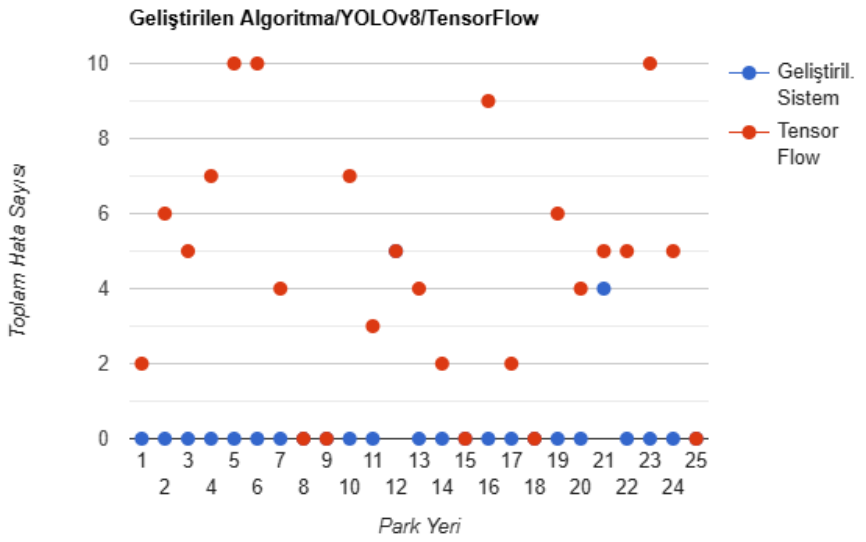
Geliştirilen sistem ve Google'ın TensorFlow nesne tespiti teknolojisini kullanan sistem, otoparklardaki çeşitli park yerlerinin boş ve dolu durumunu değerlendirmek amacıyla karşılaştırılmıştır. Bu karşılaştırma, ilk olarak 400 kare görüntü içeren gerçek dünyadan alınmış ve ücretsiz kaynak olan otopark video akışı üzerinde yapılmıştır.

Şekil 4.14: Google TensorFlow Sınıflandırması

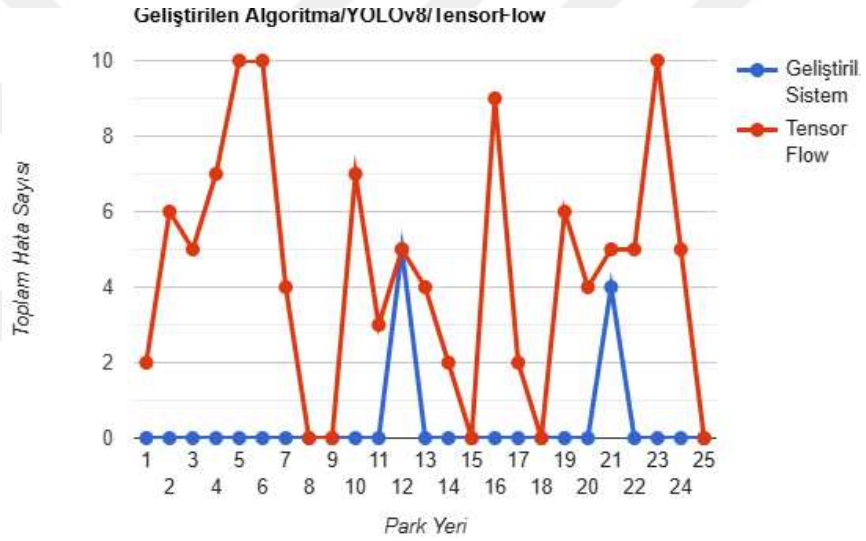
Sonuç olarak, 400 görüntü karesi içeren video akışı üzerinde yapılan bu karşılaştırmada, Google'ın TensorFlow nesne tespiti teknolojisini kullanan sistem 289 görüntü karesinde başarılı araç tespiti yaparak %72,25 başarı oranı elde etmiştir. Öte yandan, geliştirilen sistem, 391 görüntü karesinde başarılı araç tespiti yaparak %97,75 başarı oranı elde etmişti.

Çizelge 4.5: Geliştirilen Sistemin ve Google TensorFlow'un Başarısı

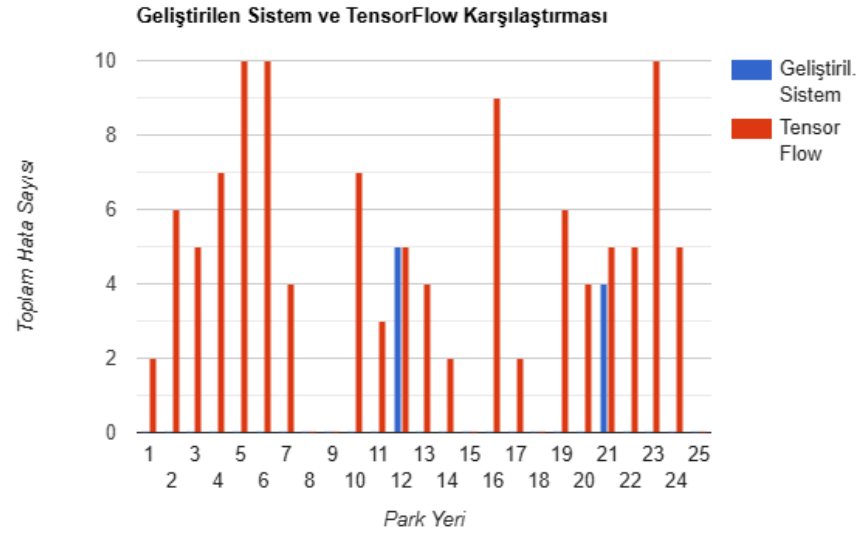
Sistem	Doğru Sınıflandırılan Kare Sayısı / Toplam Kare Sayısı	Başarı Oranı
Google TensorFlow	289/400	%72,25
Geliştirilen Sistem	391/400	%97,75



Şekil 4.15: Geliştirilen Sistem ve TensorFlow Karşılaştırması Nokta Grafiği



Şekil 4.16: Geliştirilen Sistem ve TensorFlow Karşılaştırması Çizgi Grafiği



Şekil 4.17: Geliştirilen Sistem ve TensorFlow Karşılaştırması Bar Grafiği

Google'ın TensorFlow sınıflandırma modelini kullanan nesne tespit sisteminde, iki adet dezavantaj gözlemlenmiştir. İlk dezavantaj, görüntüde bulunan bazı nesnelerin ve yayaların yanlış bir şekilde araba olarak sınıflandırılması olmuştur. Örneğin, görüntüde bulunan alışveriş arabalarının, video akışında bulunan 400 görüntü karesinin 20'sinde araba olarak sınıflandırıldığı görülmüştür.

Şekil 4.18: Google TensorFlow Tarafından Hatalı Sınıflandırma Örneği

Şekil 4.19: Geliştirilen Sistem Tarafından Doğru Sınıflandırma Örneği

Bir başka örnekte, görüntüde bulunan bazı yayaların, 4 adet görüntü karesinde araba olarak yanlış sınıflandırıldığı görülmüştür.

Şekil 4.20: Google TensorFlow Tarafından İkinci Hatalı Sınıflandırma Örneği

Şekil 4.21: Geliştirilen Sistem Tarafından İkinci Doğru Sınıflandırma Örneği

İkinci dezavantaj ise, bazı araçların görüntü karesi geçişleri sırasında stabil bir şekilde tespit edilememesi olmuştur. Bazı arabalar tespit edilip işaretlendikten sonra, bir sonraki görüntü karesinde tespit edilememekte ve işareti kaybolmaktadır. Bu durumun video akışı boyunca tekrar ettiği durumda, araçlar için işaretlemeler stabil olmamakla birlikte, işaretlemelerin bir kaybolup bir gözüktüğü gözlemlenmiştir.

a.) TensorFlow Tespiti b) TensorFlow Tespit Edememe

Şekil 4.22: Google TensorFlow İstikrarsızlık Örneği

İkinci bir karşılaştırma, 679 görüntü karesi içeren gerçek dünyadan alınmış ve ücretsiz kaynak olan otopark video akışı üzerinde yapılmıştır. Bu karşılaştırmada sonuç olarak, Google'ın TensorFlow nesne tespiti teknolojisini kullanan sistem, 601 görüntü karesinde başarılı araç tespiti yaparak %88,51 başarı oranı elde etmiştir. Öte yandan geliştirilen sistem ise, 673 karede başarılı araç tespiti yaparak %99,12 başarı oranı elde etmişti.

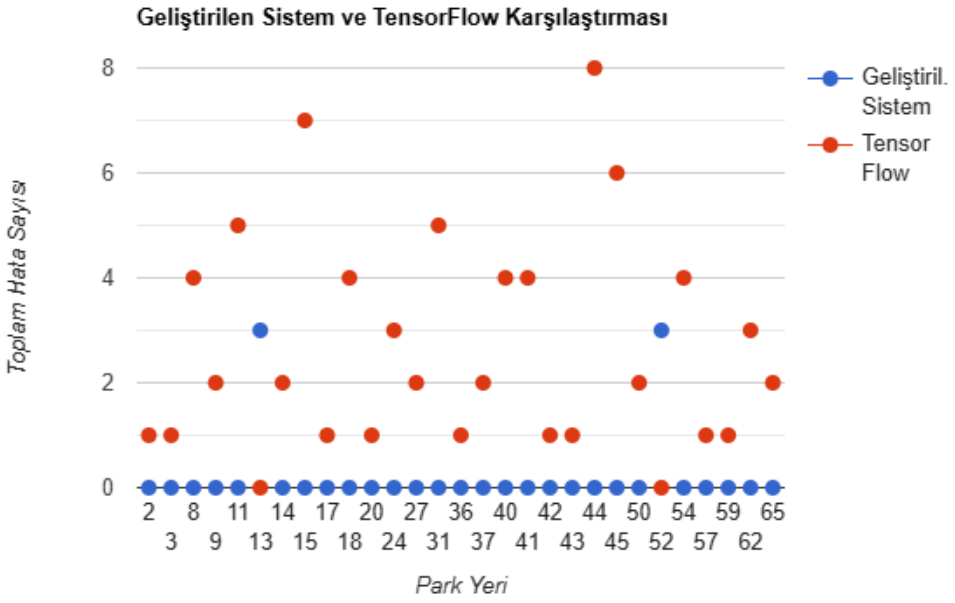
Çizelge 4.6: Geliştirilen Sistemin ve Google TensorFlow'un Başarısı (2. Video)

Sistem	Doğru Sınıflandırılan Kare Sayısı / Toplam Kare Sayısı	Başarı Oranı
Google TensorFlow	601/679	%88,51
Geliştirilen Sistem	673/679	%99,12

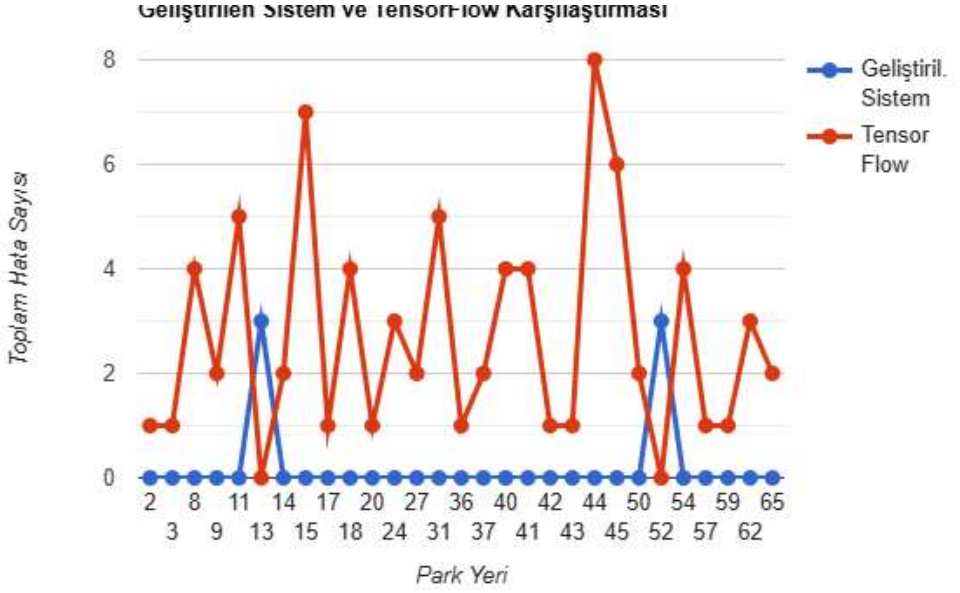
a) TensorFlow Sınıflandırması 1 (2. Video)

b) TensorFlow Sınıflandırması 2 (2. Video)

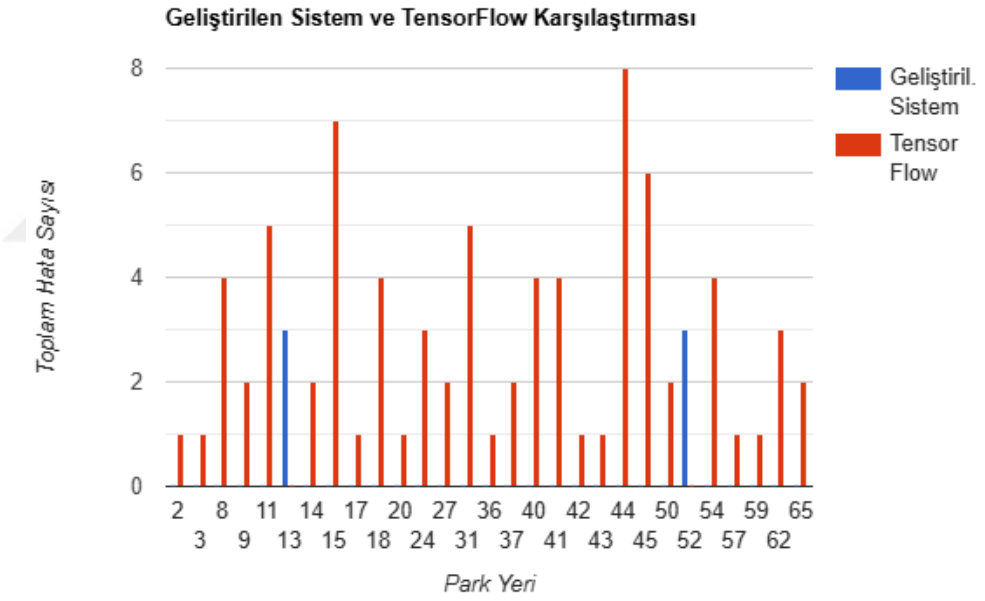
Şekil 4.23: Google TensorFlow Sınıflandırması (2. Video)



Şekil 4.24: Geliştirilen Sistem/TensorFlow Karşılaştırma Nokta Grafiği (2. Video)



Şekil 4.25: Geliştirilen Sistem/TensorFlow Karşılaştırma Çizgi Grafiği (2. Video)



Şekil 4.26: Geliştirilen Sistem/TensorFlow Karşılaştırma Bar Grafiği (2. Video)

Ayrıca, iki sistem disk üzerindeki depolama gereksinimleri açısından karşılaştırıldığında, geliştirilen sistemin, boyutu 1 KB olan yer haritası dosyası ile toplamda 7 KB disk alanı kullandığı görülmüştür.

Öte yandan, Google'ın TensorFlow nesne tespiti teknolojisini kullanan sistemin içinde bulunan, eğitilmiş modelin 60.1 MB disk alanı kullandığı görülmüştür. Ek olarak, işaretleme kodunun 4 KB ve etiket harita dosyasının 1 KB disk alanı kullandığı görülmüştür.

Sonuç olarak, geliştirilen sistem, daha az disk alanı kullanımı gerektirdiğinden avantajlıdır.

4.3 YOLOv8 ile Sınıflandırma

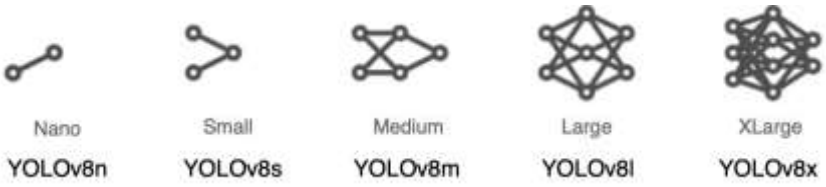
YOLO (You Only Look Once), nesne tespiti için kullanılan popüler bir derin öğrenme sistemidir. YOLOv8, benzerlerine göre daha yüksek doğruluk oranın yanında, yüksek hız ve verimlilik sunan en yeni stabil YOLO sürümdür. Gerçek zamanlı olarak, resimlerin ve video akışların içerisinde bulunan nesneleri tespit etmek ve sınıflandırmak için tasarlanmıştır.

YOLOv8, mimarisinde temel olarak evrişimsel sinir ağı kullanır. Kullanılan CNN, görüntüler gibi yapılandırılmış ızgara verilerini işlemek için özel olarak tasarlanmış bir tür derin sinir ağıdır.

YOLOv8, bir dizi evrişimsel katman, havuzlama katmanı ve doğrusal olmayan aktivasyonlar aracılığıyla görüntülerden hiyerarşik özellikler çıkarmak için CNN kullanır. YOLOv8, girdi görüntüden sınıf etiketlerini, sınırlayıcı kutuları ve diğer özellikleri doğrudan tahmin eden tek bir CNN ağı ile uçtan uca nesne tespiti yapmak için tasarlanmıştır.

Bu çalışmada, karşılaştırma yapmak için YOLOv8 teknolojisini kullanan park yeri durum tespiti sisteminin içerisinde kullanılmış olan model, video akışı içerisindeki görüntü karelerinde bulunan arabaları tanıma amacıyla eğitilmiştir.

Önceden eğitilmiş YOLOv8 modellerini karşılaştırarak, içlerinden en iyi performansı ve en yüksek doğruluk oranını sunanı seçmek amacıyla, modellerin hepsi aynı video akışı üzerinde, sınıflandırıcı algoritma ile çalıştırılmıştır. Sonuç olarak, önceden eğitilmiş YOLOv8 modelleri arasından, COCO veri seti ile önceden eğitilmiş YOLOv8m modeli bu çalışmada eğitilecek model olarak kullanılmıştır.



a) YOLOv8 Modelleri

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

b) YOLOv8 Modellerinin Teknik Farkları

Şekil 4.27: YOLOv8 Modellerinin Farklılıkları

Kaynak: Ultralytics (2024)

YOLOv8m modelinin bu çalışma için seçilmesi aşağıdaki nedenlere dayanmaktadır:

Hız ve Doğruluk Dengesi

YOLOv8m, hızlı işlem süresi ile yüksek doğruluk arasında iyi bir denge sağlar. Bu nedenle, gerçek zamanlı olarak çalışan uygulamalar için idealdir.

Geliştirilmiş Mimari

YOLOv8m modeli, önceki sürümlere kıyasla daha iyi özellik çıkarımı ve farklı nesne boyutlarıyla başa çıkma kabiliyeti sunar.

Esneklik

YOLOv8m modeli, belirli görevler veya veri setleri için ince ayar yapmaya uygundur. Böylece çeşitli özel uygulamalarda, doğruluğun artırılmasına yardımcı olur.

Dayanıklılık

YOLOv8m modeli, engellenmeler ve nesne görünümündeki değişikliklere karşı daha dayanıklı ve toleranslıdır. Böylece, çeşitli ortamlar için güvenilirlik sağlar.

Topluluk Desteği

YOLOv8m modeli, YOLO ailesinin bir parçası olarak, güçlü bir kullanıcı topluluğuna ve sürekli güncelleme desteğine sahiptir. Bu durum, uygulama geliştirme sürecinde bol miktarda kaynak ve destek bulmayı kolaylaştırır.

Uyumluluk

YOLOv8 modeli, çeşitli çerçeveler ve platformlarla iyi bir entegrasyon sağlar. Bu durum, sistemin dağıtımını kolaylaştırır.

Özetle iyi bir performans, verimlilik ve yüksek doğruluk oranı sağladığı için YOLOv8m modeli, bu çalışmada nesne tespiti için eğitilecek model olarak kullanılmıştır.

Modelin eğitim aşamasında kullanılan veri seti TensorFlow modelini eğitmek için kullanılmış veri seti ile aynıdır. Veri setinde bulunan görüntülerin dizinini ve sınıf isimlerini tutabilmek için bir veri seti YAML dosyası oluşturulmuştur. Tespit edilecek arabaların sınıf adı "car" olarak seçilmiştir.

```yaml
path: car_dataset_v8/
train: 'train/images'
val: 'valid/images'
# sınıf ismi
names:
0: 'car'
```

Şekil 4.28: YOLOv8 Model Eğitimi için YAML

YOLOv8'i özel bir veri kümesi üzerinde eğitmek için "ultralytics" paketini yüklenmiştir. Bu paket, YOLO için bir komut satırı arayüzü sağlar.

Model eğitimi için seçilen hiperparametreler: 50 epoch boyunca eğitim yapılmıştır. Böylece sınırlı eğitim ile mümümkün olan en iyi sonucın alınması hedeflenmiştir. Toplu işlem boyutu 8 olarak seçilmiştir.

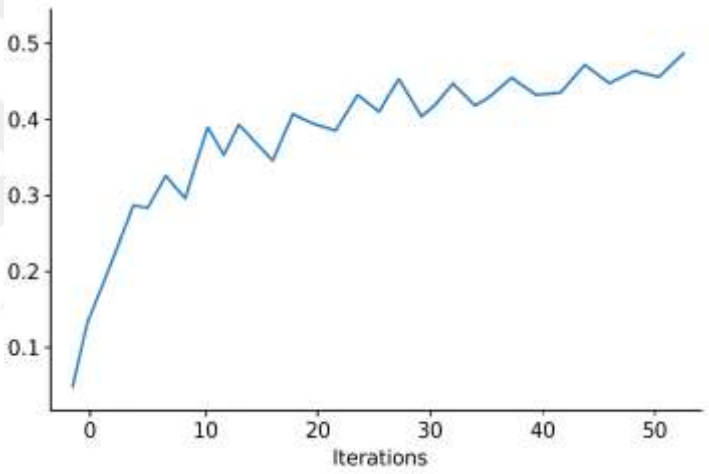
Kuş bakışı otopark görüntülerinde, arabalar olduğundan oldukça küçük boyutlarda görünebileceğinden görüntü çözünürlüğü 1280 piksel olarak belirlenmiştir. Bu durum, eğitim süresini arttırmış ancak varsayılan 640 piksel çözünürlüğe kıyasla daha iyi sonuçlar vermiştir.

Kuş bakışı otopark görüntülerindeki arabalar veri kümesi oldukça zorlu olduğundan, eğitimde ve yapılan değerlendirmelerde özellikle 0.50 IoU seviyesindeki mAP değerine odaklanılmıştır.

Aşağıda model eğitimi için YOLO komut satırı arayüzünde çalıştırılmış olan komut verilmiştir:

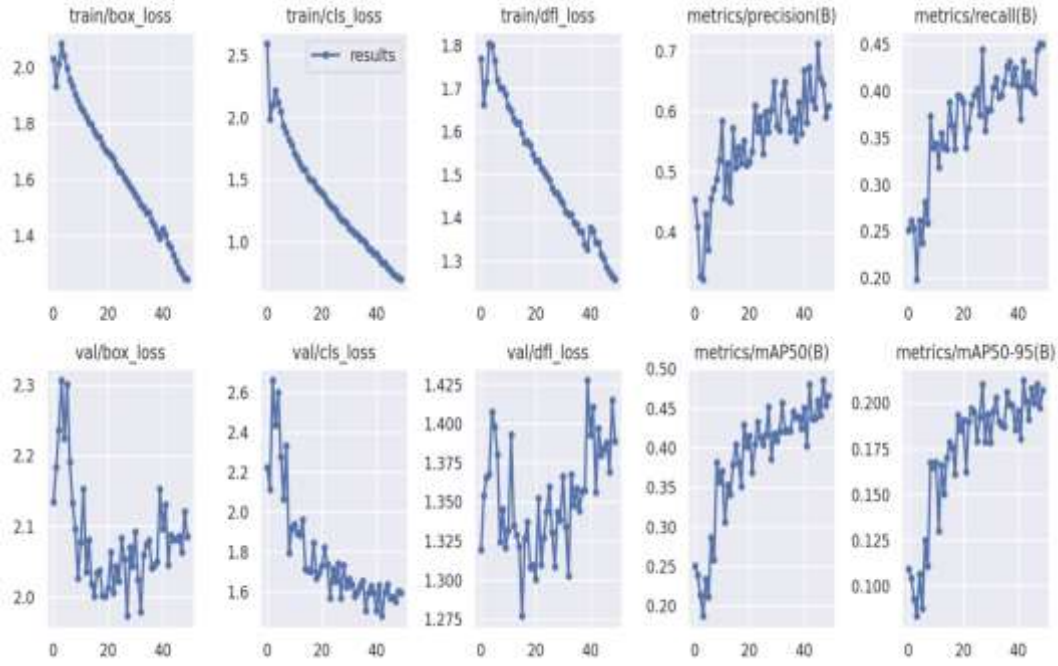
yolo task=detect mode=train model=yolov8m.pt imgsz=1280 data=car.yaml epochs=50 batch=8 name=yolov8m_car_custom

Modelin 0.50 IoU değerindeki mAP sonuçları Şekil 4.29'da net olarak görülmektedir. YOLOv8m modeli, eğitim süresi boyunca sürekli olarak gelişim göstermiştir. Böylece modelin daha uzun süre eğitilmesi, daha iyi sonuçlar elde edilmesini sağlamıştır.



Şekil 4.29: IoU 0.50 Değerinde YOLOv8m Modelinin mAP Karşılaştırması

Eğitilen YOLOv8m modeli 50 epoch içinde 48 mAP değerine ulaşmıştır. Eğitimin tamamlanmasının ardından, Şekil 4.30'deki grafikler yerel dizine kaydedilmiştir.



Şekil 4.30: YOLOv8m Modeli Veri Seti Üzerinde Eğitildikten Sonra Elde Edilen mAP ve Kayıp Grafikleri

YOLOv8 nesne tespiti teknolojisi kullanan sistem, araba olarak sınıflandırılacak nesneleri eğitilmiş model aracılığıyla video akışındaki görüntülerde tespit eder. Mevcut görüntü karesinde tespit yapıldıktan sonra, arabaları sınırlayıcı kutular ve "car" sınıf etiketini kullanarak işaretler ve aynı görüntü karesinde gösterir. Bu durum, Şekil 4.31'de gösterilmiştir.

Şekil 4.31: YOLOv8 Sınıflandırması

Park yeri haritalama dosyası kullanılarak otoparktaki kullanıcı tarafından işaretlenmiş park yerlerinin konumu belirlenir ve park yerlerinin içerisinde araba

tespit edilip edilmediğine göre park yerleri boş veya dolu olarak sınıflandırılır ve işaretlenir.

Ayrıca, otoparktaki boş ve toplam park yeri adedi görüntü karesinin sol üst köşesine yazdırılmaktadır. Bu durum, Şekil 4.32'de gösterilmiştir.

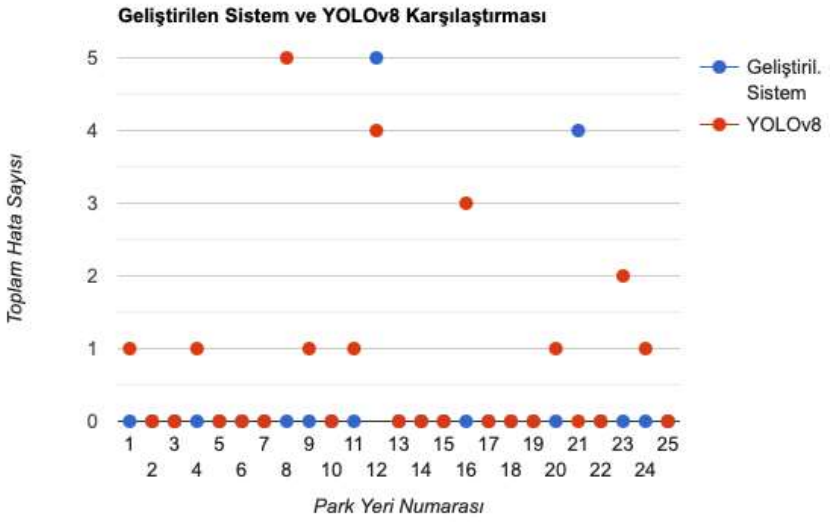
Şekil 4.32: Tespit Edilen Boş ve Dolu Park Yerleri

Geliştirilen sistem ve YOLOv8 nesne tespiti teknolojisini kullanan sistem, otoparklardaki çeşitli park yerlerinin doluluk durumunu değerlendirmek amacıyla karşılaştırılmıştır. Bu karşılaştırma, ilk olarak 400 kare görüntü içeren gerçek dünyadan alınmış ve ücretsiz kaynak olan otopark video akışı üzerinde yapılmıştır.

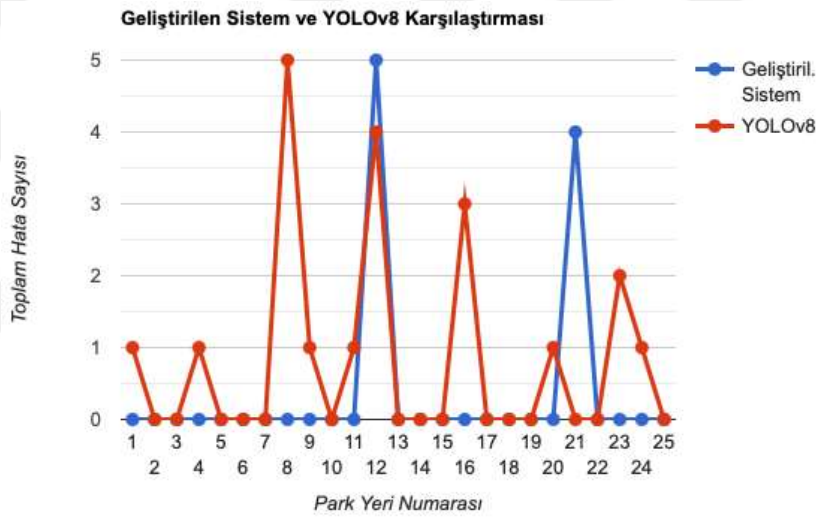
Sonuç olarak, video akışından alınan ve 400 görüntü karesi üzerinde yapılan bu karşılaştırmada, YOLOv8 nesne tespiti teknolojisini kullanan sistem 380 görüntü karesinde başarılı araç tespiti yaparak %95,00 başarı oranı elde etmiştir. Öte yandan, geliştirilen sistem ise, 391 görüntü karesinde başarılı araç tespiti yaparak %97,75 başarı oranı elde etmişti.

Çizelge 4.7: Geliştirilen Sistemin ve YOLOv8'in Başarısı

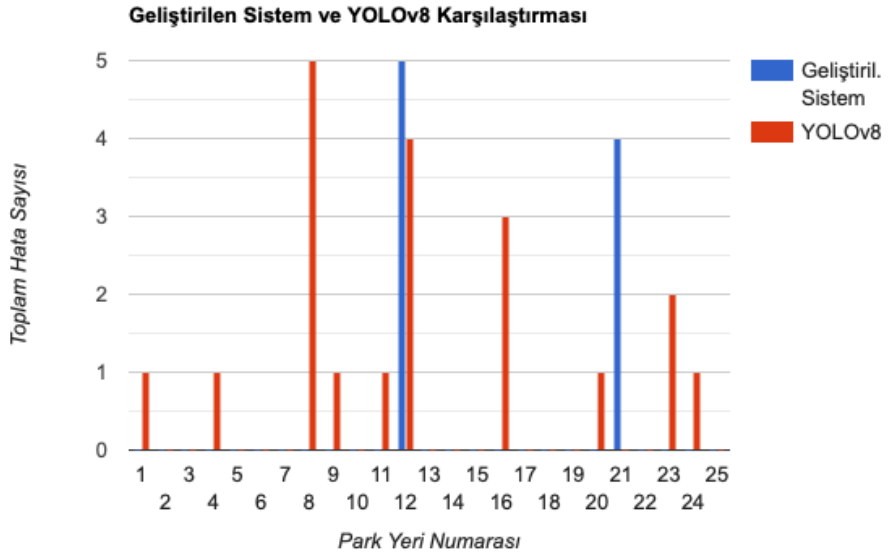
Sistem	Doğru Sınıflandırılan Kare Sayısı / Toplam Kare Sayısı	Başarı Oranı
YOLOv8	380/400	%95,00
Geliştirilen Sistem	391/400	%97,75



Şekil 4.33: Geliştirilen Sistem ve YOLOv8 Karşılaştırması Nokta Grafiği



Şekil 4.34: Geliştirilen Sistem ve YOLOv8 Karşılaştırması Çizgi Grafiği



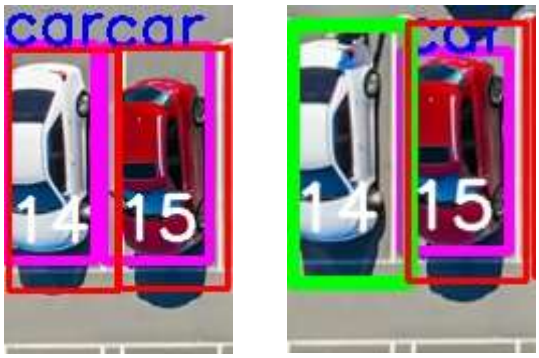
Şekil 4.35: Geliştirilen Sistem ve YOLOv8 Karşılaştırması Bar Grafiği

YOLOv8'in kullanım kolaylığı ve pratikliğine rağmen, sınıflandırma modelinde iki adet dezavantaj gözlemlenmiştir. İlk dezavantaj, görüntüde bulunan bazı nesnelerin ve yayaların yanlış bir şekilde araba olarak sınıflandırılması olmuştur. Örneğin, görüntüde bulunan alışveriş arabalarının, video akışında bulunan 400 görüntü karesinin 9'unda araba olarak yanlış sınıflandırıldığı görülmüştür.

Şekil 4.36: YOLOv8 Tarafından Yanlış Sınıflandırma Örneği

Şekil 4.37: Geliştirilen Sistem Tarafından Doğru Sınıflandırma Örneği

İkinci dezavantaj ise, bazı araçların görüntü karesi geçişleri sırasında stabil bir şekilde tespit edilememesi olmuştur. Bazı arabalar tespit edilip işaretlendikten sonra, bir sonraki görüntü karesinde tespit edilememekte ve işareti kaybolmaktadır. Bu durumun video akışı boyunca tekrar ettiği durumda, araçlar için işaretlemeler stabil olmamakla birlikte, işaretlemelerin bir kaybolup bir gözüktüğü gözlemlenmiştir.



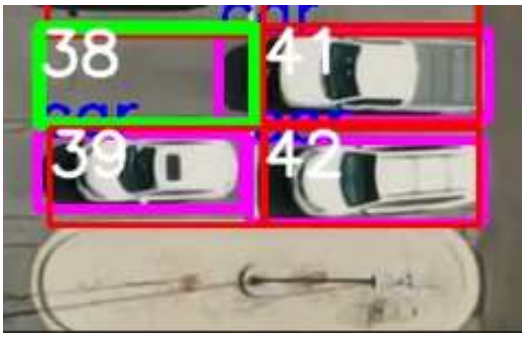
Şekil 4.38: YOLOv8 İstikrarsızlık Örneği

İkinci bir karşılaştırma, 679 görüntü karesi içeren gerçek dünyadan alınmış otopark video akışı üzerinde yapılmıştır. Bu karşılaştırmada sonuç olarak, YOLOv8 nesne tespiti teknolojisini kullanan sistem, 655 görüntü karesinde başarılı araç tespiti yaparak %96,46 başarı oranı elde etmiştir. Öte yandan geliştirilen sistem ise, 673 karede başarılı araç tespiti yaparak %99,12 başarı oranı elde etmişti.

Çizelge 4.8: Geliştirilen Sistemin ve Google TensorFlow'un Başarısı (2. Video)

Sistem	Doğru Sınıflandırılan Kare Sayısı / Toplam Kare Sayısı	Başarı Oranı
YOLOv8	655/679	%96,46
Geliştirilen Sistem	673/679	%99,12

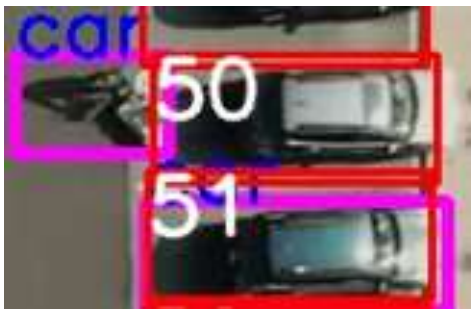
a) YOLOv8 Sınıflandırması 1 (2. Video)



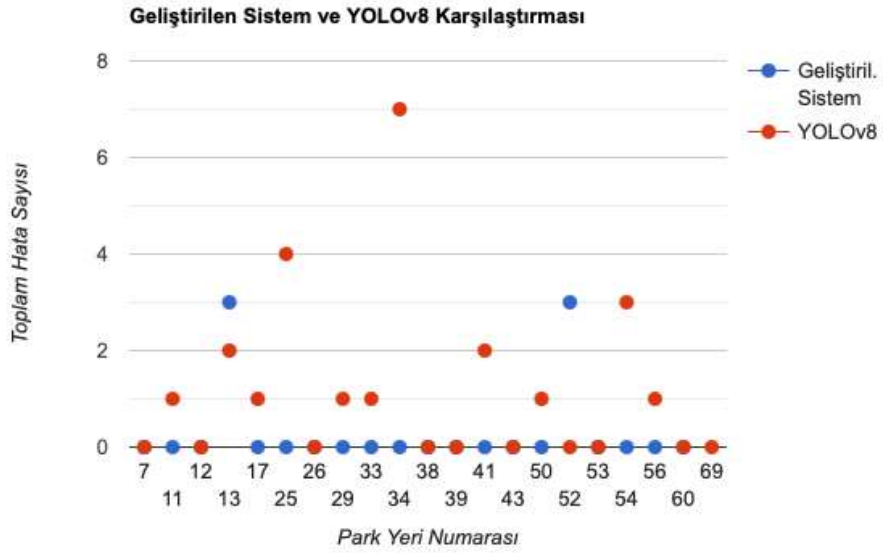
b) YOLOv8 Sınıflandırması 2 (2. Video)

Şekil 4.39: YOLOv8 Sınıflandırması (2. Video)

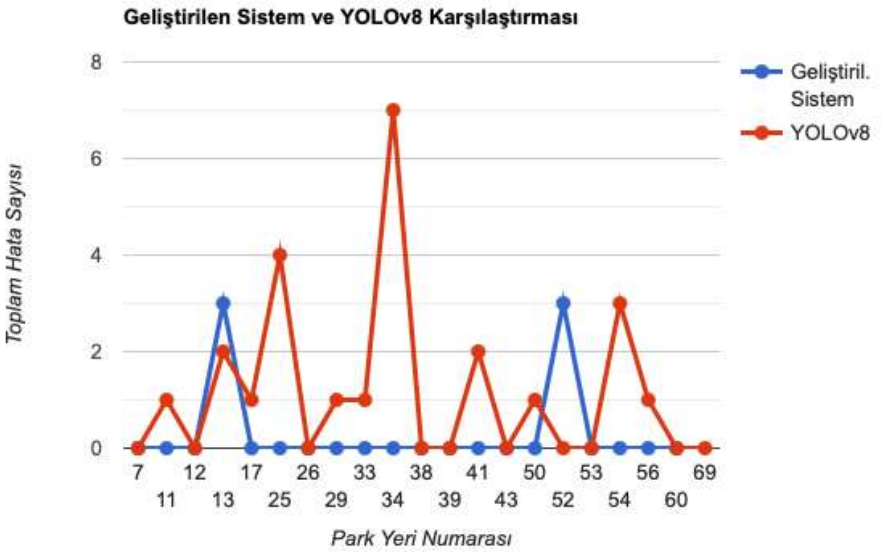
Video akışında, hareket eden bir yayanın 2 adet görüntü karesinde araba olarak yanlış sınıflandırıldığı görülmüştür.



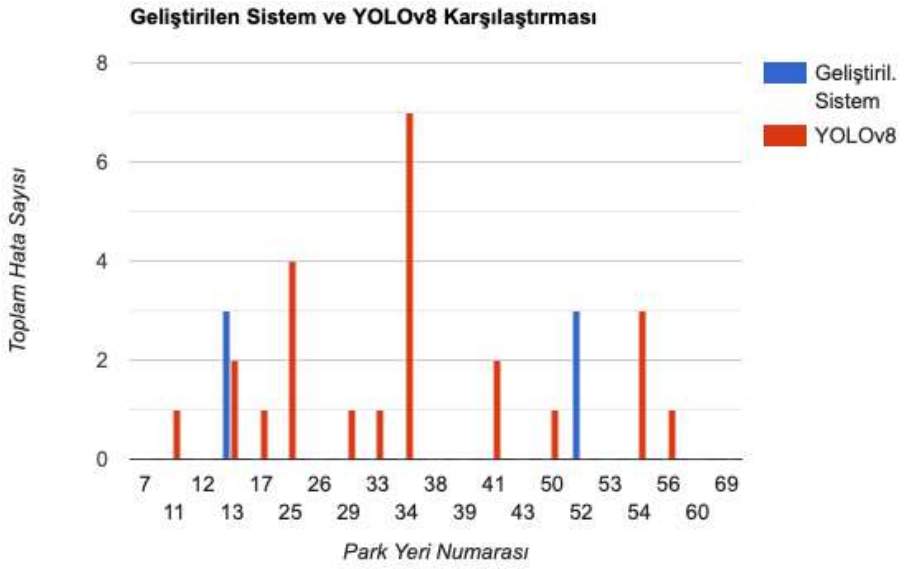
Şekil 4.40: YOLOv8 Tarafından Yanlış Sınıflandırma Örneği (2. Video)



Şekil 4.41: Geliştirilen Sistem ve YOLOv8 Karşılaştırma Nokta Grafiği (2. Video)



Şekil 4.42: Geliştirilen Sistem ve YOLOv8 Karşılaştırma Çizgi Grafiği (2. Video)



Şekil 4.43: Geliştirilen Sistem ve YOLOv8 Karşılaştırma Bar Grafiği (2. Video)

Ayrıca, iki sistem disk üzerindeki depolama gereksinimleri açısından karşılaştırıldığında, geliştirilen sistemin, toplamda 7 KB disk alanı kullandığı görülmüştü.

Öte yandan, YOLOv8 nesne tespit teknolojisini kullanan sistem içerisinde bulunan, eğitilmiş YOLOv8m modelinin 41.9 MB disk alanı kullandığı görülmüştür. Ek olarak, işaretleme kodu 4 KB ve etiket harita dosyası 1 KB disk alanı kullanmaktadır. Sonuç olarak, geliştirilen sistem, daha az disk alanı kullanımı gerektirdiğinden avantajlıdır.

4.4 Performans Bulguları

Geliştirilen sistem, Google'ın TensorFlow ve YOLOv8 teknolojisini kullanan nesne tespit sistemleri, aynı donanım kullanılarak ve aynı koşullar sağlanarak performans yönünden karşılaştırılmıştır.

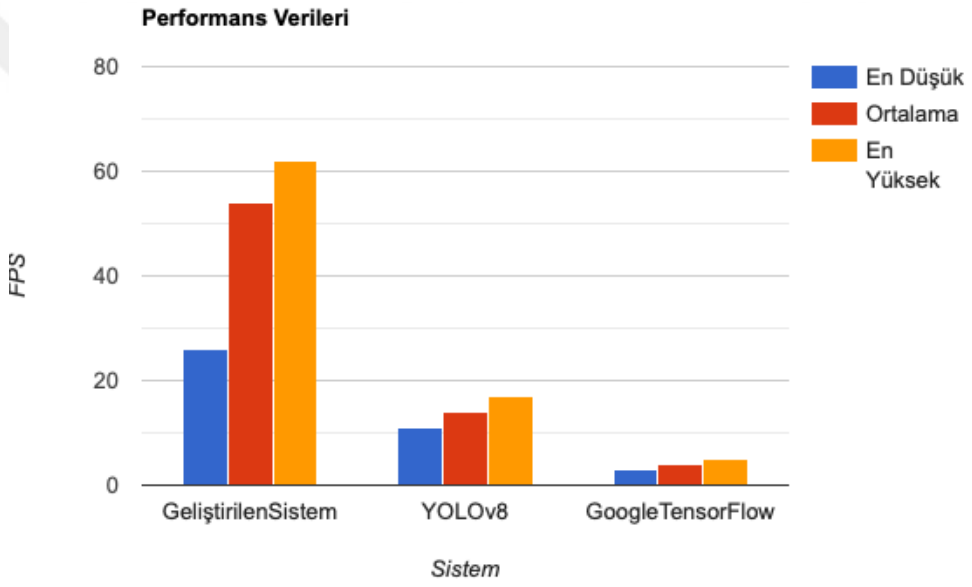
Karşılaştırma, 400 görüntü karesi içeren otopark video akışı üzerinde yapılmıştır. Sonuçlar incelendiğinde ilk olarak, geliştirilen sistemin, en düşük 26 FPS, ortalama olarak 54,28 FPS ve en yüksek 62 FPS değerlerine ulaştığı görülmüştür.

İkinci olarak, YOLOv8 teknolojisini kullanan sistemin, en düşük 11 FPS, ortalama olarak 14,83 FPS ve en yüksek 17 FPS değerlerine ulaştığı görülmüştür.

Son olarak, Google'ın TensorFlow teknolojisini kullanan sistemin, en düşük 3 FPS, ortalama olarak 4,42 FPS ve en yüksek 5 FPS değerlerine ulaştığı görülmüştür.

```
Geliştirilen Sistem
Ortalama FPS: 54.28 Min. FPS: 26 Max FPS: 62
YOLOv8
Ortalama FPS: 14.83 Min. FPS: 11 Max FPS: 17
Google TensorFlow
Ortalama FPS: 4.42 Min. FPS: 3 Max FPS: 5
```

Şekil 4.44: FPS Bazında Performans Sonuçları



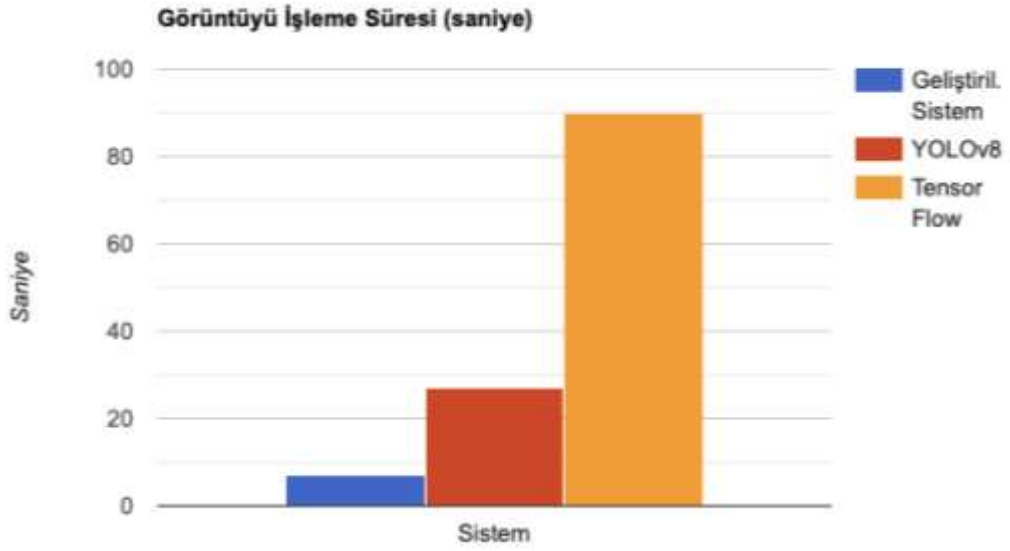
Şekil 4.45: Üç Sistemin Performans Karşılaştırması Bar Grafiği

Bu verilerden yola çıkılarak, performans değerlendirmesi yapılacak olursa;

İlk olarak, geliştirilen sistem ortalama olarak saniyede 54,28 adet görüntü karesi işleyebildiğinden, 400 görüntülük video akışını yaklaşık olarak 400/54,28=7,4 saniyede işleyip, bitirebildiği sonucu çıkmıştır.

İkinci olarak, YOLOv8 teknolojisini kullanan sistem, ortalama olarak saniyede 14,83 adet görüntü karesi işleyebildiğinden, video akışını yaklaşık olarak 400/14,83=26,97 saniyede işleyip, bitirebildiği sonucu çıkmıştır.

Son olarak, Google'ın TensorFlow teknolojisini kullanan sistem ise ortalama olarak saniyede 4,42 adet görüntü karesi işleyebildiğinden, aynı video akışını yaklaşık olarak 400/4,42=90,49 saniyede işleyip, bitirebildiği sonucu çıkmıştır.



Şekil 4.46: Üç Sistemin İşleme Süresi Bar Grafiği

Elde edilen sonuçlardan yola çıkılarak, 400 görüntülük video akışı üzerinde, geliştirilen sistemin, daha yüksek FPS değerlerine eriştiği ve saniye bazında video akışını işlemede, YOLOv8 ve Google'ın TensorFlow teknolojilerini kullanan sistemlere kıyasla daha performanslı ve hızlı olduğu görülmüştür.

Performans değerlendirmeleri ve karşılaştırmalar, adı "Apple 2025 M4 Macbook Air 13 inç" olan sistem üzerinde yapılmıştır. Farklı donanıma sahip sistemlerde ve farklı koşullarda, sonuçların daha farklı çıkabileceği göz önünde bulundurulmalıdır.

5. SONUÇ VE ÖNERİLER

Bu çalışma kapsamında, gerçek zamanlı olarak park yeri durum tespiti gerçekleştirebilen, görüntü işleme tabanlı bir sistem başarıyla geliştirilmiş ve değerlendirilmiştir. Geliştirilen sistemde Python dili ve OpenCV kütüphanesi birlikte kullanılarak görüntü işleme süreci yapılandırılmıştır. Gri tonlamalı görüntüye çevirme, Gauss bulanıklaştırma, uyarlanabilir eşikleme, medyan bulanıklığı ve morfolojik işlemler gibi teknikler ile park alanlarının doluluk durumu analiz edilmiştir. Görüntüler, kullanıcı tarafından seçilen park yeri bölgelerindeki siyah ve beyaz piksellerin yoğunluğu ve değişimlerine göre analiz edilerek park yerlerinin dolu veya boş olduğunun sınıflandırılması yapılmıştır.

Ek olarak, gerçekleştirilen deneysel çalışmalar kapsamında, geliştirilen sistem ile Google TensorFlow ve YOLOv8 tabanlı sistemlerin performansları, iki farklı video akışı üzerinde karşılaştırmalı olarak değerlendirilmiştir. Yapılan analizlerde, her bir sistemin doğru sınıflandırdığı kare sayısı ve buna karşılık gelen başarı oranları belirlenmiştir.

Birinci video akışında, geliştirilen sistem 400 kareden 391'ini doğru sınıflandırarak %97,75 başarı oranı elde etmiştir. Aynı video akışı üzerinde Google TensorFlow tabanlı sistem 289 kareyi doğru sınıflandırarak %72,25, YOLOv8 ise 380 kareyi doğru sınıflandırarak %95,00 başarı oranına ulaşmıştır.

İkinci video akışında ise geliştirilen sistem, 679 kareden 673'ünü doğru şekilde sınıflandırmış ve %99,12'lik yüksek bir başarı oranı sergilemiştir. Bu akışta, Google TensorFlow 601 doğru sınıflandırma ile %88,51, YOLOv8 ise 655 doğru sınıflandırma ile %96,46 oranında başarı sağlamıştır.

Elde edilen sonuçlar, geliştirilen sistemin hem birinci hem de ikinci video akışında diğer iki sisteme kıyasla daha yüksek sınıflandırma doğruluğuna sahip olduğunu ve genel olarak daha başarılı bir performans sergilediğini ortaya koymaktadır. Bu sonuçlar, geliştirilen sistemin gerçek zamanlı olarak otopark izleme uygulamalarında etkili bir şekilde kullanılabileceğini göstermektedir.

Ancak, sistemin bazı sınırlamaları da vardır. Özellikle, belirli hava koşulları veya ışıklandırma durumları altında performansı düşebilir. Bu sınırlamalar, gelecekteki çalışmaların odak noktalarından biri olabilir. Örneğn, histogram eşitleme veya CLAHE gibi tekniklerle ışık değişimlerine karşı dayanıklılığın artırılması sağlanabilir.

Ayrıca gelecek çalışma önerileri kapsamında, geliştirilen sistemin daha geniş veri setleriyle test edilmesi, farklı hava koşullarında ve ışıklandırma altında performansının değerlendirilmesi olabilir. Farklı açılardan, değişen çevresel koşullarda ve çeşitli otopark türlerinde toplanacak daha geniş ve çeşitli veri setleri ile çalışarak sistemin genellenebilirliği artırılabilir.

Geliştirilen sistemin derin öğrenme teknikleri ile entegrasyonu gibi konular üzerinde durulabilir. Örenğin geliştirilen sisteme, CNN tabanlı modellerin entegre edilmesi, sınıflandırma başarımını artırabilir. Özellikle araç tipi ve park yönü gibi ek bilgilerin de sınıflandırılabilmesi sağlanabilir. Ayrıca, sistemin performansını daha da artırabilir ve gerçek dünyada uygulanabilirliğini daha da geliştirebilir.

Geliştirilen sistem, IoT tabanlı yapılarla entegre edilerek merkezi bir sunucuya veri gönderimi sağlayabilir. Bu sayede mobil uygulamalar ya da web arayüzleri üzerinden kullanıcıya anlık bilgi sunulabilir.

Görüntü işleme dışında, navigasyon trafik verileri veya yol yüzeyi sensörler entegre edilerek, gelen bilgilerle park yeri uygunluğu hakkında daha bağlamsal tahminler yapılabilir.

Sonuç olarak, geliştirilen sistem, gerçek zamanlı olarak park yerlerinin durum tespiti için etkili bir çözüm sunmaktadır. Yüksek doğruluk, hızlı işleme hızı ve gerçek zamanlı uygulanabilirlik, sistemin pratik kullanımını desteklemektedir. Geleneksel yöntemlerle kıyaslandığında, işlem süresi ve kaynak tüketimi açısından daha verimli olduğu görülmüş, gerçek zamanlı uygulamalar için güçlü bir aday olduğu kanıtlanmıştır.

Gelecekte yapılacak çalışmalar, geliştirmeler ve iyileştirmelerle birlikte, bu çalışmada geliştirilen sistemin akıllı şehir uygulamalarında yaygın olarak kullanılabilecek modüler bir çözüme dönüşmesi mümkündür ve bu çalışma, otoparklardaki park yerlerinin durumunun tespiti konusunda kullanılan teknolojinin iyileştirilmesine katkı sağlayabilir.

KAYNAKLAR

[1] INRIX. (2024). “Global Traffic Scorecard 2024”. Erişim Adresi: https://inrix.com/scorecard/

[2] Conure Technology Solutions. (2025). “Smart Parking Solutions: Urban Mobility”. Erişim Adresi: https://www.conurets.com/how-smart-parking-solutions-are-transforming-urban-mobility-in-2025/

[3] G. Revathi and V. R. S. Dhulipala. (2012). “Smart parking systems and sensors: A survey”. International Conference on Computing, Communication and Applications.

[4] Muhamad Muzhafar Abd Kadir, Mohd Nizam Osman, Nor Arzami Othman, Khairul Anwar Sedek. (2020). “IoT based Car Parking Management System using IR Sensor”. Journal of Computing Research and Innovation (JCRINN) Vol. 5 No. 2 eISSN: 2600-8793

[5] S. A. Shaheen, C. J. Rodier, A. M. Eaken. (2005). “Smart parking management field test: A bay area rapid transit (bart) district parking demonstration”.

[6] Moldovan Mircea, Filip Nicolae. (2014). "Increasing of the urban traffic surveillance byautomatic information device". DOI: 10.2478/s13531-013-0152-3

[7] Mark Braibanti. (2017). “Our Newest Innovation: Ultrasonic Sensor Parking Availability Technology”. Erişim Adresi: https://inrix.com/blog/ultrasonic-sensor-parking-availability-technology/

[8] Pngtree. (2023). “Bird S Eye View Of A Textured Car Parking Lot Captured By An Aerial Drone Stunning Background Image”. Erişim Adresi: https://pngtree.com/freebackground/bird-s-eye-view-of-a-textured-car-parking-lot-captured-by-an-aerial-drone-stunning-background-image_13409550.html

[9] De Almeida, P. R., Oliveira, L. S., Britto, A. S., Silva, E. J., & Koerich, A. L. (2015). “PKLot–A robust dataset for parking lot classification”. Expert Systems with Applications, 42(11), 4937-4949.

[10] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo. (2016). “Car parking occupancy detection using smart camera networks and Deep Learning”. 2016 IEEE Symposium on Computers and Communication (ISCC).

[11] Nicholas True. (2007). “Vacant parking space detection in static images”. University of California, San Diego, 17. Erişim Adresi: https://cseweb.ucsd.edu/classes/wi07/cse190-a/reports/ntrue.pdf

[12] Debaditya Acharya, Weilin Yan, Kourosh Khoshelham. (2018). "Real-time image-based parking occupancy detection using deep learning". Proc. of the

5th Annual Conference of Research@Locate. Erişim Adresi: https://ceur-ws.org/Vol-2087/paper5.pdf

[13] Yusufbek Yuldashev, Mukhriddin Mukhiddinov, Akmalbek Bobomirzaevich Abdusalomov, Rashid Nasimov, Jinsoo Cho. (2023). "Parking Lot Occupancy Detection with Improved MobileNetV3". DOI: 10.3390/s23177642. Erişim Adresi: https://pmc.ncbi.nlm.nih.gov/articles/PMC10490723/

[14] Ronu Skariah. (2022). "Effective Image-Based Parking Occupancy Detection using Masked Region Based Convolutional Neural Network". Erişim Adresi: https://norma.ncirl.ie/6668/1/ronuskariah.pdf

[15] Nevit Dilmen. (2012). "Demonstration of how RGB image split into its three RGB channels". Erişim Adresi: https://commons.wikimedia.org/wiki/File:Beyoglu_4671_tricolor.png

[16] OpenCV Documentation Index. Erişim Adresi: https://docs.opencv.org/

[17] Python Offical Website. Erişim Adresi: https://www.python.org/

[18] jhorrocks. (2022). "Busy Parking Lot 4k Aerial Time Lapse - 4K Stok video". Erişim Adresi: https://www.istockphoto.com/tr/video/busy-parking-lot-4k-aerial-time-lapse-gm1370353417-439924325

[19] Google's TensorFlow Offical Website. Erişim Adresi: https://www.tensorflow.org/

[20] Sabina Pokhrel. (2020). "Real-Time Vehicle Detection with MobileNet SSD and Xailient". Erişim Adresi: https://xailient.com/blog/real-time-vehicle-detection-with-mobilenet-ssd-and-xailient/

[21] Ultralytics YOLOv8 Offical Website. Erişim Adresi: https://docs.ultralytics.com/models/yolov8/

[22] D.B.L. Bong, K.C. Ting, K.C. Lai. (2008). "Integrated Approach in the Design of Car Park Occupancy Information System (COINS)". IAENG International Journal of Computer Science 35(1).

[23] M.Y.I. Idris, E.M. Tamil, Z. Razak, N.M. Noor, L.W. Kin. (2009). "Smart Parking System using Image Processing Techniques in Wireless Sensor Network Environment". Information Technology Journal, 8: 114-127. DOI: 10.3923/itj.2009.114.127.

[24] Chihping Hsu, Toshimitsu Tanaka, Noboru Sugie, Kouzi Ueda. (2002). "Locating Vehicles in a Parking Lot by Image Processing". IAPR Workshop on Machine Vision Applications. Erişim Adresi: https://www.mva-org.jp/Proceedings/CommemorativeDVD/2002/papers/2002475.pdf

[25] H. Ichihashi, A. Notsu, K. Honda, T. Katada, M. Fujiyoshi. (2009). "Vacant parking space detector for outdoor parking lot by using surveillance camera and FCM classifier". IEEE International Conference on Fuzzy Systems. DOI: 10.1109/fuzzy.2009.5277099.

[26] Marc Tschentscher, Christian Koch, Markus König, Jan Salmen, Marc Schlipsing. (2015). "Scalable realtime parking lot classification: An evaluation of image features and supervised learning algorithms".

International Joint Conference on Neural Networks (IJCNN). DOI: 10.1109/ IJCNN.2015.7280319

[27] S. Valipour, M. Siam, E. Stroulia, M. Jagersand. (2016) “Parking-stall vacancy indicator system, based on deep convolutional neural networks”. IEEE 3rd World Forum on Internet of Things (WF-IoT). DOI: 10.1109/ WF-IoT.2016.7845408

[28] Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, Claudio Gennaro, Carlo Meghini, Claudio Vairo. (2016). “Deep Learning for Decentralized Parking Lot Occupancy Detection”. Expert Systems with Applications 72. DOI: 10.1016/j.eswa.2016.10.055