Journal of the Faculty of Engineering and Architecture of Gazi University 38:3 (2023) 1867-1878

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University
Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

A real-time simulation environment architecture for autonomous vehicle design

Yusuf Özçevik[1]*, Özgür Solmaz[2], Eşref Baysal[2], Mert Ökten[2]

[1]Department of Software Engineering, Hasan Ferdi Turgutlu Faculty of Technology, Manisa Celal Bayar University, 45400, Manisa, Türkiye
[2]Department of Energy Systems Engineering, Hasan Ferdi Turgutlu Faculty of Technology, Manisa Celal Bayar University, 45400, Manisa, Türkiye

Highlights:

- A simulation architecture was prepared with the Unity framework to test an autonomous driving model
- An autonomous driving model that follows road lanes and recognizes traffic signs is proposed
- In the experiments repeated with different versions of YOLO and R-CNN algorithms, the performance of the proposed system was analyzed

Keywords:

- Autonomous vehicle
- Object recognition algorithm
- Simulation environment architecture
- Unity framework

Article Info:

Research Article
Received: 30.11.2021
Accepted: 07.09.2022

DOI:

10.17341/gazimmfd.1030482

Acknowledgement:

The authors would like to thank Manisa Celal Bayar University Scientific Research Projects Coordination Unit with project code 2021-065.

Correspondence:

Author: Yusuf Özçevik
e-mail: yusuf.ozcevik@cbu.edu.tr
phone: +90 505 885 3973

Graphical/Tabular Abstract

The study presents a real-time simulation architecture for autonomous vehicle design. For this purpose, a simulation infrastructure is developed by using Unity framework, Robot Operating System and Turtlebot. Moreover, an autonomous vehicle model is prepared with real-time lane tracking and traffic sign detection sub-systems. As a result, an autonomous vehicle simulation given in Figure A is obtained without any real-world hardware costs for designing an autonomous vehicle model. The main contribution of this study is to present the experimental results on the usability of the proposed simulation architecture.

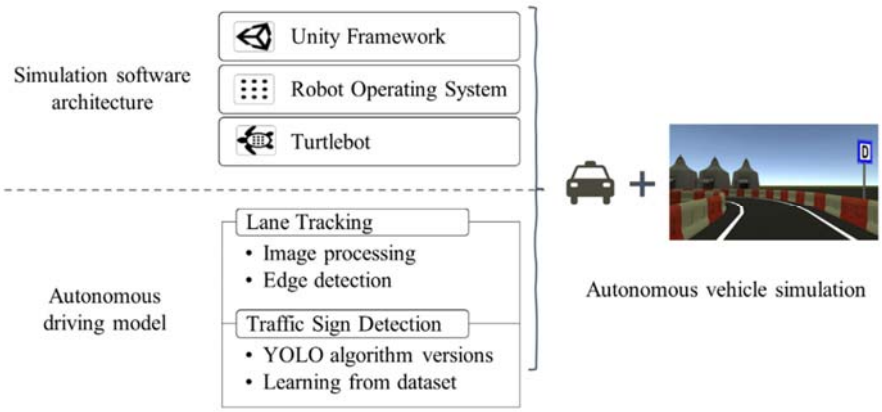


Figure A. The proposed simulation architecture for an autonomous vehicle design

Purpose:
This study aims to show the necessity of hardware-free simulation tools for autonomous vehicle design with a manageable production process in terms of time and hardware costs. To this end, a real-time simulation environment architecture is presented, including a set of required components and the feasibility of the proposed architecture is examined through the experiments conducted.

Theory and Methods:
To investigate the feasibility of the proposed real-time simulation architecture, an autonomous driving model, including lane tracking and traffic sign detection is introduced. Canny edge detection algorithm and different YOLO and R-CNN versions are deployed for lane tracking and traffic sign detection, respectively. The simulation architecture is tested with the components of the proposed autonomous driving model.

Results:
The proposed simulation architecture is evaluated on the accuracy rate of the object detection algorithm deployed for each simulation run. For this purpose, YOLO-v3, YOLO-v3 tiny, YOLO-v4, YOLO-v4 tiny, Fast R-CNN, Faster R-CNN, and Mask R-CNN algorithms are considered through the experiments. A street traffic sign dataset for Turkey is utilized for model training. According to the evaluation results, YOLOv4 is noted as the model that produces the highest accuracy rate with 95% throughout the evaluation.

Conclusion:
The evaluation results obtained from the simulation environment can be claimed as successful enough to use the proposed simulation architecture with different autonomous driving models for an autonomous vehicle design process without any real-world hardware cost.

Journal of the Faculty of Engineering and Architecture of Gazi University 38:3 (2023) 1867-1878

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Otonom araç tasarımı için gerçek zamanlı benzetim ortamı mimarisi

Yusuf Özçevik[1]*, Özgür Solmaz[2], Eşref Baysal[2], Mert Ökten[2]

[1]Manisa Celal Bayar Üniversitesi, Hasan Ferdi Turgutlu Teknoloji Fakültesi, Yazılım Mühendisliği Bölümü, 45400, Manisa, Türkiye
[2]Manisa Celal Bayar Üniversitesi, Hasan Ferdi Turgutlu Teknoloji Fakültesi, Enerji Sistemleri Mühendisliği Bölümü, 45400, Manisa, Türkiye

ÖNEÇIKANLAR

- Otonom sürüş modeli sınaması için Unity kütüphanesi ile modüler bir benzetim ortamı mimarisi hazırlanmıştır
- Şerit takibi yapan ve trafik işareti tanıyan bir otonom sürüş modeli önerilmiştir
- YOLO ve R-CNN evrişimsel sinir ağları algoritmaları önerilen sistem üzerinde analiz edilerek sonuçlar paylaşılmıştır

Makale Bilgileri
Araştırma Makalesi
Geliş: 30.11.2021
Kabul: 07.09.2022

DOI:
10.17341/gazimmfd.1030482

Anahtar Kelimeler:
Benzetim ortamı mimarisi, nesne tanıma algoritması, otonom araç, unity kütüphanesi

ÖZ

Otonom sürüş için önerilen çeşitli yaklaşımlar temelde bir görüntü işleme ve bir makine öğrenmesi sürecini içermektedir. Bu yaklaşımlarda uygun görüntü işleme tekniklerinin ve kapsamlı bir veri setinin kullanılması son derece önemlidir. Bununla birlikte, önerilen modelin gerçek zamanlı çalışması gerekir. Öte yandan, bir otonom araç modelinin tasarlanması ve imalatı ciddi donanım maliyetleri ile sonuçlanmaktadır. Ayrıca, yeni yaklaşımların geliştirilmesi için tasarım ve imalat süreçlerinin tekrarlanması gerekmektedir. Bu bağlamda, gerçek zamanlı bir benzetim mimarisinden faydalanmak, modelin daha az maliyetle bir ön doğrulaması için uygun bir yaklaşımdır. Bu çalışmada, bir otonom sürüş modelini sınamak üzere, Unity kütüphanesi ile gerçek zamanlı bir benzetim ortamı mimarisi önerilmektedir. Ayrıca, şerit takip ve nesne tanıma yaklaşımları içeren bir otonom sürüş modeli tanıtılarak, bir otonom araç benzetimi oluşturulmaktadır. Son olarak, evrişimsel sinir ağları tabanlı YOLO algoritması ve R-CNN algoritması versiyonları ile önerilen benzetim mimarisinin uygulanabilirliği sınanmaktadır. Elde edilen değerlendirme bulgularına göre, Faster R-CNN, Mask R-CNN ve YOLO-v4 algoritmalarının sırasıyla %91, %93 ve %95 doğruluk oranı ile sonuçlar ürettiği gözlemlenmektedir. Bu sonuçların, literatürde yer alan farklı trafik işareti veri setleri üzerinde elde edilen başarım oranlarına yakın olduğu tespit edilmiştir. Elde edilen bulgular göz önüne alındığında, otonom sürüş modeline sahip bir araç benzetiminin, önerilen sistem mimarisinde başarılı bir şekilde sınandığı savunulmaktadır.

A real-time simulation environment architecture for autonomous vehicle design

HIGHLIGHTS

- A modular simulation architecture was prepared with the Unity framework to test an autonomous driving model
- An autonomous driving model that follows road lanes and recognizes traffic signs is proposed
- The performance of the proposed system was analyzed on YOLO and R-CNN convolutional neural network algorithms

Article Info
Research Article
Received: 30.11.2021
Accepted: 07.09.2022

DOI:
10.17341/gazimmfd.1030482

Keywords:
Autonomous vehicle, object recognition algorithm, simulation environment architecture, unity framework

ABSTRACT

Various proposed approaches for autonomous driving basically involve an image processing and a machine learning process. It is extremely important to use appropriate image processing techniques and a comprehensive data set in these approaches. Moreover, the proposed model must work in real-time. On the other hand, designing and manufacturing an autonomous vehicle model results in serious hardware costs. In addition, the design and manufacturing processes need to be repeated to develop new approaches. In this context, utilizing a real-time simulation environment can be seen as a suitable approach for a less costly pre-validation of such models. In this study, a real-time simulation architecture is developed with Unity framework to test an autonomous driving model. In addition, an autonomous driving model that includes lane tracking and object recognition approaches is proposed, and an autonomous vehicle simulation is created. Finally, the feasibility of the proposed simulation architecture is tested with the convolutional neural networks-based YOLO algorithm and R-CNN algorithm versions. According to the findings, it is observed that Faster R-CNN, Mask R-CNN and YOLO-v4 algorithms produce results with 91%, 93% and 95% accuracy, respectively. It has been determined that these results are close to the accuracy rates obtained on different traffic sign data sets in the literature. Considering the outcomes, it is argued that a vehicle simulation with an autonomous driving model has been successfully tested in the proposed system architecture.

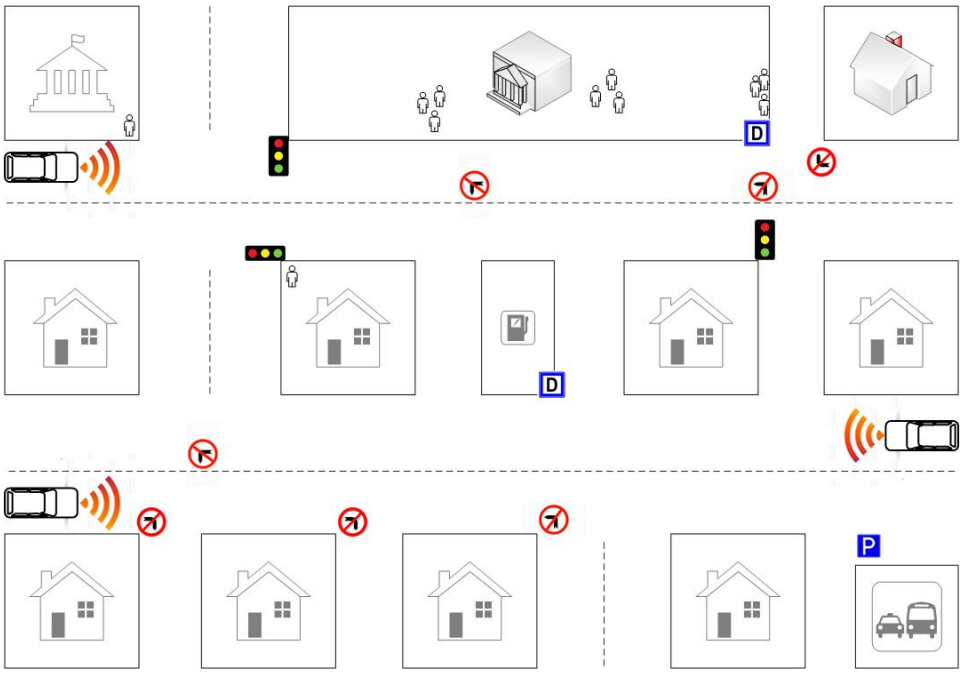
*Sorumlu Yazar/Yazarlar / Corresponding Author/Authors : *yusuf.ozcevik@cbu.edu.tr, ozgur.solmaz@cbu.edu.tr, esref.baysal@cbu.edu.tr, mert.okten@cbu.edu.tr / Tel: +90 505 885 3973

1. Giriş (Introduction)

Otonom araçlar, son yıllarda, endüstriyel ve akademik açıdan sıklıkla ele alınan konulardan biridir [1-3]. Otonom araçlarda temel prensip, dış dünyadan toplanan veriyi işleyerek, bir karar mekanizması oluşturmak ve hedeflenen amaçları gerçekleştirmektir. Örnek bir otonom araç topolojisi Şekil 1'de verilmektedir. Şekilde, kentsel bir alanda yer alan binalar, yol şeritleri, trafik ışıkları, trafik işaretleri, yayalar ve araçlar yer almaktadır. Böyle bir otonom araç topolojisinde yer alan otonom araçlar, temel olarak veri toplayıcı ve veri işleyici birimlerden oluşur. Veri toplayıcılar, bir görüntü ya da uzaklık işleyici ile çevredeki verileri toplamakta; veri işleyiciler ise, bu verileri anlamlandırarak otonom sürüş için bir karar mekanizması oluşturmaktadır.

Otonom araç topolojilerinde bulunan trafik işaretleri, şeritler, yayalar vb. gibi çevresel varlıklar araç tarafından tanınmalı; bu varlıklara ilişkin elde edilen veriler güvenli bir sürüş için uygun şekilde işlenmelidir. [4]'te yazarlar, trafik güvenliği için nesne tanımanın önemini vurgularken; nesnelere ait niteliklerin tanınırlığı büyük oranda etkilediğini göstermektedir. Bu bağlamda, literatürde, farklı makine öğrenmesi yöntemlerinden faydalanarak nesne tespit ve tanıma modelleri önerilmektedir. [5]'te yazarlar, bir gerçek dünya senaryosunda, yapay öğrenme yöntemlerinden faydalanarak otonom bir araç önermektedir. Çalışmada önerilen modelde, evrişimsel sinir ağları ile uzun-kısa vade hafıza ağları birlikte kullanılmaktadır. [6]'da yazarlar, trafik hız sınırlama işaretlerini sınıflandırmak üzere bir nesne tanıma modeli ortaya koymaktadır. Yazarlar, önerilen nesne tanıma modelinin yüksek hızda ve yüksek doğrulukta çalışması amacıyla evrişimsel sinir ağları kullanımını önermektedir. Yapılan deneysel çalışmalara göre, önerilen model, trafik hız sınırlama işaretlerini yüksek oranda doğru tespit etmektedir. [7]'de yazarlar, araç kamerası ile elde edilen görüntülerden trafik işaret levhası tespit etmeye yönelik bir yöntem önermektedir. Önerilen yöntemde, derin öğrenme, evrişimsel sinir ağları ve istatistiksel modellerden faydalanılmaktadır. Önerilen yaklaşım ile literatürde yer alan diğer modellerin kıyaslanması neticesinde, önerilen modelin daha yüksek başarım ile trafik işaret levhalarını tespit ettiği gösterilmektedir. [8]'de yazarlar dört katmanlı bir trafik işareti tanıma modeli önermektedir. Çalışmada, önerilen modelde yer alan katmanlar ve geliştirilen yazılım detaylarıyla anlatılarak, trafik işareti fotoğrafları üzerinde başarım sınaması sunulmaktadır. [9]'da ise yazarlar, gerçek dünya fotoğraflarında yer alan trafik işaretlerinden oluşan bir veri seti oluşturmak üzere çaba göstermektedir. Böylece, çalışmada kullanılan evrişimsel sinir ağları tabanlı modele ilişkin tahmin başarımının artması hedeflenmektedir.

Otonom araçlar için önerilen ve temelde şerit takibiyle birlikte nesne tespit ve tanıma yaklaşımları içeren modeller gerçek dünyada uygulanabilir olmalıdır. Yukarıda belirtilen literatür çalışmaları, önerdikleri şerit takip yaklaşımları ve/veya nesne tespit ve tanıma sistemlerinin başarımını, bir veri seti üzerinden eğitilen öğrenme modeli ile teorik olarak sınamaktadır. Oysaki, otonom araçlarda, görüntünün elde edilmesi, veriye dönüştürülmesi ve işlenerek sonuç üretilmesi, gerçek zaman kısıtları altında ele alınması gereken bir süreçtir. [10]'da RetinaNet algoritması; [11]'de ise YOLO algoritması ile gerçek zamanlı trafik işareti tanıma modelleri önerilmektedir. Performans değerlendirmesi için GTSRB veri seti kullanılmaktadır. Ayrıca, bu çalışmalarda önerilen yöntemler Faster R-CNN yöntemiyle karşılaştırılmaktadır. [12]'de ise yazarlar, trafik işareti tespit ve tanıma işlemi için gömülü bir mimari ile NVIDIA Jetson kartları kullanmayı önermektedir. Böylece, gerçek zamanlı çalışma konusunda gereken hız elde edilmektedir. Ancak, böyle bir modelde yazarlar, az enerji tüketimi ile fazla işlem yeteneğine sahip donanımlara ihtiyaç duymaktadır. YOLO algoritması ile istenen başarım elde edilse de, yöntemin fazladan enerji tüketimine ihtiyaç duyması nedeniyle, önerilen model uygulanabilirlik açısından sorgulanabilir. Bir başka çalışma olan [13]'te yazarlar, gerçek zamanlı bir otonom araç sisteminde kullanılacak görüntü işleme yaklaşımları için, işlem yeteneği yüksek donanımlara ihtiyaç duyulduğunu belirtmektedir. Böylece, bir otonom araç için gerçek zamanlı kararlar alınması adına, çevresel bileşenler yeterince hızlı bir şekilde işlenebilir. Çalışmada ayrıca, yoldaki şeritlerin gerçek zamanlı bir şekilde işlenmesi adına bir algoritma sunulmaktadır.



Şekil 1. Örnek bir otonom araç topolojisi (An example autonomous vehicle topology)

Literatürde yer alan gerçek zamanlı trafik işareti tanıma çalışmalarının birçoğu fiziksel donanımlar üzerinde çalışmaktadır. Öte yandan, gerçek dünyada bir otonom araç topolojisi oluşturmak, içerdiği bileşenler açısından maliyetli bir yöntemdir. Bu sebeple, gerçek zamanlı bir başarım analizi için sanal bir ortam oluşturmak üzere benzetim tekniğinden faydalanılabilir. Benzetim araçları farklı mühendislik disiplinlerinde yaygın olarak kullanılmaktadır. Örneğin, [14]'te yazarlar, bir yapının verimli bir mimari tasarıma sahip olması için bina benzetimi ile yapı kullanım analizi ortaya koymaktadır. [15]'te ise yazarlar, binalar için gerçekleştirilen dıştan ısı yalıtım uygulamasının ısıl ve nesnel performansını bir benzetim üzerinden değerlendirmektedir. Bu iki çalışmada sunulan benzetim ortamlarının hazırlanması için, sırasıyla Unity ve WUFI 2D gibi kütüphanelerden faydalanılmaktadır. [16-18] çalışmalarında yazarlar, C# programlama dili, Unity kütüphanesi, Unreal Engine kütüphanesi gibi alt yapıları kullanarak benzetim ortamlarının eğitim amaçlı hazırlanmasına değinmektedir. Bu çalışmalarda, farklı endüstriler için benzetim araçlarıyla oryantasyon süreci deneyimlemenin önemi vurgulanmaktadır. Örneğin, kimyasal bileşenlerin bulunduğu bir laboratuvar ortamı için benzetim araçlarıyla oryantasyon eğitimi verilerek, benzetim ortamında öğrenilen deneyimler ışığında, can ve mal kaybı ile sonuçlanabilecek olumsuz bir saha deneyiminin önüne geçilebilir. Benzer şekilde, [19]'da yazarlar, mühendislik süreçlerinde üretim planlama için benzetim araçlarının kullanılmasını önermekte; üretim planlama sırasında, karar noktalarında alınan kararlar sonucu oluşabilecek farklı senaryolar için maliyetlerin hesaplanması ve en düşük maliyete sahip stratejinin belirlenmesini hedeflemektedir. [20]'de ise yazarlar, akıllı şehirlerde kavşak yönetimi için önerdikleri yazılım tanımlı ağ mimarisini SUMO ve NS2 benzetim platformları ile analiz etmektedir. Otonom araç tasarımına ilişkin benzetim ortamlarını ele alan çalışmalardan [21]'de yazarlar, mevcut benzetim ortamlarının farklılıklarını, benzerliklerini ve sahip oldukları yetenekleri göz önüne alarak analizler ortaya koymaktadır. Benzer şekilde, [22]'de yazarlar, otonom araçların test süreçleri için bir benzetim ortamı oluşturmak üzere makine öğrenmesi yöntemlerini çalıştırabilen bir benzetim modeli ele almaktadır. [23]'te ise, trafik sakinleştirme için farklı kıvrımlama uygulamalarının performansını inceleyen yazarlar, önerilen senaryonun sınanması için yeni bir benzetim ortamı hazırlamaktadır. Ancak, önerilen bu modellerin saha çalışmasına özgü olduğu ve farklı tasarımlar için tekrar kullanılabilir olmadığı görülmektedir. Tüm bu çalışmalar incelendiğinde, modüler ve yeniden uygulanabilir benzetim mimarilerinin gerekliliği ortaya çıkmaktadır.

Belirtilen motivasyonlar göz önüne alındığında, bu çalışmada önerilen fayda aşağıdaki gibi özetlenebilir:

- Otonom bir sürüş modeli üzerinde test süreçlerinin yürütülmesi ve değerlendirilmesi için gerçek zamanlı modüler bir benzetim mimarisi önerilmektedir.
- Önerilen benzetim ortamında, yoldan elde edilen görüntüleri işleyerek şerit takibi yapan ve makine öğrenmesi ile trafik işaretlerini sınıflandırarak yönlenebilen bir otonom sürüş modeli tanıtılmaktadır.
- Trafik işaretlerini sınıflandırmak için, önerilen benzetim yazılımı ve otonom sürüş modeli ile, nesne tespit ve tanıma algoritmalarından YOLO ve R-CNN algoritmalarının farklı versiyonları kullanılarak başarım analizi ve performans karşılaştırması sunulmaktadır.

Çalışmanın geri kalanında; ikinci bölümde, önerilen benzetim ortamı mimarisi ve otonom araç modelinin detayları tanıtılmaktadır. Üçüncü bölümde, performans değerlendirmesi için kullanılan bileşenler ve değerlendirme sonuçları yer almaktadır. Dördüncü bölümde, çalışmadan elde edilen bulgular tartışılmaktadır. Beşinci bölümde ise çalışma, gelecek yönleri belirtilerek, sonuçlandırılmaktadır.

2. Önerilen Benzetim Mimarisi
(The Proposed Simulation Architecture)

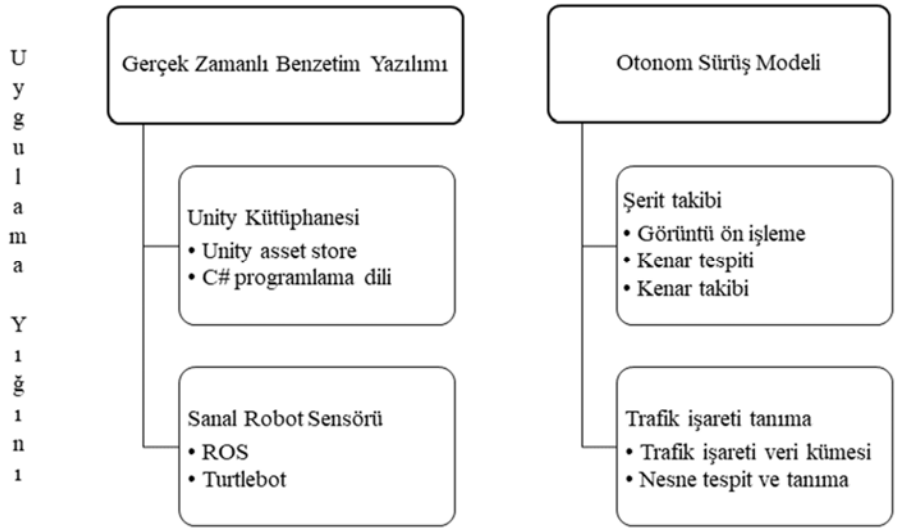
Otonom araç tasarımı için benzetim yazılımları literatürde bolca çalışılan bir konudur. [24]'te yazarlar, otonom araç tasarımı için kullanılan benzetim yazılımlarının genellikle kavramsal modellerin matematiksel doğrulaması için geliştirildiğine değinmektedir. Bu durumda, gerçek dünya bileşenlerinin birçoğu bazı varsayımlar ile benzetim ortamına dahil edilirken göreve özel benzetim ortamları oluşturulmakta ve sistemin modülerliği göz ardı edilmektedir. [25]'te ise yazarlar, güvenli bir otonom sürüş benzetim ortamının modüler bir tasarıma sahip olması gerektiğini vurgulamaktadır. Çalışmada belirtildiği üzere, otonom sürüş modelleri yaya güvenliğinden sürücü güvenliğine, trafik işareti tanımadan engel tespitine kadar birçok bileşen içermektedir. Bu nedenle, her bir bileşenin güvenilir bir şekilde doğrulanması için gereken benzetim yazılımları modüler tasarımı desteklemelidir. [26]'da yazarlar, benzetim yazılımları ile kavramsal doğrulamanın yanı sıra görselleştirme öğeleri kullanılarak gerçek dünya unsurlarının daha somut şekilde benzetim ortamına aktarılması gerektiğini savunmaktadır. Bu sebeple, üç boyutlu görsel nesneler içeren oyun motorlarının kullanılması, otonom araç tasarımı için ele alınan benzetim sistemlerinde tercih edilen yaklaşımlardan biri olarak karşımıza çıkmaktadır.

Yukarıda belirtilen motivasyonlar göz önüne alındığında, otonom sürüş modeli için tasarlanan bir benzetim mimarisinin modüler bileşenlerden oluşan, genişletilebilir, farklı senaryolara uyarlanabilir ve görselleştirilebilir olması önem arz etmektedir. Bu amaçla, bu çalışmada tasarlanan uygulama yığını temel bileşenler ve bu temel bileşenleri oluşturan alt modüllerden oluşmaktadır. Önerilen benzetim mimarisinde yer alan bileşenler Şekil 2 ile verilen uygulama yığını ile gösterilmektedir. Buna göre, uygulama yığınının en altında gerçek zamanlı benzetim yazılımı yer almaktadır. Bu yazılım üzerinde ise, otonom araç benzetimi için gerekli bileşenleri içeren otonom sürüş modeli bulunmaktadır. Bileşenlerin işlevleri ve içerdiği alt bileşenlerin tercih edilme sebebi bu bölümde detaylandırılmaktadır.

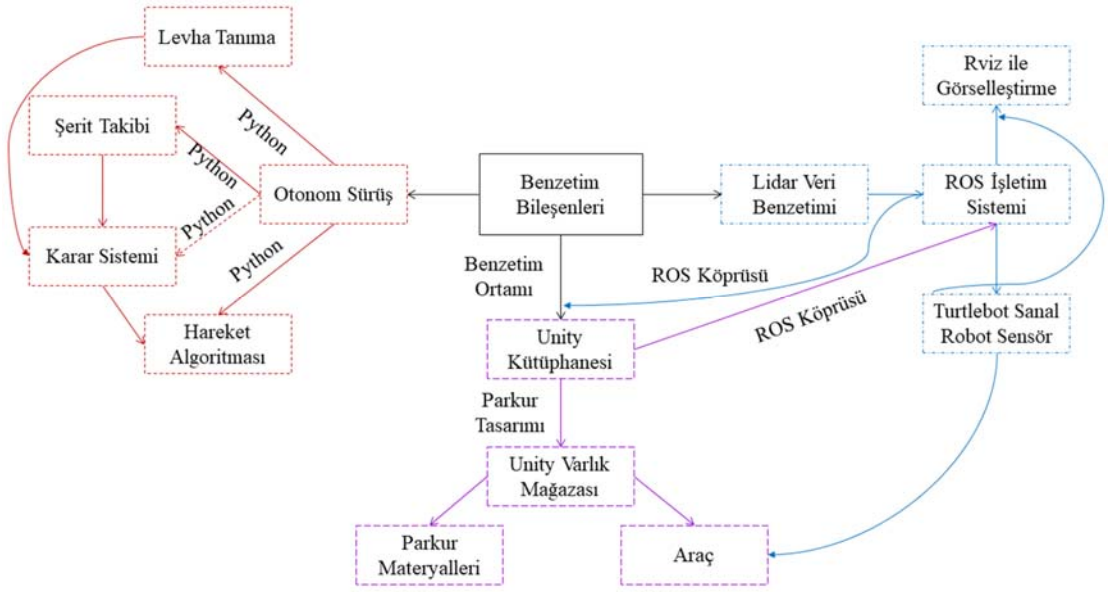
2.1. Gerçek Zamanlı Benzetim Yazılımı (Real-time Simulation Software)

Çalışmada önerilen benzetim mimarisinin temel bileşeni gerçek zamanlı benzetim yazılımıdır. Benzetim yazılımı, mimariyi oluşturan uygulama yığının en altında yer almaktadır. Böylece, farklı senaryolara özgü yeni otonom sürüş yaklaşımları bu benzetim yazılımı üzerinde sınanarak, gerçek dünya donanım maliyetleri olmaksızın bir ön değerlendirme ortamı elde edilmektedir. Geliştirilen benzetim yazılımı için gereken bileşenler Şekil 3 ile gösterilmektedir. Buna göre, çalışmada önerilen benzetim yazılımının gerçeklenmesi için üç temel bileşen ve bu bileşenlerin birbirleriyle ilişkilerine ihtiyaç duyulmaktadır.

Gerçek zamanlı benzetim yazılımını oluşturan temel alt bileşen Unity kütüphanesi kullanılarak geliştirilen benzetim ortamıdır. Çalışmada tercih edilen Unity kütüphanesi alternatifleri olan Unreal Engine, GameMaker, AppGameKit gibi oyun motorlarına göre birçok avantaja sahiptir. Bu avantajlardan bazıları, çoklu platform desteği, iki ve üç boyutlu hazır nesnelerin kullanımına olanak sağlaması, çizim araçları ile hazırlanan nesnelerin kolay entegrasyonu olarak sıralanabilir [27]. Unity kütüphanesi ile görsel varlıkların oluşturulması ve kontrol akışının sağlanması adına iki temel alt süreç ele alınmaktadır. İki boyutlu görsel varlıklar Unity Asset Store mağazası üzerinden temin edilebileceği gibi; bir tasarım aracı olan Blender yazılımıyla özel olarak da tasarlanabilmektedir. Böylece, benzetim ortamının görselleştirilmesi için gereken çeşitli parkur materyalleri ve araç nesnesi oluşturulmaktadır. Benzetim ortamının kontrol akışı ise C# programlama dili ile geliştirilen kodlar aracılığıyla sağlanmaktadır. Kontrol akışında, benzetim yazılımının



Şekil 2. Önerilen benzetim mimarisi için uygulama yığını (Application stack for the proposed simulation architecture)



Şekil 3. Benzetim yazılımına ilişkin benzetim bileşenleri (Simulation components for the emulator software)

diğer bileşenlerinden gelen bilgiler işlenerek otonom araç benzetiminin karar mekanizması kontrol edilmektedir. Örneğin, LİDAR bileşeninden gelen engel bilgisinin algılanması ve otonom sürüş sırasında değerlendirilerek karar mekanizmasının bir karar üretmesi süreci, C# programlama dilinde geliştirilen kontrol akışı ile sağlanmaktadır.

Gerçek zamanlı benzetim yazılımını oluşturan bir diğer alt bileşen ise, sanal robot sensörlerini yöneterek verilerinin benzetim ortamına aktarılmasından sorumludur. Çalışmada değerlendirilen senaryoda, bir Lidar sensörü, çevredeki engelleri algılayarak otonom sürüş sırasında oluşabilecek bir çarpışmayı önlemek için kullanılmaktadır. Bu bağlamda, Robotic Operation System (ROS) işletim sistemi ile birlikte Turtlebot ve RViz kütüphaneleri kullanılarak, bir Lidar bileşeni benzetim ortamına eklenmektedir. Turtlebot alt bileşeni, bir Lidar sensörünün benzetim ortamında sanal olarak oluşturulması için kullanılmaktadır. ROS alt bileşeni, robot işletim sistemi olarak kullanılır ve Turtlebot ile benzetimi yapılan Lidar sensörünün benzetim ortamına entegrasyonundan sorumludur. RViz alt bileşeni ise, Lidar sensörü verilerinin görselleştirilmesi ile ilgili alt bileşen olarak kullanılmaktadır. ROS alt bileşeni, alternatifleri olan YARP, MOOS, POCOLIBS, OROCOS, ROCK gibi diğer robot kontrol yazılımlarına göre daha genel amaçlı, açık kaynak kodlu, dilden bağımsız geliştirmeye olanak sağlayan ve birçok robot bileşeni ile etkileşen modüler bir platform olduğu için tercih edilmiştir. Ayrıca, Turtlebot ve Rviz kütüphaneleri ile sensör kontrolü ve görselleştirme üzerine uygun bir ekosistem oluşturmaktadır.

Yukarıda tanıtılan benzetim ortamı bileşenlerinin geliştirilmesiyle elde edilen benzetim yazılımına ait örnek bir ekran görüntüsü Şekil 4 ile verilmektedir. Şekil 4a, benzetim ortamına ilişkin oluşturulan parkurun kuş bakışı görüntüsünü içerirken; Şekil 4b'de parkurun üç boyutlu görüntüsü yer almaktadır. Benzetim yazılımı koşmaya başladıktan sonra, otonom sürüş bileşeni tarafından kullanılan bir

kamera ile şerit takibi ve trafik işareti tanıyarak yönlenme operasyonları gerçek zamanlı olarak benzetim ortamına yansıtılmaktadır. Ayrıca, Lidar bileşeninden alınan engel verileri dikkate alınarak güvenli bir sürüş ortamı oluşturul

2.2. Otonom Sürüş Modeli (Autonomous Driving Model)

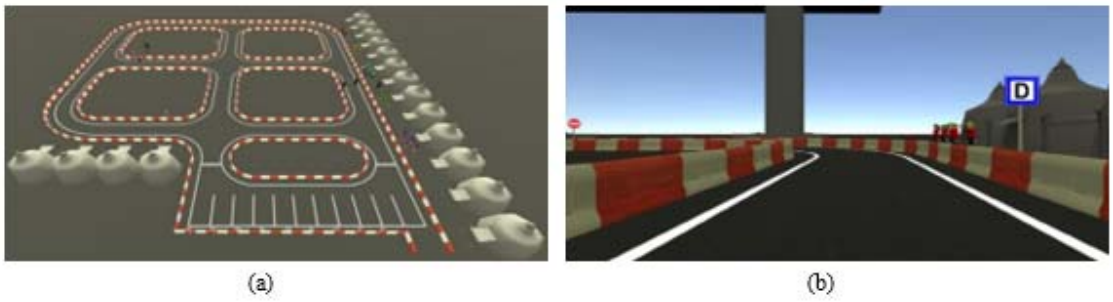
Benzetim ortamını oluşturan ana bileşenlerden bir diğeri otonom sürüş modelidir. Bu model, benzetim ortamında çalışmakta olan otonom aracın hareket kontrolünü sağlamaktadır. Çalışmada önerilen otonom sürüş modeline ilişkin yazılım, Python dili kullanılarak geliştirilmektedir. Model temel olarak trafik işareti tanıma, şerit takibi, karar akışı ve hareket algoritması alt bileşenlerini içermektedir. Böylece, benzetim ortamı içerisinde elde edilen parkur verileri işlenerek, benzetim yazılımında yer alan aracın, önerilen otonom sürüş modeline göre hareket etmesi sağlanmaktadır. Önerilen benzetim yazılımı sayesinde, farklı otonom sürüş modelleri bu bileşenin yerini alarak benzetim ortamında test edilebilir. Böylece, gerçek bir donanım ortamında çalışmanın getirmiş olduğu zaman ve bütçe maliyetleri azaltılır.

Çalışmada kullanılan otonom sürüş modeli, şerit takibi yapan ve trafik işaretlerini tanıyarak yönlenebilen bir otonom sürüş modelidir. Modelde, şerit takibi ve trafik işareti tanıma için görüntü işleme; engel algılama içinse, Lidar sensörleri kullanılmaktadır. Benzetim ortamında çalıştırılan otonom sürüş modeline ilişkin akış diyagramı Şekil 5 ile verilmektedir.

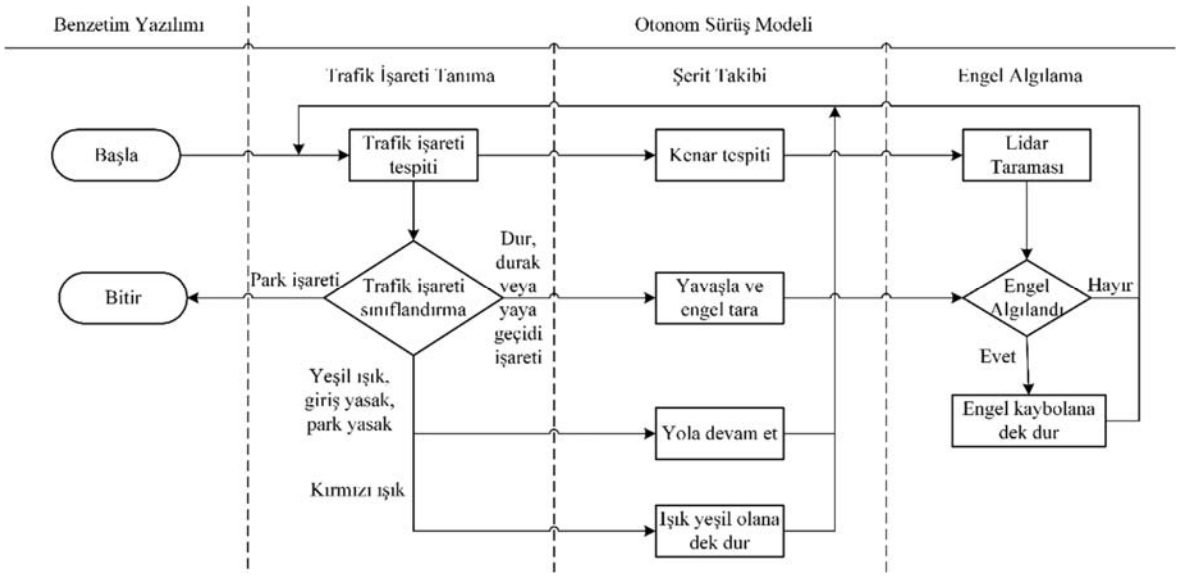
Şekil 5'e göre, benzetim ortamında paralel olarak çalışan sistemler yer almaktadır ve bu sistemler şekilde kesikli çizgilerle birbirinden ayrılmaktadır. Akış diyagramına göre, otonom araç benzetimi ile bir parkurda sürüş denemesi yapmak üzere benzetim yazılımı başlatılır. Böylece, parkurdan elde edilen trafik işaretlerinin tespiti yapılırken, diğer yandan şerit takibi ile ilgili akış devam eder. Ayrıca, otonom sürüş esnasında oluşabilecek bir çarpışmanın önüne geçmek için Lidar sensörlerinin engel algılama özelliğinden faydalanılmaktadır. Şekilde yer alan akışa göre, tespit edilen trafik işareti *park* ise benzetim sonlanmaktadır. *Dur*, *durak*, veya *yaya geçici* gibi trafik işaretleriyle karşılaşıldığında, Lidar taramasına başvurularak bir engel olup olmadığı araştırılır. Bir engel algılanması durumunda, otonom araç benzetimi engel ortadan kalkana kadar durmakta ve sonra tekrar hareket etmektedir. *Yeşil ışık*, *park yasak*, *giriş yasak* gibi trafik işaretleriyle karşılaşılması durumunda ise benzetim süreci şerit takip sistemine göre devam etmektedir. *Kırmızı* ışık gibi bir trafik işareti ile karşılaşıldığında ise, otonom araç benzetimi yeşil ışık yanana kadar durmaktadır. Çalışmada önerilen benzetim yazılımı için kullanılan trafik işaretlerine ilişkin detaylar bölüm 3.1'de yer almaktadır. Önerilen benzetim mimarisi üzerinde farklı trafik işaretlerini içeren otonom sürüş modelleri oluşturmak mümkündür.

2.2.1. Şerit takibi (Lane tracking)

Modelde, şerit takibi için görüntü işleme yaklaşımlarıyla yol görüntüsü üzerinde bazı ön işlemler uygulanarak kenar tespiti yapılır ve araç şeritlere göre konumlandırılır [28]. Bu bağlamda, elde edilen



Şekil 4. Benzetim ortamının kuş bakışı ve 3 boyutlu görüntüsü (Bird's eye view and 3D view of the simulation environment)



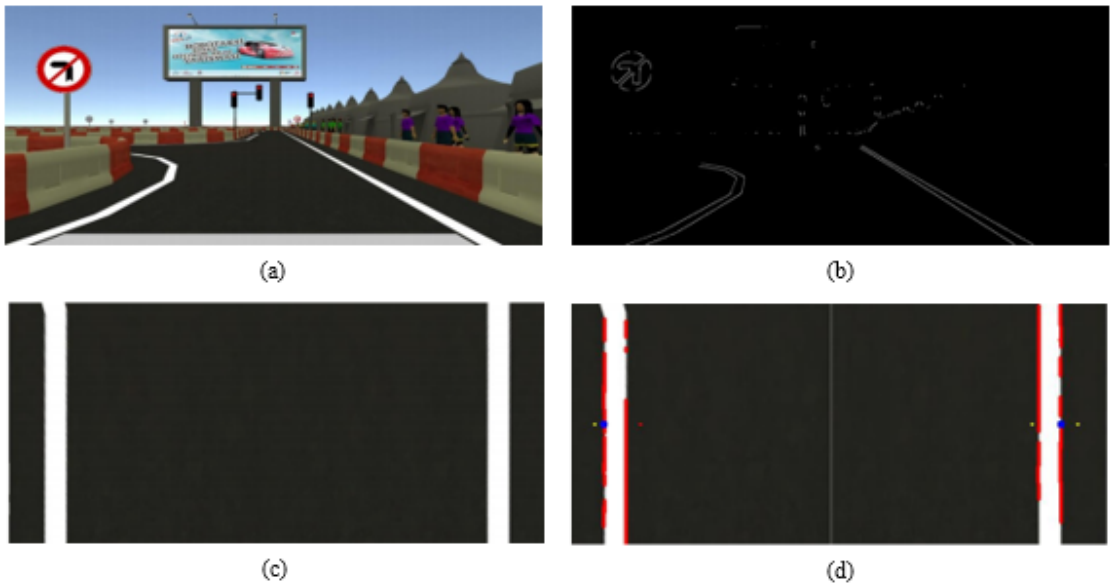
Şekil 5. Benzetim yazılımında çalışan otonom sürüş modelinin akış diyagramı
(Flowchart of autonomous driving model running in simulation software)

yol görüntüsü üzerinde gri tonlama, Canny kenar tespit algoritması ve Gaussian filtresi uygulanarak gürültüsüz ve yüksek doğrulukta bir şerit takip mekanizması elde edilmektedir. Otonom sürüş modelindeki şerit takibi için ele alınan ilk bileşen kenar tespitidir. Bu bağlamda, John F. Canny tarafından önerilen Canny kenar belirleme algoritması kullanılmaktadır [29]. Canny yönteminin uygulanması için, ilk olarak, kameradan gelen anlık veri, gri tonlamalı resim haline getirilmektedir. Ardından, benzetim ortamından elde edilen görüntü perspektif dönüşüm işlemi ile işlenir. Sonrasında, Canny kenar tespit algoritması, her pikselin yoğunluk gradyanını ölçer ve büyük renk değişimlerine sahip pikseller üzerine noktalar çizilir. Son olarak, Gaussian filtresi ile görüntü üzerindeki gürültü azaltılır. Bu adım, belirgin gürültünün kenar bulucu üzerindeki etkisini azaltmak için görüntüyü hafifçe yumuşatmaktadır. Bir başka deyişle, gereksiz ayrıntıların kenar gibi görünmemesi için kullanılmaktadır. Benzetim sırasında elde edilen bir görüntü için şerit takibinde kullanılan kenar tespit yöntemi Şekil 6'da örneklendirilmektedir. Şekil 6a orijinal görüntüyü gösterirken; orijinal görüntü üzerine ön işlemlerin uygulanması sonucu Şekil 6b ile verilen görüntü elde edilmektedir. Sonraki adımda, perspektif dönüşümü ile Şekil 6c'de yer alan görüntü oluşmaktadır. Son olarak, Canny kenar tespit algoritmasının ve Gaussian filtresinin uygulanmasıyla birlikte Şekil 6d'de verilen ve şeritlerin net bir şekilde tespit edildiği görüntü elde edilmektedir. Böylece, önerilen otonom sürüş modelinde yer alan şerit takibi için görüntü işleme süreci başarıyla uygulanmaktadır ve otonom araç benzetimi şeritleri ortalayacak şekilde yola konumlandırılmaktadır.

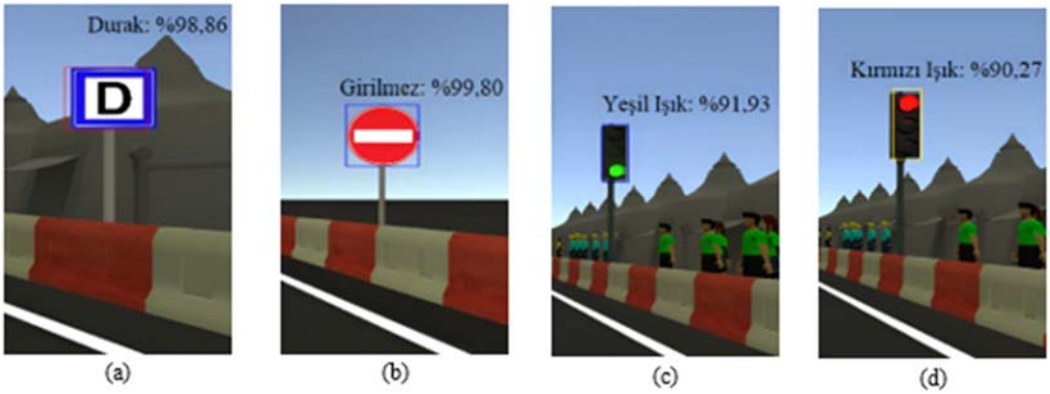
2.2.2. Trafik işareti tanıma (Traffic sign recognition)

Önerilen benzetim yazılımı için otonom sürüş modelinde trafik işareti tanıma özelliği OpenCV kütüphanesi ile geliştirilmektedir. Trafik işaretlerinin tespiti ve tanınması için evrişimsel sinir ağları tabanlı YOLO ve R-CNN algoritmalarının farklı versiyonları kullanılmaktadır. Böylece, trafik işaretleri sınıflandırılarak uygun sürüş kararları elde edilmektedir. Trafik işaretlerinin, geliştirilen benzetim yazılımında, tespiti ve tanınmasına ilişkin örnek bir görüntü elde etme ve işleme süreci Şekil 7'de gösterilmektedir.

Trafik işareti tespiti ve tanınması için, ilk olarak, benzetim ortamından elde edilen trafik işareti görüntüsü YOLO ve R-CNN algoritmalarının nesne tespit yeteneği ile tespit edilir. Görüntünün tespiti sonrasında, eğitilen model dikkate alınarak görüntü sınıflandırma süreci devreye



Şekil 6. Önerilen otonom sürüş modelinde şerit takibi için kenar tespitinin yapılması
(Edge detection for lane tracking in the proposed autonomous driving model)



Şekil 7. Önerilen benzetim ortamında YOLO ve R-CNN algoritmalarının farklı versiyonları ile nesne tespiti ve nesne tanıma
(Object detection and object recognition with different versions of YOLO and R-CNN algorithms in the proposed simulation environment)

girer ve otonom sürüş modeli dikkate alınarak benzetim yazılımının kontrol akışı düzenlenir. Örneğin, algılanan görüntü trafik ışığı ise ve kırmızı yanıyor ise araç durur; aksi takdirde şerit takip sistemine göre harekete devam eder. Böylece, önerilen benzetim mimarisinde koşan bir otonom araç benzetimi, trafik işaretlerini tanıyarak başarıyla yönlenir.

Önerilen benzetim ortamı mimarisi ve otonom sürüş modeli, tanıtılan ayrıntıları göz önüne alınarak YOLO ve R-CNN algoritmalarının farklı versiyonları ile test edilmektedir. Benzetim ortamından elde edilen trafik işareti görüntülerinin sınıflandırılması için farklı makine öğrenmesi yöntemleri de kullanılabilir. Bu çalışmada, nesne tespit ve tanımada yüksek başarım gösterdiği bilinen YOLO-v3, YOLO-v3 tiny, YOLO-v4, YOLO-v4 tiny, Fast R-CNN, Faster R-CNN ve Mask R-CNN modelleri kullanılarak önerilen benzetim mimarisinin kullanılabilirliği sınanmaktadır.

3. Performans Değerlendirmesi (Performance Evaluation)

Bu bölümde, önerilen gerçek zamanlı benzetim mimarisi için hazırlanan değerlendirme ortamı tanıtılmaktadır. Dahası, benzetim yazılımı üzerinde test edilen evrişimsel sinir ağları tabanlı YOLO ve R-CNN algoritmaları versiyonlarının trafik işareti tanımadaki başarımı ve önerilen benzetim mimarisinin kullanılabilirliğine ilişkin değerlendirme bulguları detaylandırılmaktadır.

3.1. Değerlendirme Ortamı (Evaluation Environment)

Çalışmada hazırlanan değerlendirme ortamında iki temel fiziksel bileşen yer almaktadır. Bunlardan ilki, Unity kütüphanesi ile hazırlanan benzetim yazılımının koştuğu Windows işletim sistemine sahip bir makinedir. Diğeri ise, veri setinin YOLO ve R-CNN algoritmalarının farklı versiyonları ile eğitildiği bir Google Colab sunucusudur. Windows makine üzerinde, benzetim ortamında bulunan bileşenleri oluşturmak adına gereken kurulumlar şöyledir:

- Unity Hub geliştirme ortamı ve Unity kütüphanesi ile benzetim ortamının çatısı oluşturulmaktadır.
- Ubuntu 16.04 sürümüne sahip bir sanal makine kurularak, benzetimde Lidar sensörünü temsil etmesi için Turtlebot sanal robot sensör bileşeni hazırlanmaktadır.
- ROS işletim sistemine sahip bir sanal sunucu kurularak, Unity ve Turtlebot arasında bağlantı oluşturulup; sanal robotun, Unity programına veri aktarması sağlanmaktadır. Ayrıca, ROS işletim sistemi üzerinde koşan RViz yazılımı üzerinden görselleştirme yapılmaktadır.

Değerlendirme ortamı için model eğitiminde kullanılan Google Colab sunucusuna ilişkin teknik özellikler ise aşağıdaki gibidir:

- Intel(R) Xeon(R) CPU @ 2.30GHz işlemci
- 46 bit fiziksel, 48 bit sanal adresleme
- 46080 KB ön bellek
- 12GB NVIDIA Tesla K80 GPU
- 13 GB RAM

Çalışmada kullanılan trafik işaretleri veri seti için bölgesel bir veri seti seçimi tercih edilmektedir. Farklı ülkelerde kullanılan bazı trafik işaretleri, farklı geometrik şekillere sahip materyaller üzerine basıldığı için, kullanılan makine öğrenmesi yönteminin başarımını etkilemektedir. Bu durum, Almanya'da kullanılan trafik işaretlerinden oluşan bir veri setini ve Çin'de kullanılan trafik işaretlerinden oluşan bir veri setini aynı makine öğrenmesi yöntemiyle karşılaştıran fakat farklı başarımlar elde eden çalışmalarla desteklenmektedir [30], [31]. Bu bağlamda, bu çalışmada, Türkiye'nin sokak görüntülerinden elde edilen ve farklı trafik işaretlerinin etiketlenmesiyle oluşturulan bir seti setinin kullanımı tercih edilmektedir [32]. Bu veri setinde yer alan trafik işaretleri, YOLO ve R-CNN algoritmalarının farklı versiyonları ile eğitilerek, bir test veri seti üzerinde sınanmaktadır. Böylece, benzetim ortamında, seçilen trafik işaretleri kullanılarak, önerilen gerçek zamanlı benzetim mimarisinin başarımı test edilmektedir. Buna göre, benzetim yazılımı için seçilen 8 farklı trafik işareti sınıfı ve bu işaretler ile karşılaşıldığında benzetim ortamının aldığı karar açıklaması Tablo 1 ile verilmektedir.

Özetle, değerlendirme ortamı için yukarıda belirtilen özelliklere sahip fiziksel ve sanal bileşenlerden faydalanılarak, 8 farklı trafik işareti içeren bir parkurda oluşturulan rotalar için, önerilen otonom sürüş modelinin YOLO ve R-CNN algoritmalarının farklı versiyonları ile başarım analizi ortaya konmaktadır.

3.2. YOLO ve R-CNN Algoritmalarının Farklı Versiyonları ile Önerilen Benzetim Ortamının Analizi
(Analysis of the Proposed Simulation Environment with Different Versions of YOLO and R-CNN Algorithms)

Çalışmada önerilen otonom sürüş modeli, kullanılan trafik işaretleri veri seti ile, YOLO-v3, YOLO-v3 tiny, YOLO-v4, YOLO-v4 tiny, Fast R-CNN, Faster R-CNN, ve Mask R-CNN olmak üzere 7 farklı nesne tespit ve tanıma algoritması ile ayrı ayrı eğitilmektedir. Farklı eğitim modelleri, tanıtılan otonom sürüş bileşenleri ile benzetim ortamında koşarak, elde edilen bulgular analiz edilmektedir. Böylece, her bir benzetim koşumu için, kullanılan nesne tanıma algoritmalarının eğitim sürecine ve nesne tanımadaki başarımına ilişkin analizler ortaya konmaktadır.

İlk olarak, çalışmada nesne tespit ve tanıma için kullanılan YOLO ve R-CNN algoritmalarının farklı versiyonları ile veri setinin eğitimine ilişkin analizler Tablo 2'de verilmektedir. Buna göre, çalışmada hazırlanan veri seti, her bir model için subdivision ve batch boyutu sırasıyla 32 ve 8 olan bir yapay sinir ağı üzerinde eğitilmektedir. Ayrıca, her bir model için Google Colab sunucusuna ortalama 20 mb/s internet erişim hızı ile bağlantı kurulmaktadır. Her bir modelin ortalama görüntü çerçevesi işleme sayısı ise ortalama FPS sütununda belirtilmektedir. Bu değerler, modelin görüntüyü işlerken ne kadar

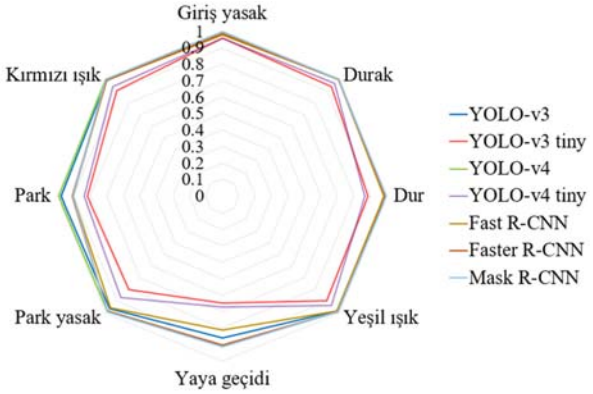
Tablo 1. Hazırlanan test veri setinde yer alan trafik işaretleri ve benzetim ortamındaki anlamları
(Traffic signs in the prepared test data set and their meanings in the simulation environment)

Trafik İşareti Sınıfı	Benzetim Ortamındaki Anlamı
Dur	Araç durur, çevresinde bir engel algılamaz ise yola devam eder.
Durak	Araç belli süre durur, yolcu alıp bırakarak yola devam eder.
Giriş yasak	Araç şerit çizgilerine göre alternatif bir yoldan harekete devam eder.
Kırmızı ışık	Araç yeşil ışık yanana kadar durur, sonrasında harekete devam eder.
Park	Araç durur ve benzetim süreci sonlanır.
Park yasak	Araç şerit çizgilerine göre harekete devam eder.
Yaya geçidi	Araç yavaşlar, engel algılanırsa durur, aksi takdirde yola devam eder.
Yeşil ışık	Araç harekete devam eder.

fazla detayı ele aldığını gösterirken eğitim süresi üzerinde de önemli rol oynamaktadır. Örneğin YOLO-v3 tiny algoritması saniyede 220 görüntü çerçevesi işlediği için, her bir görüntüye ilişkin daha yüzeysel bilgileri öğrenir. Bu bağlamda, modelin eğitim süresi de diğer modellere göre daha kısa sürmektedir. Diğer yandan, farklı R-CNN algoritmaları saniyede 5 ile 7 görüntü çerçevesini daha fazla bilgi içeriği elde edecek şekilde işleme eğilimdeyken, öğrenme modeli bu algoritmalarla daha uzun sürede eğitilmektedir. Son olarak, her bir model için, eğitim sürelerine ait bilgiler, eğitim süresi sütununda yer almaktadır.

Önerilen benzetim mimarisinin analizi için eğitilen YOLO ve R-CNN algoritmalarının farklı versiyonları, hazırlanan veri seti üzerinde, 10-kat çapraz doğrulama yöntemi kullanılarak analiz edilmektedir. Buna göre, her bir algoritma için, veri seti on eşit parçaya bölünerek, dokuz parça ile eğitilip; kalan bir parça ile test edilmektedir. Deneyler, veri setinin farklı parçaları için on kez tekrarlanmakta ve elde edilen sonuçlara ilişkin detaylar Tablo 3 ile verilmektedir. Tabloda, YOLO ve R-CNN algoritmalarının farklı versiyonları için, her bir trafik işaretinin tanınmasıyla ilişkili ortalama doğruluk oranları yer almaktadır. Ayrıca, her bir trafik işaretinin veri setindeki miktarı dikkate alınarak, her bir öğrenme modelinin genel doğruluk oranı da tablonun en alt satırında belirtilmektedir.

Tablo 3 ile özetlenen değerlendirme sonuçları Şekil 8'de görselleştirilmektedir. Buna göre, YOLO ve R-CNN algoritmalarının farklı versiyonlarının, sınama ortamında kullanılan sekiz farklı trafik işaretini tanımadaki başarım oranı sekizgen bir şekil ile ifade edilmektedir. Şekilde yer alan köşeler test ortamında kullanılan farklı trafik işaretlerine denk gelmektedir. Şeklin merkezi, trafik işaretlerini tanımada %0 doğruluk oranını ifade ederken; tam köşelere denk gelen noktalar ise %100 doğruluk oranı ile trafik işareti tespit ve tanımayı ifade etmektedir. Bu bağlamda, şeklin merkezinde 0 değeri yer alırken, merkezden köşelere doğru ilerledikçe 1 değerine yaklaşılmaktadır. Elde edilen bulgulara göre, yeşil renk ile gösterilen YOLO-v4 algoritması, her bir trafik işareti için şeklin köşelerine yakın değerler üretmektedir. Diğer bir deyişle, trafik işareti tespit ve tanımada diğer YOLO versiyonlarından daha başarılıdır. Başarım oranı yüksek bir diğer YOLO algoritma versiyonu, mavi renk ile gösterilen YOLO-v3 algoritması olarak görülmektedir. YOLO-v4 ve YOLO-v3 algoritma versiyonları ile yürütülen deneylerde, her bir trafik levhası için şeklin köşelerinde yer alan 1 değerine yakın değerler elde edilmektedir. Farklı YOLO algoritması versiyonlarının başarım sırasını, mor renk ile ifade edilen YOLO-v4 tiny takip etmektedir. Bu algoritmanın, özellikle *park* ve *yaya geçidi* trafik işaretlerini tanımada daha düşük başarım gösterdiği görülmektedir. Kırmızı renk ile gösterilen YOLO-v3 tiny versiyonu ise karşılaştırılan algoritma versiyonlarından en düşük başarıma sahip versiyon olarak tespit edilmektedir. Bu versiyonun *park*, *park yasak* ve *yaya geçidi* trafik işaretlerini tanımada düşük başarım gösterdiği tespit edilmektedir.



Şekil 8. Çalışmada kullanılan farklı algoritmaların farklı trafik işaretlerinin tespiti konusunda başarım kıyaslaması
(Performance comparison of different algorithms used in the study in detecting different traffic signs)

Farklı R-CNN algoritması versiyonları incelendiğinde ise, tüm trafik işaretleri için, en başarılı sonuçların Mask R-CNN versiyonu ile elde edildiği görülmektedir. Mask R-CNN modeli ile elden edilen doğruluk oranı, en yüksek başarımın elde edildiği YOLO-v4 algoritmasının bir miktar altında kalırken, diğer YOLO algoritması versiyonlarından daha yüksektir. R-CNN algoritması versiyonlarından Faster R-CNN algoritması, Fast R-CNN'e göre daha yüksek doğrulukta tahminlime yaparken; Mask R-CNN algoritması

Tablo 2. Otonom sürüş modelinde kullanılan farklı algoritmaların eğitim sürecine ilişkin veriler
(Data on the training process of different algorithms used in the autonomous driving model)

Model	Subdivision/batch boyutu	Ort. Internet hızı	Saniyede işlenen ortalama çerçeve sayısı	Eğitim süresi
YOLO-v3			45 fps	8 saat
YOLO-v3 tiny			220 fps	3 saat
YOLO-v4			21 fps	18 saat
YOLO-v4 tiny	32/8	20 mb/s	34 fps	13 saat
Fast R-CNN			5 fps	29 saat
Faster R-CNN			7 fps	26 saat
Mask R-CNN			6 fps	28 saat

Tablo 3. Otonom sürüş modelinde kullanılan farklı algoritmaların ortalama doğruluk oranı
(Average accuracy rate of different algorithms used in autonomous driving model)

Trafik İşareti Sınıfı	YOLO v3	YOLO v3 tiny	YOLO v4	YOLO v4 tiny	Fast R-CNN	Faster R-CNN	Mask R-CNN
Dur	0,988	0,884	0,993	0,865	0,982	0,992	0,994
Durak	0,998	0,938	0,999	0,965	0,998	0,999	0,999
Giriş yasak	0,988	0,957	0,995	0,958	0,976	0,988	0,995
Kırmızı ışık	0,993	0,902	0,997	0,940	0,991	0,993	0,997
Park	0,977	0,819	0,989	0,838	0,903	0,908	0,905
Park yasak	0,965	0,799	0,985	0,867	0,957	0,986	0,989
Yaya geçidi	0,860	0,650	0,907	0,672	0,812	0,905	0,911
Yeşil ışık	0,986	0,899	0,991	0,936	0,987	0,992	0,994
Genel Doğruluk Oranı	*0,92*	*0,78*	*0,95*	*0,81*	*0,87*	*0,91*	*0,93*

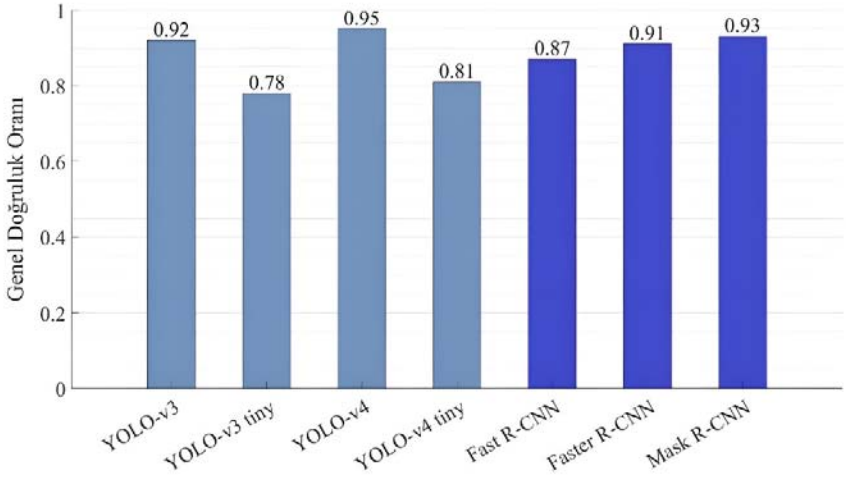
ile elde edilen doğruluk oranlarının bir miktar gerisinde kalmaktadır. Faster R-CNN algoritmasını YOLO-v3 algoritması ile karşılaştırmak gerekirse, *park* işareti hariç diğer trafik işaretleri için daha yüksek başarım ile nesne tanımasına rağmen, *park* işaretindeki düşük başarımı nedeniyle genel doğruluk oranında YOLO-v3 algoritmasının hemen gerisinde kalmaktadır. Diğer yandan, Fast R-CNN algoritması *dur*, *durak* ve *yeşil ışık* işaretlerini YOLO-v3 algoritmasına benzer bir doğruluk oranı ile tespit etse de, diğer trafik işaretlerindeki doğruluk oranının daha düşük olması nedeniyle genel doğruluk olanında YOLO-v3 algoritmasının bir hayli gerisinde kalmaktadır. Son olarak, tüm R-CNN versiyonlarının, YOLO-v3 tiny ve YOLO-v4 tiny algoritmalarına göre, tüm trafik işaretlerinin tanınması için çok daha yüksek doğruluk ile çalıştığı savunulabilir.

Çalışmada ele alınan evrişimsel sinir ağları tabanlı YOLO ve R-CNN algoritmalarının farklı versiyonları için tüm veri seti miktarı dikkate alındığında elde edilen genel doğruluk oranları Şekil 9 ile gösterilmektedir. Tablo 3'ün en alt satırında yer alan genel doğruluk oranı verilerine göre elde edilen şekilde, farklı YOLO algoritması versiyonları ve farklı R-CNN algoritması versiyonları iki farklı renk ile gruplanarak kendi içlerindeki kıyaslama görselleştirilmektedir. Buna göre, YOLO-v4 en başarılı YOLO versiyonu iken; Mask R-CNN en başarılı R-CNN versiyonu olarak tespit edilmektedir. Ayrıca, şekilde yer alan tüm algoritmalar içerisinde, YOLO-v4 algoritmasının %95 ile en yüksek, YOLO-v3 tiny algoritmasının ise %78 ile en düşük doğruluk oranı sonuçlarını ürettiği görülmektedir.

4. Sonuçlar ve Tartışmalar (Results and Discussions)

Çalışmadan elde edilen bulgulara göre, YOLO-v4 algoritması, trafik işaretlerini tanımada diğer algoritmalar arasında en yüksek genel doğruluk oranı (%95) ile sonuçlar üretmektedir. Ayrıca, veri setinde yer alan tüm trafik işaretlerinin tekil olarak tanınmasında da en yüksek doğruluk oranına sahiptir. YOLO-v4 algoritmasının başarımını, %93 genel doğruluk oranı ile Mask R-CNN algoritması takip etmektedir. Mask R-CNN algoritması doğruluk oranı olarak YOLO-v4 algoritmasına yakın değerler üretse de modelin eğitimi YOLO tabanlı algoritma versiyonlarına göre çok daha uzun sürmektedir. Genel doğruluk oranı başarımında takip eden algoritma, %92 doğruluk oranı ile YOLO-v3 ve %91 doğruluk oranı ile Faster R-CNN'dir. Her iki algoritma arasında YOLO-v4 ve Mask R-CNN arasındaki ilişkiye benzer bir ilişki söz konusudur. YOLO-v3 ve Faster R-CNN için trafik işareti tanımada genel doğruluk oranı ve tekil doğruluk oranları yakın olsa da, bu modellerin eğitimi için gereken süreler oldukça farklıdır. Başarım sırasını takip eden algoritma %87 genel doğruluk oranı ile Fast R-CNN olarak tespit edilmiştir. Ayrıca, Fast R-CNN, deneyler sırasında model eğitimi en uzun olan algoritma olarak karşımıza çıkmaktadır. Çalışmadan elde edilen analizlerde tiny olarak adlandırılan ve hafifletilmiş bir eğitim süreci içeren YOLO-v4 tiny ve YOLO-v3 tiny sırasıyla %81 ve %78 genel doğruluk oranlarıyla en son sırada yer alan iki algoritma olarak belirlenmiştir. Bu algoritmalar ile modelin eğitim süresi normal YOLO versiyonlarına göre ciddi oranda düşürülürken, doğruluk oranında kayıplar yaşandığı savunulabilir.

Çalışmada yürütülen deneylerde, YOLO algoritmasının farklı versiyonları arasında daha güncel olanların daha başarılı doğruluk oranı ile nesne tanıma yaptığı; hafifletilmiş algoritmalarınsa daha kısa sürede çalıştığı gözlemlenmektedir. Bu bağlamda, çalışmada gözlemlenen değişim, YOLO versiyonlarında kullanılan evrişimsel sinir ağları mimarisinin yeteneklerine paralel hareket etmektedir. Doğruluk oranı üzerinde benzer bir ilişki farklı R-CNN algoritması versiyonları arasında da mevcuttur. Diğer bir deyişle, daha eski bir sürüm olan Fast R-CNN algoritması diğerlerine göre daha düşük doğruluk oranı üretmektedir. Ayrıca, R-CNN algoritması versiyonlarında kullanılan yapay sinir ağı mimarisi gereği, eğitim sırasında daha fazla süreye ihtiyaç duyulduğu not edilmiştir. Son olarak, YOLO algoritması versiyonları ve R-CNN algoritması versiyonlarının birbirlerine göre üstünlükleri genel doğruluk oranı ve trafik işaretlerinin tekil olarak tanınmasındaki doğruluk oranı şeklinde ortaya konmaktadır. Çalışmadan elde edilen bulguları, literatürde yer alan ve evrişimsel sinir ağları tabanlı yöntemler ile trafik işareti tespit ve tanıma modelleri öneren çalışmalarla kıyasladığımızda, benzer doğruluk oranları üretildiği görülmektedir. Mevcut çalışmalarda Almanya, Belçika, Çin ve İngiltere gibi bölgesel trafik kurallarına ve buna bağlı trafik işaretlerine sahip veri setleri üzerinde, tekil trafik işaretlerinin birçoğu için %95 ile %99 arasında başarıma sahip doğruluk oranları elde edildiği tespit edilmiştir [33-36]. Buna göre, çalışmada ele alınan yöntemler ile elde edilen bulgular arasında uygun bir korelasyon olduğu ortaya konmaktadır. Sonuç olarak, önerilen benzetim mimarisi ile kullanılan veri seti üzerinde uygun sonuçlar alındığı ve bir otonom araç benzetiminin başarıyla elde edildiği gösterilmektedir. Bu nedenle, önerilen gerçek zamanlı benzetim mimarisinin, otonom araç sistemlerinde çalışılan yenilikçi teknolojilerin geliştirilmesi sırasında kullanılabileceği savunulmaktadır.



Şekil 9. Çalışmada kullanılan farklı algoritmaların trafik işareti tanımada tüm veri seti üzerindeki genel doğruluk oranları
(The overall accuracy rates of the different algorithms used in the study for traffic sign recognition on the entire data set)

5. Sonuçlar (Conclusions)

Otonom araç teknolojisinin yaygınlaşmasıyla birlikte ortaya çıkan en önemli sorunlar, uygun görüntü işleme tekniklerinin belirlenmesi ve kapsamlı bir makine öğrenmesi yöntemi ile gerçek zamanlı karar verebilen sürüş algoritmasının oluşturulması olarak sıralanabilir. Ayrıca, bu modellerin gerçek bir test ortamında değerlendirilmesi ciddi donanım maliyetlerine sebebiyet vermektedir. Bu bağlamda, bu çalışmada önerilen benzetim mimarisi ile, yeni yaklaşımların sınanmasına donanım maliyeti olmaksızın olanak sağlayan bir benzetim yazılımı önerilmektedir. Ayrıca, şerit takibi yapabilen ve trafik işaretlerine göre yönlenebilen bir otonom sürüş modeli tanıtılmaktadır. Geliştirilen benzetim yazılımı ve önerilen otonom sürüş modeli, evrişimsel sinir ağları tabanlı YOLO ve R-CNN algoritmalarının farklı versiyonları ile değerlendirilmektedir. Böylece, önerilen sistemin güvenilir bir nesne tanıma modeli ile eğitim ve sınanması gerçekleştirilerek, elde edilen bulgular ışığında uygulanabilirliği değerlendirilmektedir. Buna göre, YOLO-v3, YOLO-v3 tiny, YOLO-v4, YOLO-v4 tiny, Fast R-CNN, Faster R-CNN, ve Mask R-CNN olmak üzere 7 farklı nesne tespit ve tanıma algoritması ile performans incelemesi ortaya konmaktadır. YOLO-v4 algoritmasının farklı trafik işaretlerinin birçoğunda en yüksek doğruluk oranına sahip sonuçlar ürettiği görülürken; YOLO-v3 tiny algoritmasının diğerlerine göre daha kısa sürede eğitilebildiği gözlemlenmektedir. R-CNN algoritması versiyonlarından en yüksek doğruluk oranına sahip olanı Mask R-CNN olarak tespit edilmekte olup; ürettiği sonuçların, YOLO-v4 algoritması ile elde edilen doğruluk oranı sonuçlarından biraz daha düşük olduğu görülmüştür. Öte yandan, diğer R-CNN algoritması versiyonlarının YOLO-v3 ile yakın sonuçlar ürettiği savunulurken; daha sade/hafif öğrenme modelleri olan YOLO-v3 tiny ve YOLO-v4 tiny algoritmalarının diğerlerine göre daha düşük doğruluk oranı ile çalıştığı tespit edilmiştir. Çalışmadan elde edilen bulgulara göre, R-CNN versiyonları için model eğitiminin, YOLO versiyonları ile model eğitiminden daha uzun sürdüğü tespit edilen bir diğer durumdur. Evrişimsel sinir ağlar tabanlı YOLO ve R-CNN algoritmalarının farklı versiyonları ile ortaya konan analizlerde doğruluk oranı ve eğitim sürelerine ilişkin bulguların yanı sıra, literatürde yer alan farklı çalışmalarla kıyaslamalara yer verilmektedir. Buna göre, çalışmada yürütülen analizler, daha önce Almanya, Belçika, Çin ve İngiltere'de yerel trafik işareti veri setleri üzerine yapılan çalışmalara benzer sonuçlar üretmektedir. Dolayısıyla, çalışmada önerilen benzetim ortamı üzerinde elde edilen verilere göre, gerçek dünyada imalatı yapılabilecek bir otonom araç tasarımının başarıyla oluşturulduğu ve sınandığı savunulabilir.

Gelecekte, önerilen benzetim sistemi için tanıtılan bileşenler gerçek bir test ortamında test edilecek ve değerlendirme sonuçları ele alınacaktır. Böylece, geliştirilen benzetim yazılımı ile sınanan bir otonom aracın, önerilen otonom sürüş modeli ile imalat sürecine ilişkin bir çalışma ortaya konması planlanmaktadır. Ayrıca, önerilen otonom sürüş modelinin daha yüksek doğruluk oranı ile trafik işareti tanıması için farklı ve daha büyük veri setleri ile değerlendirilmesi amaçlanmaktadır. Bu bağlamda, veri seti için yeni verilerin etiketlenmesi ve mevcut veriler üzerinde uygulanacak işlemler ile veri artırımı yapılması hedeflenmektedir.

Teşekkür (Acknowledgement)

Bu çalışma Manisa Celal Bayar Üniversitesi Bilimsel Araştırma Projeleri Koordinasyon Birimi tarafından "2021-065" kodlu proje ile desteklenmiştir. Yazarlar, Manisa Celal Bayar Üniversitesi Bilimsel Araştırma Projeleri Koordinasyon Birimi'ne teşekkür eder.

Kaynaklar (References)

1. Ali A., Lateef N. A., Shabnam G., SeyedAli G., Ali S. Users, planners, and governments perspectives: A public survey on autonomous vehicles future advancements. Transportation Engineering, 3, 100044, 2021.
2. Faisal, A., Yigitcanlar, T., Kamruzzaman, M., Currie, G. Understanding autonomous vehicles: A systematic literature review on capability, impact, planning and policy. Journal of Transport and Land Use, 12, 2019.
3. Hancock, P., Nourbakhsh, I., Stewart, J. On the future of transportation in an era of automated and autonomous vehicles. Proceedings of the National Academy of Sciences, 116 (16), 7684–7691, 2019.
4. Mutu M.A., Yakar F., Effects of driver characteristics, questionnaire type, and sign design on drivers' comprehension of traffic signs. Journal of the Faculty of Engineering and Architecture of Gazi University, 37 (2), 595-608, 2022.
5. Bingöl, M.S., Kaymak, Ç., Uçar, A. Derin Öğrenme Kullanarak Otonom Araçların İnsan Sürüşünden Öğrenmesi. Fırat Üniversitesi Müh. Bil. Dergisi, 31 (1):177-185, 2019.
6. Aytaç, Z., İşeri, İ., Dandil, B. Trafik Hız Sınırlama Levhalarının Derin Öğrenme ile Sınıflandırılması. 516-519. 10.36287/setsci.4.6.147, 2019.
7. Çetinkaya, M., Acarman, T. Trafik İşaret Levhası Tespiti için Derin Öğrenme Yöntemi. Akıllı Ulaşım Sis-temleri ve Uygulamaları Dergisi, 3 (2), 140-157, 2020.
8. Laguna, R., Barrientos, R., Blázquez, L.F., Miguel, L.J. Traffic Sign Recognition Application Based on Image Processing Techniques. Proceedings of the 19th World Congress The International Federation of Automatic Control, August 24-29, Cape Town, South Africa, 2014.
9. Zhu, Z., Liang, D., Zhang, S., Huang, X., Li, B., Hu, S. Traffic-Sign Detection and Classification in the Wild. 2110-2118. 10.1109/CVPR.2016.232, 2016.
10. Rajendran, S., Shine, L., Pradeep, R.,Vijayaraghavan, S. Fast and Accurate Traffic Sign Recognition for Self Driving Cars using RetinaNet based Detector. In 2019 International Conference on Communication and Electronics Systems (ICCES), 784-790, 2019.
11. Rajendran, S., Shine, L., Pradeep, R., Vijayaraghavan, S. Real-Time Traffic Sign Recognition using YOLOv3 based Detector. In 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1-7, 2019.
12. Artamonov, N., Yakimov, P. Towards Real-Time Traffic Sign Recognition via YOLO on a Mobile GPU. Journal of Physics: Conference Series, 1096, 012086, 2018.
13. Bounini, F., Gingras, D., Lapointe, V., Pollart, H. Autonomous Vehicle and Real Time Road Lanes Detection and Tracking. 1-6. 10.1109/VPPC.2015.7352903, 2015.
14. Fürst, J., Fierro, G., Bonnet, P., Culler, D. BUSICO 3D: Building Simulation and Control in Unity 3D. 326-327. 10.1145/2668332.2668380, 2014.
15. Kılıçaslan A.E., Kuş H., Evaluation of the hygrothermal performance of external thermal insulation applications on the outer walls of existing buildings. Journal of the Faculty of Engineering and Architecture of Gazi University, 36 (1), 89-104, 2021.
16. Nagpal, L., Jaglan, M., Kathait, A., Mathur, A., Vichare, A. SOUL: Simulation of Objects in Unity for Learning. 8-13. 10.1109/ICCT46177.2019.8968786, 2019.
17. So, H.Y., Chen, P., Wong, G., Chan, T. Simulation in medical education. Journal of the Royal College of Physicians of Edinburgh, 49, 52-57, 2019.
18. Shah, A., Mai, C. L., Shah, R., Levine, A. I. Simulation-Based Education and Team Training. Otolaryngologic clinics of North America, 52 (6), 995–1003, 2019.
19. Kikolski, M., Study of Production Scenarios with the Use of Simulation Models. Procedia Engineering, 182, 321-328, 2017.
20. Balta M., Özçelik İ., A proposal of SDN based VANET architecture for urban intersection management systems. Journal of the Faculty of Engineering and Architecture of Gazi University, 34 (3), 1451-1468, 2019.
21. Yurtsever, E., Lambert, J., Carballo, A., Takeda, K. A survey of Autonomous Driving: Common Practices and Emerging Technologies. IEEE Access, 8, 58443-58469, 2020.
22. Tuncali, C., Fainekos, G., Ito, H., Kapinski, J. Simulation-based Adversarial Test Generation for Autonomous Vehicles with Machine Learning Components. In 2018 IEEE Intelligent Vehicles Symposium (IV), 1555-1562, 2018.
23. Aydın, M. M., Akgöl, K., Günay, B. The investigation of different chicane applications in traffic calming studies using driving simulator, Journal of the Faculty of Engineering and Architecture of Gazi University, 34 (4), 1793-1806, 2019.

24. Rosique, F., Navarro, P., Fernández, C., Padilla, A. A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research. Sensors, 19 (3), 2019.

25. Borraz, R., Navarro, P. J., Fernández, C., Alcover, P. M. Cloud Incubator Car: A Reliable Platform for Autonomous Driving. Applied Sciences, 8 (2), 303, 2018.

26. Schöner, H. P. Simulation in Development and Testing of Autonomous Vehicles. In 18. Internationales Stuttgarter Symposium, 1083-1095, 2018.

27. Buyuksalih, I., Bayburt, S., Buyuksalih, G., Baskaraca, A. P., Karim, H., Rahman, A. A. 3D Modelling And Visualization Based On The Unity Game Engine--Advantages And Challenges. ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences, 4, 2017.

28. Solmaz, Ö., Özçevik, Y, Baysal, E., Ökten, M., Panpallı, A. Gerçek Zamanlı Simülasyonda Şerit Takibi için Otonom Araç Tasarımı, 2nd International Symposium on Automotive Science and Technology (ISASTECH), 310-316, 8-10 Eylül, Ankara, Türkiye, 2021.

29. Canny, J., A Computational Approach to Edge Detection. IEEE Trans. Pattern Anal. Mach. Intell., 8 (6), 679–698, 1986.

30. Yasmina, D., Karima, R., Ouahiba, A. Traffic signs recognition with deep learning. In 2018 International Conference on Applied Smart Systems (ICASS), 1-5, 2018.

31. Yang, Y., Luo, H., Xu, H., Wu, F. Towards Real-Time Traffic Sign Detection and Classification. IEEE Transcations on Intelligent Transportation Systems, 17 (7), 2022-2031, 2016.

32. Solmaz, Ö., Özçevik, Y, Baysal, E., Ökten, M., Çulha, A. Trafik Levhası Tanıma için Sokak Görüntüleri Kullanarak Veri Seti Oluşturma ve Sınama, 2nd International Symposium on Automotive Science and Technology (ISASTECH), 344-349, 8-10 Eylül, Ankara, Türkiye, 2021.

33. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C. The German Traffic Sign Recognition Benchmark: A Multi-class Classification Competition. In IEEE 2011 International Joint Conference on Neural Networks, 1453-1460, July, 2011.

34. Jain, A., Mishra, A., Shukla, A., Tiwari, R. A Novel Genetically Optimized Convolutional Neural Network for Traffic Sign Recognition: A New Benchmark on Belgium and Chinese Traffic Sign Datasets. Neural Processing Letters, 50 (3), 3019-3043, 2019.

35. Changzhen, X., Cong, W., Weixin, M., Yanmei, S. A Traffic Sign Detection Algorithm Based On Deep Convolutional Neural Network. In 2016 IEEE International Conference on Signal and Image Processing (ICSIP), 676-679, August, 2016.

36. Qian, R., Zhang, B., Yue, Y., Wang, Z., Coenen, F. Robust Chinese Traffic Sign Detection And Recognition With Deep Convolutional Neural Network. In 2015 11th International Conference on Natural Computation (ICNC), 791-796, August, 2016.