T.C.
NECMETTİN ERBAKAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OTONOM ARAÇ ÜZERİNDE ARTIRILMIŞ GERÇEKLİK UYGULAMASI

Mehmet BİLBAN

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Mayıs-2020
KONYA

TEZ KABUL VE ONAYI

Mehmet Bilban tarafından hazırlanan "**OTONOM ARAÇ ÜZERİNDE ARTIRILMIŞ GERÇEKLİK UYGULAMASI**" adlı tez çalışması 05/05/2020 tarihinde aşağıdaki jüri tarafından oy birliği ile Necmettin Erbakan Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Prof. Dr. Erkan ÜLKER	.....
Danışman Dr. Öğr. Üyesi Yusuf UZUN	.....
Üye Prof. Dr. Sabri KOÇER	.....

Yukarıdaki sonucu onaylarım.

Prof. Dr. Süleyman Savaş DURDURAN
FBE Müdürü

Bu tez çalışması Necmettin Erbakan Üniversitesi Bilimsel Araştırma Koordinatörlüğü tarafından 181351003 nolu proje ile desteklenmiştir.

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Mehmet BİLBAN

Tarih: 05/05/2020

ÖZET

YÜKSEK LİSANS TEZİ

OTONOM ARAÇ ÜZERİNDE ARTIRILMIŞ GERÇEKLİK UYGULAMASI

Mehmet BİLBAN

Necmettin Erbakan Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Yusuf UZUN

2020, 57 Sayfa

Jüri
Prof.Dr. Erkan ÜLKER
Prof.Dr. Sabri KOÇER
Dr. Öğr. Üyesi Yusuf UZUN

Otonom araç geliştirme teknolojisinin ilerlemesiyle, araç içinde ve araç dışında kullanılmak üzere ek uygulamalara olan ihtiyacımız da artmaktadır. Literatürdeki, geliştirilen birçok uygulama araç üzerinden alınan verileri doğrudan araç içerisinde monitör, ekran üstü göstergeler ve donanımsal cihazlar üzerinde göstermeye yöneliktir. Geliştirilen bu uygulamalar, sadece tanımlı bir problem ve belirli bir otonom sistem için geliştirilmiştir. Bu çalışmada, temel otonom bir araç yazılım alt yapısı ve Android cihazlar üzerinde çalışabilen bir mobil Artırılmış Gerçeklik uygulaması geliştirilmiştir. Mobil Artırılmış Gerçeklik uygulamasının, hem araç içerisinde hem de araç dışında hizmet vermesi ve çeşitli alanlarda geliştirilen birden fazla otonom sistemlerle birlikte kullanılabilirliği en önemli özelliklerindendir. Mobil Artırılmış Gerçeklik uygulaması araç içerisinde seyahat eden tüm yolcular için, hız, enerji tüketimi, anlık direksiyon hareketleri ve dış ortamda bulunup sistem tarafından tanınabilen nesneleri göstermektedir. Böylece, araç içerisindeki kişilerin kendilerini daha güvende hissedeceği ve otonom bir aracın insanlar tarafından daha güven duyulan bir araç olduğunu düşünmesi hedeflenmiştir. Aynı zamanda, bu uygulama ile olası kaza durumlarında araç üzerinden alınan geçmiş verilerin geriye dönük sorgulanabileceği görülmüştür. Bu çalışmada geliştirilen sunucu-istemci mimari alt yapısı ile ortamdan ve konumdan bağımsız otonom araç aktiviteleri izlenebilmektedir. Bu tez çalışmasında, otonom araç üzerinde Artırılmış Gerçeklik uygulamasının kullanımı amaçlanmıştır ve bir senaryo üzerinde uygulanabilirliği gösterilmiştir.

Anahtar Kelimeler: Artırılmış Gerçeklik, Derin Öğrenme, Mobil Uygulama, Otonom Araç

ABSTRACT

MS THESIS

AUGMENTED REALITY APPLICATION ON AUTONOMOUS VEHICLE

Mehmet BİLBAN

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF NECMETTİN ERBAKAN UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE
IN COMPUTER ENGINEERING

Advisor: Asst. Prof. Dr. Yusuf UZUN

2020, 57 Pages

Jury
Prof.Dr. Erkan ÜLKER
Prof.Dr. Sabri KOÇER
Asst. Prof. Dr. Yusuf UZUN

With the advancement of autonomous vehicle development technology, we also need additional applications to be used inside and outside the vehicle. Many applications developed in the literature are intended to display data received from the vehicle directly on the monitor, on-screen displays, and hardware devices. These applications are developed only for a defined problem and for a specific autonomous system. In this study, a basic autonomous vehicle software infrastructure and a mobile Augmented Reality application that can run on Android devices have been developed. The most important feature of the Mobile Augmented Reality application is that it serves both inside and outside the vehicle and can be used with multiple autonomous systems developed in various fields. The Mobile Augmented Reality application shows speed, energy consumption, instant steering movements, and objects that can be found in the external environment and recognized by the system for all passengers traveling in the vehicle. Thus, it is aimed at people in the vehicle who will feel safer and think that an autonomous vehicle is a vehicle that is more trusted by people. At the same time, with this application, it has been seen that historical data received from the vehicle can be queried retrospectively in case of possible accidents. With the server-client architectural infrastructure developed in this study, autonomous vehicle activities independent of environment and location can be monitored. In this thesis, the use of Augmented Reality application on the autonomous vehicle is intended and its applicability is shown on a scenario.

Keywords: Augmented Reality, Autonomous Vehicle, Deep Learning, Mobile Application

ÖNSÖZ

Tez çalışmam boyunca bilgisi ve yardımları ile beni yönlendiren değerli danışmanım Dr. Öğr. Üyesi Yusuf UZUN'a, tüm eğitim hayatım boyunca benden maddi ve manevi desteklerini esirgemeyen her zaman yanımda olan sevgili aileme ve eşime sonsuz teşekkürler.

Mehmet BİLBAN
KONYA-2020

İÇİNDEKİLER

ÖZET **1**

ABSTRACT **2**

ÖNSÖZ **3**

İÇİNDEKİLER **4**

SİMGELER VE KISALTMALAR **5**

ŞEKİLLER DİZİNİ **6**

ÇİZELGELER DİZİNİ **7**

1. GİRİŞ **8**

2. KAYNAK ARAŞTIRMASI **12**

3. MATERYAL VE YÖNTEM **14**

3.1 Otonom Araç Mimarisi ve Geliştirilmesi 14
3.1.1 RACECAR temel yazılım mimari bileşenleri 15
3.1.2 RACECAR'ın otonom araca dönüştürülmesi 15
3.1.3 Robot işletim sistemi mimarisi ve kavramları 16
3.1.4 Veri toplama 17
3.1.5 Ağ yapısı 20
3.2 Mobil Artırılmış Gerçeklik Uygulaması Mimarisi ve Geliştirilmesi 22
3.2.1 Artırılmış gerçeklik için bilgisayar grafikleri temelleri 22
3.2.2 Artırılmış gerçeklik izleme ve takip sistemi mimarisi 30
3.2.3 Android sinir ağları ve tensorflow lite ile nesne tanıma 33
3.2.4 Otonom bir araç için mobil AR uygulaması geliştirilmesi 38

4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA **40**

4.1 Otonom Araç Konfigürasyonu Sonuç Bulguları 40
4.2 Otonom Araç – Mobil AR Uygulaması Entegrasyon İşlemi ve Sonuçları 46

5. SONUÇLAR VE ÖNERİLER **51**

5.1 Sonuçlar 51
5.2 Öneriler 52

KAYNAKLAR **54**

ÖZGEÇMİŞ **57**

SİMGELER VE KISALTMALAR

Simgeler

δ	: Radyan olarak direksiyon açı değeri
θ	: Aracın ağırlık merkezinden yörünge düzlemindeki yönelim
$+X_{ih}$, $+Y_{ih}$	: Aracın ön lastiklerinde bir değişim olduğundaki değişim miktarı

Kısaltmalar

AG	: Artırılmış Gerçeklik (Augmented Reality)
SAE	: Otomotiv Mühendisleri Topluluğu (Society of Automotive Engineers)
OA	: Otonom Araç
İHA	: İnsansız Hava Araçları
NHTSA	: Amerika Birleşik Devletleri Ulaştırma Bakanlığı Ulusal Karayolu Trafik Güvenliği İdaresi
SG	: Sanal Gerçeklik (Virtual Reality)
ADAS	: Gelişmiş Sürücü Yardım Sistemleri
RİS	: Robot İşletim Sistemi (ROS)
GUI	: Grafiksel Kullanıcı Arayüzü
CNN	: Evrişimsel Sinir Ağları (Convolutional Neural Networks)
YSA	: Yapay Sinir Ağları
DNN	: Derin Sinir Ağları (Deep Neural Networks)
TSA	: Tekrarlayan Sinir Ağları (Recurrent Neural Networks)
DİA	: Derin İnanç Ağları
YOK	: Yığın Otomatik kodlayıcı
ÇKA	: Çok Katmanlı Algılayıcı
KBM	: Kısıtlı Boltzmann Makineleri (RBM)
OK	: Otomatik kodlayıcı (AE)
ASA	: Android Sinir Ağları
İT	: İşaretçi-Tabanlı (Marker-Based)
İZ	: İşaretsiz (Markerless)
RACECAR	: Rapid Autonomous Complex-Environment Competing Ackermann-steering Robot
VESC	: Vedder Electronic Speed Controller
ReLU	: Doğrultulmuş Doğrusal Birim

ŞEKİLLER DİZİNİ

Şekil 3.1. MIT RACECAR 14
Şekil 3.2. Otonom araç bileşenleri 16
Şekil 3.3. RİS haberleşme diyagramı 17
Şekil 3.4. Gerçek zamanlı veri toplama işleminde, Rqt ile RİS üzerindeki düğümlerin gösterimi 20
Şekil 3.5. NVIDIA evrişimsel sinir ağı 21
Şekil 3.6. Geometrik grafik gösterimi (V: Vertex, n: Normal) 23
Şekil 3.7. 3B grafik dönüşüm boru hattı 23
Şekil 3.8. Koordinat dönüşüm hattı 24
Şekil 3.9. Koordinat dönüşüm hattında yer alan görünüm dönüşümü 26
Şekil 3.10. Kameranın kesik piramit görünümlü görüş alanı 28
Şekil 3.11. Görünüm alan dönüşümü 29
Şekil 3.12. İşaretçi tabanlı mobil artırılmış gerçeklik uygulamasının elektronik bir devre ile gerçek zamanlı iletişimi 32
Şekil 3.13. İşaretsiz mobil artırılmış gerçeklik uygulamasının elektronik bir devre ile gerçek zamanlı iletişimi 33
Şekil 3.14. Android yapay sinir ağları uygulama programlama arayüzü 34
Şekil 3.15. Android yapay sinir ağları uygulama programlama arayüzü akışı 35
Şekil 3.16. TensorFlow Lite tarafından desteklenen mobil platformlar 36
Şekil 3.17. TensorFlow Lite ile nesne tanıma işlemi 37
Şekil 3.18. RACECAR sunucu – mobil AR iletişim 38
Şekil 3.19. Mobil AR- Flask Sunucu RESTful Web servis mimari 39
Şekil 4.1. Gerçek zamanlı veri toplama işleminde sistem aktivite diyagramı 40
Şekil 4.2. Gerçek zamanlı veri toplama işleminde kullanılan ortamın krokisi 41
Şekil 4.3. Gerçek zamanlı veri toplama işleminde alınan orijinal boyutta bir görüntü 41
Şekil 4.4. Orijinal görüntüye kırpma işlemi uygulandığında elde edilen görüntü 41
Şekil 4.5. Direksiyon açı değerlerinin dağılımı grafiği 42
Şekil 4.6. Daha dengeli bir dağılımı ifade eden histogram değerleri 42
Şekil 4.7. MSE kayıp fonksiyon grafiği 45
Şekil 4.8. Eğitim ve test veri seti üzerinde modelin doğruluk değeri grafiği 45
Şekil 4.9. Parkur ortamı için levha tasarımı ve ölçüleri 46
Şekil 4.11. Eşit aralıklarla yerleştirilen levhaların mobil AR ile ilk tanıma işlemi 48
Şekil 4.12. Stop levhası tanıma işlemi 49
Şekil 4.13. Yaya tanıma işlemi 49
Şekil 4.14. Trafik levhası tanıma işlemi 50

ÇİZELGELER DİZİNİ

Çizelge 3.1. Rospy ile RİS üzerinde düğüm başlatma kodları……18
Çizelge 3.2. Python Keras ile uçtan uca öğrenme modeli……...21
Çizelge 3.3. Nesne tespiti dönüş değerleri……..37
Çizelge 4.1. Otonom bir araç için veri seti özellikleri…….40
Çizelge 4.2. Işık değeri dönüşüm işlemi…….43

1. GİRİŞ

Otonom, Fransızca kökenli bir kelime olup kendi iradesi ile hareket eden anlamına gelmektedir. “Yapay zeka”, ”robot” ve “kendi kendini süren araba” gibi kelimeler günümüzde yaygın olarak kullanılmaktadır ancak “otonom” terimi ile karşılaşma olasılığımız diğerlerine göre daha düşüktür. Bu terimin yaygın olarak kullanılmamasındaki sebeplerden biriside bu terimin tarihteki kullanım sıklığı ile alakalı olabilir.

Otonom terimi, geriye dönük zaman içerisinde bakıldığında matematik ve ağ gibi alanlarda kullanıldığı görülür. Yeni bir terim olması ve yeni teknolojilerin ortaya çıkmasıyla farklı boyutları kapsamaya devam etmektedir.

Daha modern bir anlayışla otonom bir sistemde, donanım ve yazılım bir işlemi gerçekleştirebilmek için birlikte çalışırlar. Bir sistemin otonom olabilmesi için, sistemin bilgi toplayabilmesi, bilgilerden çıkarım yapabilmesi ve asıl ana hedefe ulaşabilmek için bir eylem gerçekleştirmesi gereklidir.

Otonom sistemler, insan desteği olmadan ve sensörlerden aldığı verilerle çevresini algılayan, navigasyon ve yönlendirme yeteneğine sahip araçlar olarak tanımlanabilir.

"Sürücüsüz otomobil" ve "Kendi kendini süren otomobil" terimleri, otonom araçları belirtmek için yaygın olarak kullanılırlar. Bu çalışmada, “Otonom Araç” veya kısaca “OA” terimi, aksi belirtilmedikçe, kendi kendini süren bir otomobili ifade eder.

Otonom sistemler, endüstriyel robotlar, insansız hava araçları, sürücüsüz otomobiller, insansız su altı araçları, insansız tarım araçları, insansız hava-uzay araçlarının tamamını içeren bir alandır. Otomobil endüstrisi diğer alanlara göre daha fazla tüketici pazarına sahiptir. Bu yüzden, otonom araçların gelişmesinde daha fazla artış görülmüştür.

Otonom araç teknolojisindeki gelişmeler, son on yılda, yapay zeka teknolojisi, otomotiv ve bilgi teknolojilerindeki rekabet ve ciddi yatırımlardaki gelişmelerle birlikte hızla ilerlemiştir (Rastogi 2017).

SAE, uluslararası sürüş otomasyon seviyelerini, tam otomasyonlu ve otomasyonsuz olarak 6 seviyeden oluştuğunu belirtmiştir (Committee 2016). Burada Seviye 0, otomasyonsuz ve tüm dinamik sürüş kontrollerinin bir sürücü tarafından gerçekleştirildiğini ifade eder.

ADAS, uzun yıllardır araçların bir parçası olmuştur. Şerit Kalkış Asistanı, Acil Durum Frenleme Sistemleri, Park Asistanı, Dinamik Hız Sabitleyici ADAS özelliklerinden bazılarıdır. Bu özelliklere sahip araçlar Seviye 1 Otonom Araç olarak sınıflandırılır. Bu sınıflandırma, SAE uluslararası topluluk (Committee 2016) tarafından formüle edilmiş ve 2016 yılında NHTSA tarafından kabul edilen şartnamelerden alınmıştır (Rastogi 2017). Seviye 2, 3 ve 4 ise, otomasyonsuz bir araçtan tam otomasyonlu ve otomatik pilot yardımıyla sınırlı sürüş modunda çalışan araçları ifade etmektedir. Dinamik sürüş görevleri ve otomatik sürüş sistemi ile insansız tam otomasyonlu sistem ise, Seviye 5 olarak belirtilmiştir.

AG (Artırılmış Gerçeklik) teknolojisi, 1960'lara kadar uzanmaktadır. İlk olarak geliştirilen sistem alt yapısı hem AG hem de SG (Sanal Gerçeklik) için kullanılmıştır (E.Sutherland 1968).

Günümüzde AG ise, bilgisayar ortamında oluşturulan sanal görüntülerin gerçek dünya içerisinde yerleştirilip kullanılmasına izin veren teknolojidir. AG teknolojisi, otomotiv, endüstri, tıp, robot, inşaat ve eğitim gibi birçok alanda yaygın olarak kullanılmaktadır.

AG, birçok kamu veya özel şirketlerin otonom sistemleri geliştirmeden önce başvurduğu bir teknolojidir. Bu teknoloji, tasarlanan algoritmalar, sistem entegrasyonu, iletişim protokolleri ve görüntüleme şekli gibi simüle edilmesi gereken birçok alanda yön gösteren bir pusula olarak kullanılmaktadır.

Otonom araçlar yakın gelecekte kişisel ulaşımın vazgeçilmez teknolojisi haline gelecektir. Otonom araç üreticileri, insan gözünün gerçek dünyayı nasıl algıladığını araç içerisinde gösterebilmenin yollarını araştırmışlar ve çözüm olarak AG teknolojisinin, kullanılabilecek teknolojilerden biri olduğu sonucuna varmışlardır (Fredriksson, Kulcsar, and Sjöberg 2015). Bu teknolojiyle, sensörlerden ve kameralardan alınan verilerin gerçek dünya nesneleri ile birleştirilip sunabilmesi hedeflenmiştir. AG teknolojisi hız, enerji durumu, yol durumu ve araç hakkındaki karmaşık tüm durumları kolay bir şekilde gösterebilmektedir.

Otomobil endüstrisi, sayısız hükümet yönetmeliği ve diğer kısıtlamalarla karşı karşıyadır. Otonom aracın yolda ilerlerken hız bilgisi, trafik ışıklarındaki davranışları, seyir halinde ilerlerken tepkilerinin kayıt altında tutulması gerekmektedir. Olası bir aksilik veya kaza anında geriye dönük tüm bilgilere ulaşılabiliyor olmalıdır. Zaman içerisinde, otonom araçların sayısı arttıkça AG teknolojisinin araç içinde kullanılan teknolojilerden biri haline gelmesi beklenmektedir. Otonom aracın içerisinde seyahat

eden yolcular, aracın ne kadar bir hızla gittiğini, yakıt veya enerji durumunu ve yol durumu hakkındaki bilgileri sunan bir ekrandan takip edebildikleri için kendilerini daha güvende ve huzurlu hissedeceklerdir.

Bugün, sürücülerin en büyük sorunlarından birisi, hem navigasyon sistemine odaklanmak hem de sonra dış dünyaya yeniden odaklanmak gibi, sürüş sırasında birçok görevi yönetmektir. Bazı sürücüler, aynı anda çoklu işler yapmakta iyi değillerdir. Bu tarz problemlerin üstesinden gelebilmek için, AG sistemlerinin, sürücülere yardımcı olduğu görülmektedir (Modi 2018).

Günümüzde otonom araçların büyük ilgi görmesi, birçok devlet ve özel kuruluşların bu alanda yapmış oldukları çalışmalar, otonom sistemlerin bu tez çalışmasında da ana konu haline gelmesinde büyük bir rol oynamıştır.

Gelecek on yıl içerisinde otonom sistemlerin gelişmesindeki artışla birlikte araştırmacılar ve geliştiricilerin en çok zorlanacağı sorular şunlar olacaktır: Otonom bir sistem tarafından gerçek dünya nasıl algılanmaktadır? Otonom bir sistem, topladığı verileri veya çıkarımlar sonucunda elde ettiği sonuçları nasıl görüntülemelidir? Bu çalışma, bu tarz sorulara cevap aramaktadır.

Bu tez kapsamında, yüksek lisans tez projesi ile alınan otonom araç sistemi üzerinde mobil AG uygulaması geliştirilmiştir. Mobil AG uygulaması üzerinde araç üzerinden anlık olarak hız, direksiyon açı değeri değişimi, enerji tüketimi ve test senaryosunda belirtilen nesnelerin tanıma işlemleri gerçekleştirilmiştir. Belirtilmiş test senaryoları ve araç üzerinden alınan veriler araç içerisinde ve araç dışında kullanılabilecek şekilde tasarlanmıştır.

Bu çalışmada, araç üzerinden veri alabilmek ve istemciler ile paylaşabilmek için sunucu-istemci mimari yapısı geliştirilmiştir. Bu mimari ile mobil AG uygulamasının birçok otonom sistem alt yapısında da kullanılabilmesi hedeflenmiştir. Geliştirilen sunucu-istemci mimari yapısı ile birden fazla istemciye aynı anda cevap verilebilmektedir (Uzun 2019).

Çalışmada kullanılan otonom araç sistemi, kapalı bir ortamda ve planlanan test senaryosu üzerinde hareket edebilme kabiliyetine sahiptir. Engelden kaçma, trafik işaretlerine uyarak hareket edebilme ve en uygun yol seçimi gibi işlemler bu tez kapsamı dışındadır.

Bu çalışmada, geliştirilen AG uygulaması, aşağıda belirtilen otonom sistemlerde kullanılabilecek şekilde bir altyapıya sahiptir.

- Kendi Kendini Süren Araçlar
- İnsansız Hava Araçları
- İnsansız Hava-Uzay Araçları
- İnsansız Denizaltı Araçları
- İnsansız Tarım Araçları
- Termik ve Nükleer Santraller
- Nesnelerin İnterneti Uygulamaları

Bu tez çalışmasında, otonom araçlar için AG uygulaması geliştirilirken ihtiyaç duyulan tüm yazılım alt yapısı aşağıda sunulmuştur:

- Kendi kendini süren araç yazılımı
- Otonom araç için Derin Öğrenme yazılımı
- Sunucu yazılımı
- Mobil Artırılmış Gerçeklik Uygulaması
- Nesne tanıma için Mobil Derin Öğrenme yazılımı

Bu çalışmada, geliştirilen yazılım alt yapısı üç ana mimari altında sunulacaktır.

1. Otonom Araç Mimarisi ve Geliştirilmesi
2. Mobil Artırılmış Gerçeklik Uygulama Mimarisi ve Geliştirilmesi
3. Otonom bir Araç için Mobil AR Uygulaması Geliştirilmesi

Otonom Araç Mimarisi ve geliştirilmesinde, MIT tarafından açık kaynak kodlu olarak geliştiricilere açılan MIT-RACECAR kullanılmıştır. Tüm yazılım ve donanım alt yapısı bu araç üzerinde çalışacak şekilde geliştirilmiştir.

Mobil Artırılmış Gerçeklik Uygulaması için, Android işletim sistemleri için bir mobil uygulama yazılmıştır. Geliştirilen bu mobil uygulama tüm Android cihazlarda çalışabilecek şekilde tasarlanmıştır.

Sunucu-İstemci Mimarisi için, sunucu taraflı bir yazılım geliştirilmiştir. Bu yazılım, otonom araçtan aldığı verileri mobil cihaza gönderir. Aynı zamanda, otonom aracın çevre birimlerinden alınan verileri işleyerek istemci ile paylaşır.

2. KAYNAK ARAŞTIRMASI

Otonom su altı araçları, insan müdahalesine gerek kalmadan uzun süreli ve uzun mesafeli görevleri yerine getirme özelliğine sahiptir. Su altı otonom araçları konumlandırılırken arazi tabanlı navigasyon yöntemleri önerilmiştir. Arazi tabanlı navigasyonun en önemli özelliklerinden birisi ise yardımcı sinyal ve cihaz gerekmemesidir (Melo and Matos 2017).

Artan dünya nüfusu ile birlikte tarımda üretimin artarak devam etmesi gerekmektedir. Bu durumda, verimi artırmak için otonom araçlar dikkate alınması gereken konuların başında gelmektedir. Son yıllarda farklı teknikler ve algoritmalar ile tarımsal üretimde birçok gelişme elde edilmiş olsa da çiftçilerin otonom tarım araçları hakkındaki düşüncelerinin olgunlaşması için daha fazla çalışma yapılması gerektiği belirtilmiştir (Mousazadeh 2013).

Otonom araçlardaki son teknolojik gelişmeler sonucunda, yakın zaman içerisinde otonom araçların, araç filolarına katılması beklenmektedir. Bu araçlar, sürücülü araçlara göre daha yumuşak hızlanma ve yavaşlama hareketleri gösterebileceği için trafikte meydana gelen dalgalanmaları azaltabileceği belirtilmiştir (Stern vd. 2018).

Sürücüsüz araçlar, insan kaynaklı dur-kalk trafiğini azaltmak için tasarlandığında sürücülü araçlara göre araç emisyon değerlerini düşürülebileceği gösterilmiştir (Stern et al. 2018).

Birçok otomobil baş üstü gösterge paneline (HUD) sahiptir. Bu teknoloji, AG uygulamasını mümkün kılmaktadır (Tönnis et al. 2005).

AG'nin, otomobillerde sürücünün dikkatini bir noktaya toplayabilen ve tehlikeli durumlarda sürücüyü uyarabilen bir teknoloji olduğu düşünülmektedir (Tönnis et al. 2005).

Otonom araç teknolojisinin gelişmesiyle birlikte zaman içerisinde güvenlik sorunlarına daha çok odaklanılacağı ifade edilmiş ve tüm bu işlemlerin AG uygulamalarının gelişmesine yol açacağı belirtilmiştir (Fredriksson, Kulcsar, and Sjöberg 2015).

Otonom araçlar için aktif güvenlik önlemlerini araştırabilmek için yeni test yöntemlerine ve donanımlarına ihtiyaç vardır. Gelecekte konfor ve güvenlik sistemlerini geliştirirken araştırılacak konulardan birisi de AG uygulaması olacağı öngörülmektedir (Fredriksson, Kulcsar, and Sjöberg 2015).

2030 yılına kadar açık yollarda tam otonom araçların görülmesi beklenmektedir. Üreticiler ve araştırmacılar araştırmalarını aradaki otomasyon seviyelerine odaklamaktadır. Bu açıdan birçok proje yapılmıştır. Bunlar arasında, bir Fransız projesi olan LRA projesi (Fransız Yerelleştirme ve AG), otonom sürüş için AG ara yüzü tasarlamayı amaçlamaktadır (Fredriksson, Kulcsar, and Sjöberg 2015)

Japonya Otomobil Araştırma Enstitüsü tarafından geliştirilen AG aracı, sürücüyü gerçek bir çarpışma riskine sokmadan, gerçekçi trafik kazası ve çarpışma senaryoları oluşturmak için geliştirilmiştir (Fredriksson, Kulcsar, and Sjöberg 2015).

AG teknolojilerinin kullanılması, saha dışı arazilerdeki otonom araç testlerini sanal nesnelerle tamamlar. Bu, üreticilerin ve sistem tedarikçilerinin, yeni işlevlerin müşteriye kabulünü kapsamlı bir şekilde araştırmalarını ve yanlış yöndeki gelişmeler riskini azaltmalarını sağladığı düşünülmüştür (Fredriksson, Kulcsar, and Sjöberg 2015).

Bazı araştırmacılar tarafından, endüstriyel robotların programlanmasında AR teknolojisinin kullanılabilecek teknolojilerden biri olduğu sonucuna varılmıştır (Pan et al. 2012)

Her ne kadar robotlar otomotiv fabrikalarında daha otonom hale getirilse de çalışanların üretim, montaj ve lojistik sürece dahil olabilmesi için Artırılmış Gerçeklik gözlükleriyle kullanması gerektiği bazı araştırmacılar tarafından belirtilmiştir (Rüßmann et al. 2015).

İnsansız Hava Aracı'nın (İHA) inşaat sahalarında kullanılmasıyla birlikte şantiye alanlarındaki sorunları daha net görmek amaçlanmaktadır. Yapılan bir çalışmada, İHA'lardan alınan gerçek görüntülerle ve AG uygulamasındaki sanal nesnelerin birleştirilmesiyle mühendislerin şantiye alanlarını gelişmiş bir bakış açısıyla görmeleri sağlanmıştır (Wen and Kang 2014).

MIT Hava-Uzay kontrol laboratuvarında Siber Fiziksel Sistemlerin hızlı bir şekilde prototipleşmesi için bir mimari önerilmiştir. Siber Fiziksel Sistemler (CPS), fiziksel sistemlerin kontrol, hesaplama ve iletişim teknolojileriyle entegrasyonuna dayanan mühendislik sistemlerini ifade etmektedir. Bu mimari, simüle edilmiş bir dış ortamda planlama ve öğrenme algoritmalarının kontrollü bir şekilde test edilmesini sağlayan deneysel ortamı ifade etmektedir (How et al. 2015). Siber Fiziksel Sistemler için geliştirilen AG alt yapısı kullanılarak, otonom araç test süreçlerini kolaylaştırdığı belirtilmiştir.

3. MATERYAL VE YÖNTEM

Bu bölümde ele alınan konular şunlardır: Bölüm 3.1.'de Otonom araç mimarisi ve geliştirilmesi için gerekli olan yazılım ve donanım özelliklerine değinilmiştir. Bölüm 3.2'de Mobil AG uygulaması mimarisi ve geliştirilmesi için gerekli olan yazılım özellikleri ve mobil YSA'ların nasıl kullanılacağı anlatılmıştır.

3.1 Otonom Araç Mimarisi ve Geliştirilmesi

MIT RACECAR, robotik araştırma ve eğitim için açık kaynaklı 1/10 ölçekli bir otonom araç platformudur. Bu platform, Traxxas RC şasisi üzerine yerleştirilmiş, sensörler ve bilgisayar donanımı barındırmaktadır. RACECAR'ın tasarımı ve geliştirilmesi, Lincoln Laboratuvarı'nın BeaverWorks Girişimi, Havacılık ve Uzay Bilimleri Bölümü ve Massachusetts Institute of Technology'deki Bilgi ve Karar Sistemleri Laboratuvarı tarafından yapılmıştır ("MIT RACECAR Mobile Platform" 2017). RACECAR ve donanım bileşenleri Şekil 3.1'de gösterilmektedir ("Openzeka" 2019).

Şekil 3.1. MIT RACECAR

RACECAR'ın özellikleri aşağıdaki gibidir:

- 'TRAXXAS' Slash 4X4 Platinum Edition
- 4200 mAh LiPo 7,4V 25C güç kaynağı
- Dijital Balance Profesyonel Şarj Cihazı
- Male Traxxas To Female XT60 Batarya Konnektörü

- 2 Şarj Portlu USB 3.0 7-Portlu Hub (UH720)
- ‘TP-Link Archer C7’ Wireless Çift Band Gigabit Router (AC1750)
- VESC FOCBOX Motor Kontrolcüsü
- ‘SparkFun 9DoF Razor’ IMU M0
- ‘Stereolabs ZED’ kamera
- ‘RPLidar A2M6’, Scanse Sweep Lidar
- ‘Logitech F710’ Kablosuz Joystick
- ‘Lexar’ 64GB SDXC UHS-I 633X 95Mb/sn
- ‘NVIDIA Jetson TX2’ geliştirici Kiti
- MARC Güç Bankası

3.1.1 RACECAR temel yazılım mimari bileşenleri

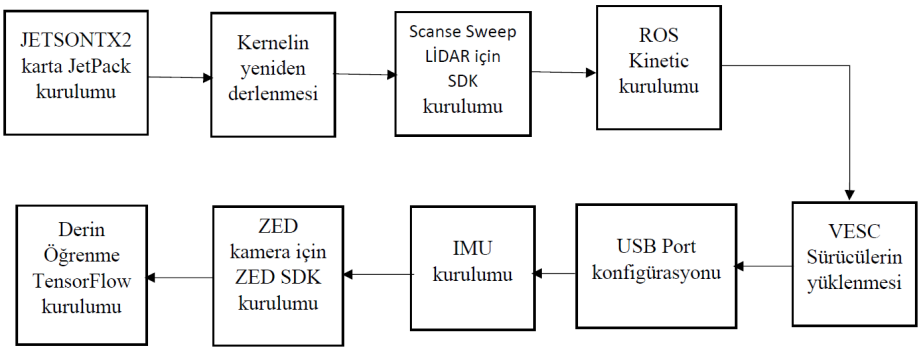
Araç üzerinde bütünleşik Jetson TX2 geliştirici kiti bulunmaktadır. Bu kart, Ubuntu 16.04 ile çalışmaktadır. RACECAR platformuna, araç üzerindeki bütünleşik diğer sistemlerle bir arayüz oluşturabilmek için Robot İşletim Sistemine dayalı bir yazılım mimarisi kurulmuştur (Quigley et al. 2009). Yazılım bileşenleri şunlardır:

- Farklı önceliklere sahip birden fazla düğüme farklı komutlar gönderilmesine izin veren bir kontrol çoklayıcı
- IMU, entegre LiDAR, kamera bileşenleri için sürücüler ve RİS sağlayıcı
- Jetson üzerinden motor kontrolcüsüne, USB arabirimi üzerinden komut vermek için kullanılan düğümler
- Mekânsal konfigürasyon için dönüşüm çerçeveleri
- Düşük hızlarda araç kontrolünü sağlayabilmek için açık kaynak VESC yazılımı

3.1.2 RACECAR’ın otonom araca dönüştürülmesi

Bir araç, tek başına insan sürüş eylemlerini gerçekleştiremediği için donanım ve yazılımın birlikte çalıştığı sistemler kullanılır. Buradaki ihtiyaç analizi yapılırken, gerçek dünya’da insan sürüş eylemlerini bir robota nasıl yaptırabiliriz sorusundan yola çıkılmıştır. Bir aracın otonom olarak hareketini sağlamak için farklı yazılım ve donanım

sistemlerinin entegre çalışması gerekmektedir. Bunlar Şekil 3.2'de diyagram şeklinde gösterilmektedir.



Şekil 3.2. Otonom araç bileşenleri

Bu çalışmada, aracın otonom sürüş eylemini gerçekleştirebilmesi için sırasıyla şu işlemler takip edilir:

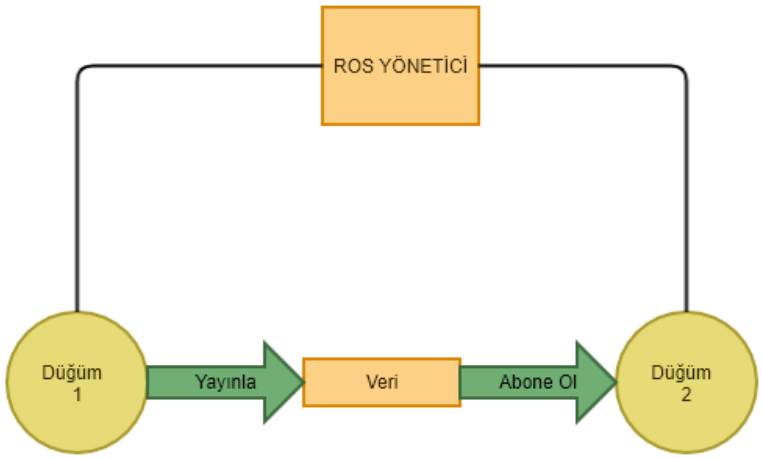
- Araç, otonom sürüş eyleminin gerçekleştirileceği alana getirilir
- Aracın hareketi için Robot İşletim Sistemi üzerinde bir alan oluşturulur ve terminalden veri toplama eylemini gerçekleştiren kod çalıştırılır
- Araç, Joystick yardımıyla ilgili alanda sürülür
- Veriler toplanır
- Veriler dosyaya yazılır
- Derin Öğrenme işlemi için ağ yapısı oluşturulur
- Derin Öğrenme Algoritmaları ile ağ eğitimi gerçekleştirilir
- Eğitim sonucunda model oluşturulur
- Otonom sürüş eylemi için Robot İşletim Sistem üzerinde bir düğüm başlatılır
- Terminalden otonom sürüş eylemini gerçekleştiren kod çalıştırılır

3.1.3 Robot işletim sistemi mimarisi ve kavramları

Robot İşletim Sistemi, robot yazılımı yazmak için geliştirilmiş bir çerçevedir. Araştırma ve ticari amaçlı birçok projede kullanılmaktadır. RİS geliştirilmeden önce birçok robotik uygulama geliştiricisi, kendi robotu için tekrar kullanılmayan algoritmalar ve yazılımlar geliştirmiştir. Geliştiriciler RİS'ten önce, her bir robot için sıfırdan kod yazmak zorunda kalmaktaydı ve bu da çok zaman alan bir işlemdi. Aynı zamanda

geliştiriciler kendi başlarına algoritma geliştirmek zorunda kaldıkları için, robotun geliştirilme sürecinde fazla zaman harcanması anlamına gelmektedir. Günümüzde robotik uygulama geliştiricileri RİS ile birlikte hızlı ve esnek bir şekilde robot yazılımları geliştirebilmektedirler ("ROS" 2019). Özellikle birden fazla uygulama birbiri ile haberleşmek yani veri göndermek isterse RİS'in sağlamış olduğu çerçeveden faydalanmaktadır. Veri gönderme ve alma işlemlerinde soket programlama yerine RİS kullanmak daha uygun bir seçimdir (Joseph 2018).

Şekil 3.3, Düğüm 1 ve Düğüm 2 olarak iki programı gösterir. Bu düğümlerden herhangi biri başladığında, RİS yöneticisi ile iletişim kurar. RİS yöneticisi, düğümlerin gönderdiği veya aldığı bilgilerden haberdardır. Veri gönderen düğümlere yayıncı düğümler, veri alan düğümlere abone düğümler denir. RİS yöneticisi, bilgisayar üzerinde çalıştırılan yayıncı ve abone düğümlerin bilgilerine sahiptir. Düğüm 1, veriyi Düğüm 2'ye göndermek isterse sırasıyla şu işlemler gerçekleştirilir: a) Düğüm 1 yayın yapar, b) Düğüm 2 abone olur, c) RİS Yöneticisi veri göndermek ve almak isteyen düğümlerin birbiriyle iletişim kurmasını sağlar (Joseph 2018).



Şekil 3.3. RİS haberleşme diyagramı

3.1.4 Veri toplama

Sürücüsüz bir şekilde aracın hareket ettirilebilmesi için sürüş yapılacak yolda veri toplanması gereklidir. Veriler, Necmettin Erbakan Üniversitesi Seydişehir Ahmet Cengiz Mühendislik Fakültesi 1. Kat kuzey kısmında kurulan bir platformla toplanmıştır. Verileri toplamak için, Python programlama dili ile bir kod geliştirilmiştir. Geliştirilen bu kod, koridorda sürülen araçtan hız, açı ve 30 görsel / sn veri alıp dosyaya yazmaktadır.

Verilerin toplanıldığı alanda büyük bir dış pencere, farklı açılarda ışıklar ve parlak bir zeminde fayanslardan kaynaklanan yansımalar mevcuttur. Toplanan tüm veriler sonraki aşamalarda uçtan uca öğrenme işlemleri için bir veri tabanı olarak kullanılır.

RİS üzerinde düğümler oluşturmak, hizmet ve mesajları yayınlayabilmek için RİS istemci kütüphanelerine ihtiyacımız vardır. RİS istemci kütüphaneleri, RİS kavramlarını uygulama işlevlerine sahip kod koleksiyonlarından oluşmaktadır. Bu çalışmada, RİS için geliştirilen rospy istemci kütüphanesi kullanılmıştır. Rospy istemci kütüphanesi, Python programcılarının RİS Konuları, Servisler ve Parametreler ile hızlı bir şekilde ara yüzleşmesini sağlar. Rospy'nin tasarımı çalışma süresi performansına göre uygulama hızını (yani geliştirici zamanını) destekler. Böylece algoritmalar hızlı bir şekilde prototip edilebilir ve test edilebilir (“Rospy” 2017). Veri toplamaya başlamadan önce RİS üzerinde veri alınacak tüm çevre birimleri ve düğümlerin başlatılması gereklidir. Aşağıda verilen Python kod parçacığı, rospy ile RİS üzerinde düğümleri başlatmaktadır.

Çizelge 3.1. Rospy ile RİS üzerinde düğüm başlatma kodları

Satır No	Kod
1	`rospy.init_node('collect_data')`
2	`rospy.Subscriber('/zed/left/image_rect_color/compressed',CompressedImage, self.zed_callback)`
3	`rospy.Subscriber('/ackermann_cmd',AckermannDriveStamped, self.drive_call, queue_size=1)`

Bu şekilde bir tanımlama işleminden sonra RİS üzerinde tüm düğümler birbirlerine erişebilir duruma gelirler. ZED ve ackermann düğümleri verileri yayınlar, abone olan collect_data düğümü ise verileri alır. Bir başka ifadeyle, veri toplama yaşam döngüsü içerisindeki en önemli işlem gerçekleştirilmiş olur.

RİS, komut satırları araçlarıyla birlikte, sensor verilerini görselleştirmek için GUI araçlarına sahiptir. Popüler GUI ise Rviz ve Rqt araçlarıdır. Otonom aracın çalışma esnasında düğüm yapılarını, yayıncı ve abone düğümleri ve birbirlerine olan bağlantılarını görselleştirmek mümkündür. RİS hesaplama grafiğini görselleştirmek için rqt_graph GUI eklentisi kurulur (“Rqt_graph” 2018). Bu eklenti sayesinde, araç veri toplarken RİS üzerinde oluşturulan düğümler ve aralarındaki ilişki görselleştirilebilir.

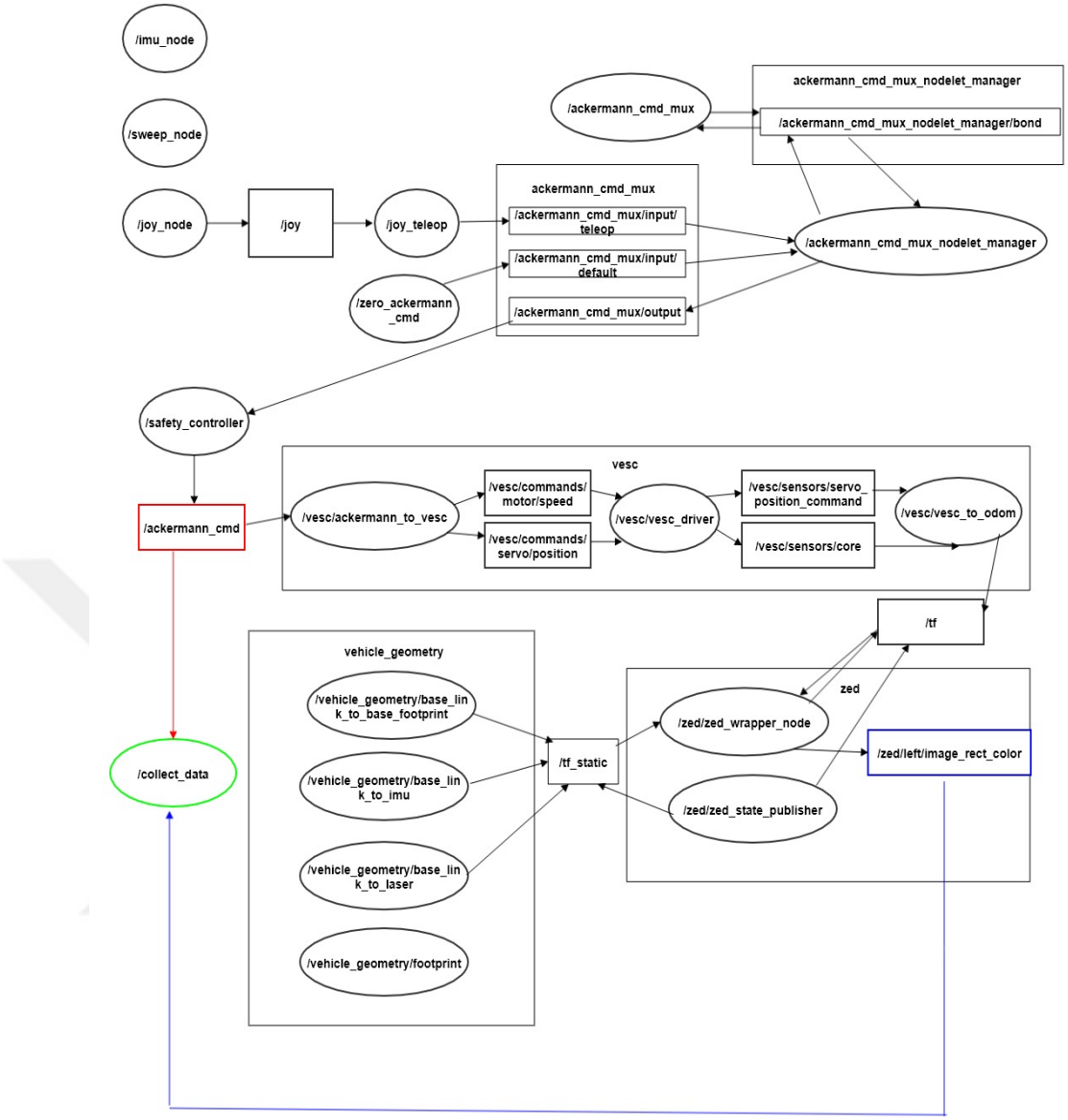
Gerçek zamanlı çalışma anında RİS üzerindeki düğüm yapıları Şekil 3.4'te gösterilmiştir. Terminalden RİS düğümünü çalıştırmak için aşağıda belirtilen kodlar girilir:

```
cd~/calisma_alani
source devel/setup.bash
$ rosrun ros_dosya_adı düğüm_adı
```

RİS üzerindeki düğüm yapılarını göstermek için terminalden aşağıda belirtilen kodlar girilir:

```
$roscore
$ rosrun rqt_gui rqt_gui
```

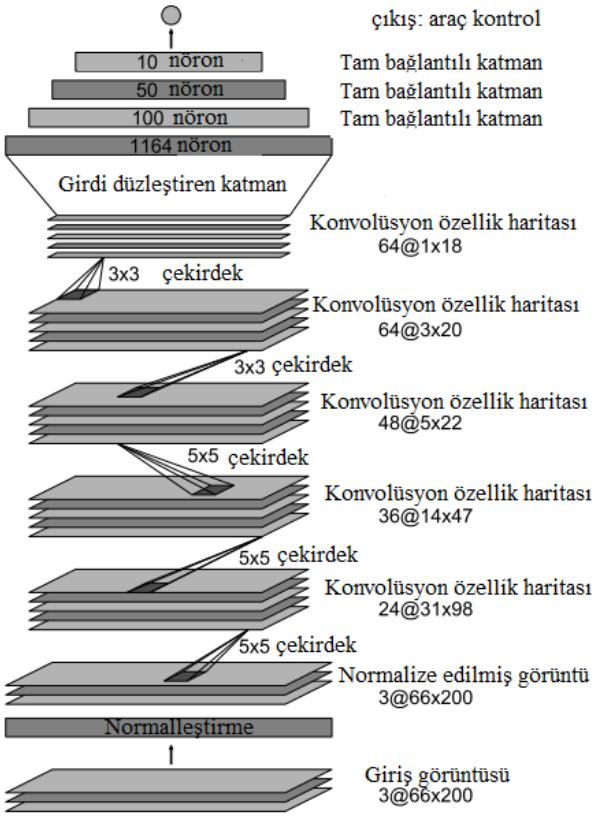
Şekil 3.4, Rqt ile RİS üzerindeki düğüm yapısını açıkça göstermektedir. /joy_node üzerinden sürücünün gerçek zamanlı sürüş eylemleri bir joystick aracılığıyla ackermann_cmd_mux düğümüne iletilmektedir. Aracın gerçek zamanlı hız ve direksiyon açı değerleri, /ackerman_cmd ile /collect_data isimli veri toplama düğümüne iletilmektedir. Aynı zamanda, /ackermann_cmd düğümü üzerinden vesc düğümüne aktarılan veriler aracın gerçek zamanlı bir direksiyon açısı ile hareketini sağlamakta ve pozisyonunu güncellemektedir. Vesc sürücü düğümünden ve vehicle_geometry isimli araç geometrisi hakkında bilgi veren düğümden veriler zed düğümüne iletilmektedir. Zed düğümü ise almış olduğu 2B görsel verileri /collect_data isimli düğüme iletmektedir. Bu şekilde, eş zamanlı araç direksiyon açıları ve direksiyon açılarına karşılık gelen 2B ortam görüntüleri toplanılır. Toplanılan tüm veriler bir dosyaya anlık olarak yazılmaktadır. Ayrıca RİS üzerinde çağrılan servisler, düğümler ve çalıştırılan kod blokları kaydedilerek log olarak bakılabilmektedir.



Şekil 3.4. Gerçek zamanlı veri toplama işleminde, Rqt ile RİS üzerindeki düğümlerin gösterimi

3.1.5 Ağ yapısı

Bu çalışmada, NVIDIA'nın otonom araçlar için geliştirmiş olduğu uçtan uca öğrenme modeli kullanılmıştır (Bojarski et al. 2016). Evrişimsel Sinir Ağı yapısı Şekil 3.5'de gösterilmektedir ("Nvidia Cnn" 2019). Bu ağ yapısı, görüntü normalleştirmesi için Keras Lambda fonksiyonu, üç 5x5 evrişimsel katmanı, iki 3x3 evrişimsel katmanı ve tam bağlantılı katmanlardan oluşmaktadır. Aynı zamanda bu model, RGB'den YUV renk uzayına dönüşümü, 5x5 evrişimsel katmanı üzerinde 2x2 kayma işlemlerini içermektedir.



Şekil 3.5. NVIDIA evrişimsel sinir ağı

Çalışmada kurulan uçtan uca derin öğrenme modeli, Python programlama dili ile Keras kullanılarak Çizelge 3.2'de belirtildiği gibi oluşturulmuştur.

Çizelge 3.2. Python Keras ile uçtan uca öğrenme modeli

Satır No	Kod
1	model = Sequential()
2	model.add(Lambda(lambda x: x / 127.5 - 1.0, input_shape=(66, 200, 3)))
3	model.add(Conv2D(24, (5, 5), activation="relu", strides=(2, 2)))
4	model.add(Conv2D(36, (5, 5), activation="relu", strides=(2, 2)))
5	model.add(Conv2D(48, (5, 5), activation="relu", strides=(2, 2)))
6	model.add(Conv2D(64, (3, 3), activation="relu"))
7	model.add(Conv2D(64, (3, 3), activation="relu"))
8	model.add(Flatten())

```
9   model.add(Dense(100))
10  model.add(Dense(50))
11  model.add(Dense(10))
12  model.add(Dense(1))
13  model.compile(loss='mse', optimizer='adam')
```

Evrişimsel sinir ağı modeline 533x312 boyutunda “png” tipinde bir görüntü giriş parametresi olarak girilmektedir. Eğitim işlemi bitine kadar bu işlem 15690 kez tekrarlatılmaktadır. Eğitim işleminin tamamlanması ile birlikte bir model elde edilmektedir. Daha sonra, otonom sürüş işleminin gerçekleştirilebilmesi için ağdaki tüm parametreler ve ağırlıklar “json” uzantılı bir dosyaya kaydedilmektedir. Otonom sürüş işlemi esnasında zed kameradan alınan her bir görüntü model üzerinde sorgulatılmaktadır. Model daha önce öğrenmiş olduğu veri seti üzerinden bir direksiyon açı değeri tahmini gerçekleştirmektedir. Ağ tek bir çıkış değeri tahmin eden bir regresyon ağıdır.

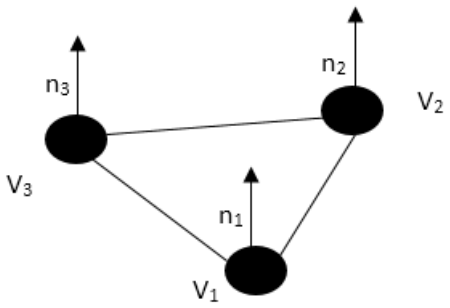
3.2 Mobil Artırılmış Gerçeklik Uygulaması Mimarisi ve Geliştirilmesi

Android cihazlar için, ASA ve AG teknolojisi kullanılarak mobil uygulama geliştirilmiştir. Geliştirilen bu uygulama, gerçek zamanlı otonom bir araç için takip ve izleme alt yapısına sahiptir. Bu tez çalışmasında, otonom araçlar için geliştirilen mobil AG uygulaması şunları içermektedir:

- Gerçek zamanlı nesne tanıma
- Anlık araç hız değeri değişimi
- Eş zamanlı enerji tüketimi gösterimi
- Gerçek zamanlı direksiyon açısı değişimi

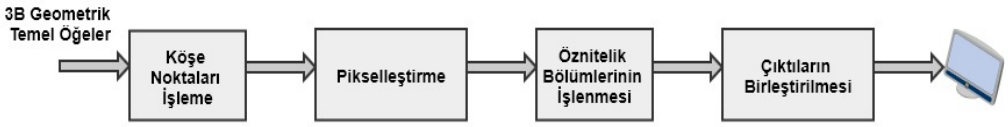
3.2.1 Artırılmış gerçeklik için bilgisayar grafikleri temelleri

Bilgisayar grafikleri, AG uygulamalarının geliştirilmesinde ele alınması gereken bir alandır. Bilgisayar grafikleri en temel düzeyde, 3B statik bir sahne ortamında görsel bir dönüşümü ifade eder. 3B statik bir sahne, geometri dönüşümleri, ışıklar ve malzeme özelliklerinden oluşmaktadır. Bilgisayar grafiklerinde kullanılan geometri ilkeleri şunlardır: köşe noktası, kenar ve üçgen’dir. Şekil 3.6, grafiksel bir obje için geometrik temel ilkeleri ifade etmektedir.



Şekil 3.6. Geometrik grafik gösterimi (V: Vertex, n: Normal)

Köşe noktaları, 3B olarak ifade edelirse, V(x, y, z) şeklinde genel bir tanımlama yapılır. Her köşe için yüzey yönünü ifade eden normaller ise, n=(n_x, n_y, n_z) şeklinde tanımlanır. Şekil 3.7, genel olarak 3B bir grafik dönüşüm boru hattını ifade etmektedir.



Şekil 3.7. 3B grafik dönüşüm boru hattı

Şekil 3.7, aşağıdaki işlem adımlarını ifade etmektedir:

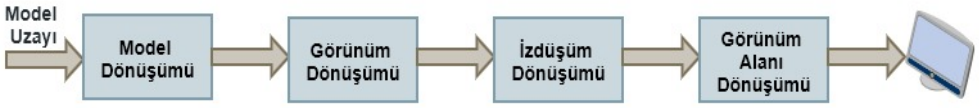
Köşe Noktaları İşleme: Köşe noktaları ve normalleri işleme / dönüştürülmesi işlemi gerçekleştirilir.

Pikselleştirme: Her bir veri (köşelerden birbirine bağlı), 3B uzayda piksel olarak ele alınır. Bu işlem adımından sonra renk, pozisyon, normal ve doku gibi özellikler piksel ızgaraları ile ifade edilir.

Öznitelik Bölümlerinin İşlenmesi: Bireysel olarak her bir parça işlenir.

Çıktıların Birleştirilmesi: 3B alan verilerinin her bir parçasının birleştirilerek, görüntülemek için 2B renk pikseline dönüştürülür.

Grafik boru hattı dönüşüm işlemlerinde, noktasal haritaları bir koordinat sisteminden diğerine dönüştürürken kullanılan başlıca koordinat sistemleri vardır. Bu koordinat sistemleri sırasıyla: Sağ el koordinat sistemi, nesne koordinatları, dünya koordinatları, görüntüleme koordinatları ve pencere koordinatları şeklinde ifade edilir. 3B grafiksel bir nesnenin, koordinat dönüşüm hattı, Şekil 3.8'de gösterilmektedir (Wetzstein 2018).



Şekil 3.8. Koordinat dönüşüm hattı

Model dönüşümü, Görünüm dönüşümü ve İzdüşüm dönüşümü adımları Şekil 3.7'de verilen köşe noktaları işleme adımına karşılık gelmektedir. Görünüm alanı dönüşümü ise Şekil 3.7'de verilen pikselleştirme işlem adımına karşılık gelmektedir.

Model Dönüşümü: Dünya üzerinde var olan nesnelerin veya modellerin dönüşüm adımlarını ifade eder.

Görünüm Dönüşümü: Kameranın, konumlandırılması ve yönlendirilmesi gibi işlem adımlarının dönüşümünün gerçekleştirildiği işlem adımıdır.

İzdüşüm Dönüşümü: Geniş açılı, teleskopik veya normal bir kamera lensi seçim işlemlerini kapsar. Aynı zamanda, kameranın görüş alanını ayarlamak için odak uzaklığını etkileyen faktörlerin ayarlanmasını ifade eder.

Görünüm Alanı Dönüşümü: Görüntüleme alanının boyutunu ve şeklini tanımlar.

Her köşeyi ifade eden noktaları, nesne koordinatlarından dünya koordinat düzlemine dönüştürülürse, 3.1 denklemi ile model dönüşümü elde edilir:

$$v = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3.1}$$

3x3 tipinde ölçeklendirme matrisi kullanılarak:

$$S(S_x, S_y, S_z) = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{pmatrix} \tag{3.2}$$

3.3 Denklemi ile ölçeklendirilmiş köşe noktalarını ifade eden matris dönüşümü yazılır.

$$S_v = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} S_x x \\ S_y y \\ S_z z \end{pmatrix} \tag{3.3}$$

$R_x(\theta), R_y(\theta), R_z(\theta)$ 3x3 tipinde döndürme matrisleri olup model dönüşüm matrisi ile çarpılarak döndürülmüş köşe noktalarına karşılık gelen matrisler denklem 3.4'de hesaplanır.

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix},$$
$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}, \quad (3.4)$$
$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Z için Döndürülmüş köşe noktalar matrisi ise denklem 3.5'deki gibidir.

$$R_z v = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \\ z \end{pmatrix} \quad (3.5)$$

Dönme ve ölçekleme matris dönüşümleri için 3 sütunlu matrislere ihtiyaç vardır.

Fakat, Model dönüşüm vektörünün 3x3 tipinde bir matris ile çarpılarak çevirisini (translation) temsil etmek mümkün değildir. Bu yüzden, homojen koordinat sistemi kullanılır. Homojen koordinat sisteminde matris çevirme işlemleri 4x4 matrisler kullanılarak gerçekleştirilir.

Homojen koordinat sistemi kullanılarak, model dönüşüm matrisi 3.6 denklemindeki gibi olur.

$$v = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (3.6)$$

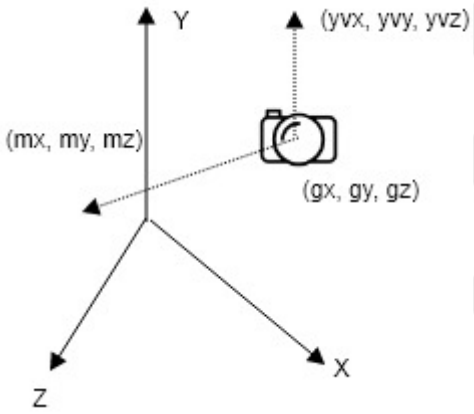
Matris çeviri işlemi için öncelikle 4x4 tipinde bir matris olarak denklem 3.7'deki gibi tanımlanır:

$$T(d) = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.7)$$

Tüm homojen koordinat sistemi tanımlamalarından sonra, çeviri işlemi denklem 3.8'deki gibi olur.

$$Tv = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + d_x \\ y + d_y \\ z + d_z \\ 1 \end{pmatrix} \tag{3.8}$$

Görünüm dönüşüm işleminde, dünya koordinat düzleminde yer alan bir kameradan, 4x4 tipinde matris ile kamera veya görünüm dönüşümü ifade edilmesi gerekirse, Şekil 3.9' daki gibi bir gösterim kullanılır. Şekil 3.9, görünüm dönüşümünü ifade etmektedir.



Şekil 3.9. Koordinat dönüşüm hattında yer alan görünüm dönüşümü

Kamera, göz pozisyonuna göre denklem 3.9'deki gibi belirtilirse:

$$\text{göz pozisyonu} = \begin{pmatrix} g_x \\ g_y \\ g_z \end{pmatrix} \tag{3.9}$$

Kamera, referans noktaya göre denklem 3.10'daki gibi belirtilirse:

$$\text{merkez} = \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix} \tag{3.10}$$

Kameranın yukarı yönlü hareketi bir vektör olarak tanımlanırsa denklem 3.11 elde edilir:

$$\text{yukarı vektör} = \begin{pmatrix} yv_x \\ yv_y \\ yv_z \end{pmatrix} \tag{3.11}$$

Tüm tanımlama işlemlerinden sonra, denklem 3.12'deki 3 vektör hesaplanmaktadır:

$$Z^c = \frac{göz\ pozisyonu - merkez}{||göz\ pozisyonu - merkez||},\ X^c = \frac{yukarı\ vektör\ x\ Z^c}{||yukarı\ vektör\ x\ Z^c||},\ Y^c = Z^c\ x\ X^c \tag{3.12}$$

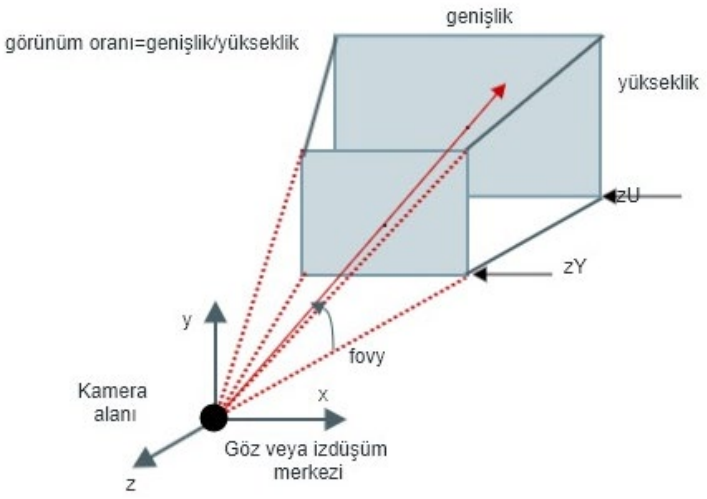
Tüm bu tanımlama işlemleri ardından M olarak adlandırılan, görünüm dönüşüm matrisi ise denklem 3.13'de verilmiştir.

$$M = R \cdot T\ (-e) = \begin{pmatrix} x_x^c & x_y^c & x_z^c & 0 \\ y_x^c & y_y^c & y_z^c & 0 \\ z_x^c & z_y^c & z_z^c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -g_x \\ 0 & 1 & 0 & -g_y \\ 0 & 0 & 1 & -g_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.13}$$

$$= \begin{pmatrix} x_x^c & x_y^c & x_z^c & -(x_x^c g_x + x_y^c g_y + x_z^c g_z) \\ y_x^c & y_y^c & y_z^c & -(y_x^c g_x + y_y^c g_y + y_z^c g_z) \\ z_x^c & z_y^c & z_z^c & -(z_x^c g_x + z_y^c g_y + z_z^c g_z) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Model görünüm matrisi, döndürme, ölçeklendirme ve çeviri işlemlerini içeren birleştirilmiş modeldir. Aynı zamanda, görünüm matrisidir.

İzdüşüm dönüşümü ise, 2B bir kanvas üzerine 3B nokta yansıtan 4x4 tipinde bir matris içerir. Kamera yerleştirilip yönlendirildikten sonra, ne görebileceğine ve nesnelerin ekrana nasıl yansıtıldığına karar vermemiz gerekir. Bu, bir izdüşüm modu seçerek (perspektif veya ortografik) ve bir görüntüleme hacmini veya kırpma hacmini belirterek yapılır. Kırpma hacminin dışındaki nesneler sahneden çıkarılır ve görülemez. Kamera, kesik piramit şeklinde görünüm sergileyen ve dört parametre ile belirtilen sınırlı görüş alanına sahiptir. Bu parametreler, fovy, Aspect, zNear ve zFar şeklindedir. Şekil 3.10, kameranın kesik piramit görünümünü 4 parametre ile ifade etmektedir ("3D Graphics with OpenGL" 2012).



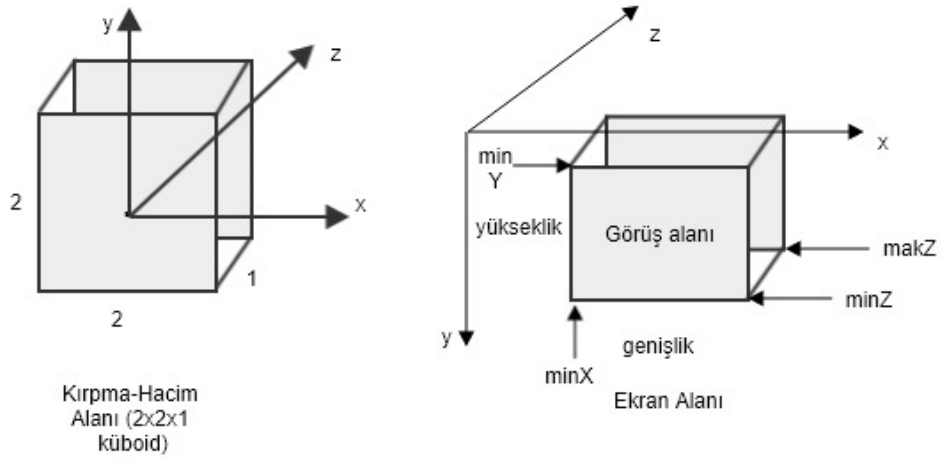
Şekil 3.10. Kameranın kesik piramit görünümlü görüş alanı

Şekil 3.10'da fovy: toplam dikey görüş açısını belirtir, görünüm oranı ise genişliğin yüksekliğe oranıdır, zU yakın düzlemi ve zY ise uzak düzlemi ifade etmektedir.

$M_{izd\text{üşüm}}$ Matrisi denklem 3.14'de verilmiştir:

$$M_{izd\text{üşüm}}=\begin{bmatrix} \frac{\cot\left(\frac{fovy}{2}\right)}{g\ddot{o}r\ddot{u}n\ddot{u}m\ oranı} & 0 & 0 & 0 \\ 0 & \cot(fovy/2) & 0 & 0 \\ 0 & 0 & \frac{zU}{zU-zY} & \frac{zY\ x\ zU}{zU-zY} \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad (3.14)$$

Son dönüşüm olan, Görünüm alanı dönüşümü işlemi, uygulama penceresindeki, ekranın koordinatlarında (sol üst köşede orijinli olarak piksel cinsinden ölçülen) dikdörtgen bir görüntüleme alanıdır. Görünüm alanını, uygulama penceresinin üzerine kamera tarafından yakalanan sahneyle eşleştirmek için görüntüleme alanının boyutunu ve şeklini tanımlar. Tüm ekranı kaplayabilir veya kaplamayabilir. 3B grafiklerde, üst üste binen pencerelerin sıralanması gibi durumlar için gerekli olan z sıralamasını desteklemek için bir görünüm alanı 3 boyutludur. 3B Görünüm alanı dönüşüm işlemi, kırpma-hacmiyle (2x2x1 küboid) ifade edilir. Şekil 3.11, görünüm alan dönüşümünü göstermektedir ("3D Graphics with OpenGL" 2012).



Şekil 3.11. Görünüm alan dönüşümü

Görüş Alanı matrisini tanımlamak için sırasıyla: ilk olarak y eksenin yansıma matrisi hesaplanır. İkinci adımda, x, y, z eksenlerinin ölçeklendirilmesi yapılır. Üçüncü adımda, orijinin çevirisi bulunur.

Y ekseninin yansıması denklem 3.15'de, x, y, z'nin ölçeklendirilmesi denklem 3.16'da ve orijin çevirisi denklem 3.17'de verilmiştir:

$$My = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

x, y, z'nin ölçeklendirilmesi:

$$Mö = \begin{bmatrix} g/2 & 0 & 0 & 0 \\ 0 & y/2 & 0 & 0 \\ 0 & 0 & makZ - minZ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

Orijin Çevirisi:

$$Mç = \begin{bmatrix} 1 & 0 & 0 & minX + g/2 \\ 0 & 1 & 0 & minY + y/2 \\ 0 & 0 & 1 & minZ \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

Görünüm Alanı Matrisi Denklem 3.18'deki gibi elde edilir:

$$M_{\text{görüşalanı}} = MçMöMy = \begin{bmatrix} g/2 & 0 & 0 & \min X + g/2 \\ 0 & -y/2 & 0 & \min Y + y/2 \\ 0 & 0 & makZ - \min Z & \min Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.18)$$

Görünüm alanı, tüm ekranı kaplıyorsa minX, minY ve minZ değerleri 0'a eşittir. Aynı zamanda, Şekil 11'de sağ kısımda ifade edilen ekran alanı dönüşümünde genişlik değeri ekranın genişliğine ve yükseklik değeri ekranın yükseklik değerine eşittir.

3.2.2 Artırılmış gerçeklik izleme ve takip sistemi mimarisi

1997 yılında Azuma AR tanımını 3 temel kriteri gerçekleştiren bir sistem olarak tanımlamıştır (T.Azuma 1997):

1. Gerçek ve sanal birleşimi,
2. Gerçek zamanlı etkileşim,
3. 3B gerçek ve sanal objelerin birlikte olduğu sistem

Günümüzde AG ise, bilgisayar ortamında oluşturulan sanal görüntülerin gerçek dünya içerisinde yerleştirilip kullanılmasına izin veren teknoloji olarak tanımlanmıştır.

AG teknolojisi, SG'den farklıdır. Sanal Gerçeklik, sanal olarak bilgisayar ortamında hazırlanan Dünya'nın içinde yaşanmasını beklemektedir. AG ise, gerçek nesneler üzerine tasarlanmış grafiksel objelerin yerleştirilmesi ile oluşturulan bir teknolojidir. Bir başka ifadeyle, AG gerçek ve sanal yaşam arasındaki bağlantıyı sağlayan bir teknolojidir.

AG uygulamalarının kullanılabilmesi için tablet, akıllı telefonlar ve bilgisayarlara ihtiyaç duyulmaktadır. AG uygulamalarının artması ile birlikte mobil cihazlar için geliştirilen uygulama sayısı gün geçtikçe artmaktadır.

AG teknolojisi, dijital verilerin gerçek dünya ortamına eklenmesini ve etkileşim kurulabilmesi için ortam oluşturur. Dijital verileri kaydetmek, izlemek ve var olan medya içerikleri ile işlemler yapabilmek için işaretçi-tabanlı platformlar(örneğin, Vuforia, ARToolkit, ARTag) kullanılmaktadır. Geçtiğimiz 15 yıl boyunca AG teknolojisinde İşaretçi-Tabanlı yapılan işlemlerden İşaretsiz dünyaya geçiş görülmüştür. Günümüzde mobil cihazlarda içeriklerin dinamikleştirilmesi ve kullanıcıların kendi AG dünyalarını

geliştirebilmeleri için birçok AG tarayıcısı (örneğin, Wikitube, Junaio ve Layar) bulunmaktadır.

Kullanıcıların yapmış oldukları eylemlere göre grafiksel objelerin en uygun yere göre konumlandırma işlemi AR uygulamaları için oldukça önemlidir. Günümüzde AG uygulamalarını kullanım alanlarına göre iki gruba ayrılabilir: 1) İşaretçi-Tabanlı teknoloji, 2) İşaretsiz Teknoloji

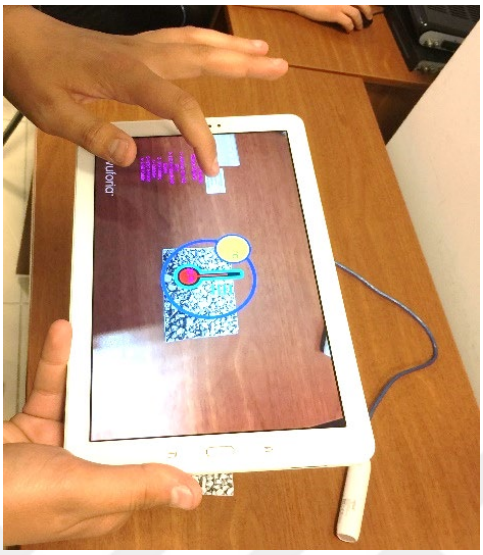
İlk olarak otonom bir araç için, Mobil AG uygulaması, geliştirilmeden önce gerçek zamanlı elektronik bir devre ile nasıl haberleşebilir sorusundan yola çıkılarak bir uygulama geliştirilmiştir. İlk etapta geliştirilen Mobil AG uygulaması, iki farklı AG alt yapısını destekleyecek şekilde hibrit bir yazılım olarak geliştirilmiştir. Geliştirilen bu uygulama ile İşaretçi-Tabanlı ve İşaretsiz olmak üzere iki farklı uygulama alt yapısı test edilmiştir. Bu uygulama, Android mobil uygulama geliştirme ortamında yazılan Bluetooth alıcı (receiver) kod alt yapısı ile elektronik devre üzerinde yer alan Bluetooth modülünden ortam verilerini toplamaktadır. Bu işlem adımları, aşağıda açıklanan AG sistemleri içerisinde gösterilmektedir. Tüm bu işlem ve test adımlarından sonra, otonom bir araç için, en uygun haberleşme teknolojisi, mobil uygulama ekran dizaynı ve yöntemi belirlenmiştir.

İşaretçi-Tabanlı (İT) AG Sistemi: İT tabanlı izleme, takip edilecek nesneye fiziksel olarak işaretleyicilerin eklenmesi ile elde edilir. İşaretçiler kullanılacağı alana göre uygun bir dizayna sahip olmalıdır. Gerçek nesneler üzerine konumlandırılan işaretçiler daha önceden sisteme tanıtılmıştır. İşaretçilerin hızlı ve kolay algılanabilmesi için gerçek nesneler üzerindeki konumu daha önceden belirlenmelidir. Düzgün bir yerleştirme açısına sahip olan işaretçiler kamera tarafından kolaylıkla algılanabilmektedir. Bu şekilde, İT izleme işlemi işaretçinin görünürlüğüne bağlıdır. İT izleme sisteminin bazen kullanılabilirliği mümkün olmamaktadır. Endüstriyel ortamda işaretçinin takip edilebilirliğini etkileyecek birçok nesne olabilir (makineler, cihazlar) (Palmarini et al. 2018). Bu durum, AG sisteminde izleme problemine neden olmaktadır.

Bunlara ek olarak, İT bir sistemin sorunsuz çalışabilmesi için işaretçilerin titizlikle korunması gerekmektedir. Bu sebeplerden dolayı, takip edilebilirliği uygun olmayan alanlarda kullanımı uygun değildir (Zhu, Ong, and Nee 2013).

İT AG sistemi için geliştirilen, ön hazırlık mobil AG uygulaması aşağıda gösterilmektedir. Bu uygulama, Arduino ile hazırlanmış elektronik bir devre üzerinden ısı, ışık ve nem gibi ortam verilerini alarak ekranda göstermektedir. 2B ve 3B grafiksel objelerin ekranda gösterilebilmesindeki temel şart ise işaretçi olarak daha önceden

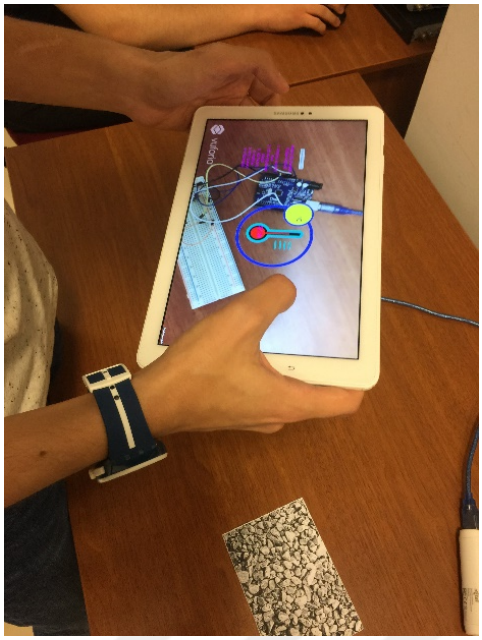
belirlenmiş bir görsele cihaz kamerasının uygun açıdan tutulması gerekliliğidir. Şekil 3.12, İT gerçek zamanlı elektronik devre ile haberleşme alt yapısına sahip bir uygulamanın test adımıdır.



Şekil 3.12. İşaretçi tabanlı mobil artırılmış gerçeklik uygulamasının elektronik bir devre ile gerçek zamanlı iletişimi

İşaretsiz (İZ) AG Sistemi: İZ yöntemi, ortam üzerinde ek bir yerleştirme veya işaretçiye ihtiyaç olmadan nesne tanıma işlemini gerçekleştiren bir yöntemdir. Nesne tanıma işlemi, modellerin veya 3B bir nesneyi temsil eden noktalar kümesinden çıkarım yaparak gerçekleştirilmektedir (Paulo Lima et al. 2017). İZ tanıma işlemi, kullanıcılarda minimum süreç hazırlığını sağlamaktadır. İZ obje tanıma işlemi İT obje tanıma yöntemine göre daha karmaşık işlemler gerektirmektedir. Bu yöntemin, İT yöntemden daha iyi olduğu kanıtlanamamıştır (Khandelwal et al. 2015).

İşaretsiz AG sistemi için geliştirilen, ön hazırlık mobil AG uygulaması Şekil 3.13'de gösterilmektedir. Bu uygulama, Arduino ile hazırlanmış elektronik bir devre üzerinden ısı, ışık ve nem gibi ortam verilerini alarak ekranda göstermektedir. Elektronik devre, 3B tarama işlemi ile taranıp sisteme tanıtılmıştır. 2B ve 3B grafiksel objelerin ekranda gösterilebilmesindeki temel şart ise işaretçi olarak daha önceden belirlenmiş bir nesneye (elektronik devrenin kendisi) cihaz kamerasının uygun açıdan tutulması gerekliliğidir. Şekil 3.13, İşaretsiz gerçek zamanlı elektronik devre ile haberleşme alt yapısına sahip bir uygulamanın test adımı görselidir.



Şekil 3.13. İşaretsiz mobil artırılmış gerçeklik uygulamasının elektronik bir devre ile gerçek zamanlı iletişimi

3.2.3 Android sinir ağları ve tensorflow lite ile nesne tanıma

Android Yapay Sinir Ağları Uygulama Programlama Arayüzü (UPA), mobil cihazlarda makine öğrenmesi için yoğun hesaplama gerektiren işlemleri yürütmek için tasarlanmış bir Android C UPA'dır. Android Yapay Sinir Ağları Uygulama Programlama Arayüzü, sinir ağları oluşturabilen ve eğiten yüksek seviyeli makine öğrenme çerçeveleri (TensorFlow Lite, Caffee2 vd.) için işlev katmanı sağlamak üzere tasarlanmıştır. UPA, Android 8.1 (UPA Seviyesi 27) sürümü olan veya daha yüksek olan tüm cihazlarda kullanılabilir. Android Yapay Sinir Ağları UPA ile daha önce eğitilmiş ve cihaz içine eklenmiş modellerden çıkarım yapabilmeyi destekler. Görüntü sınıflandırma, kullanıcı davranışını tahmin etme, bir arama sorgusuna en uygun yanıt seçimi gibi işlemler kullanılan çıkarım örnekleri arasındadır. Tüm bu işlemlerin cihaz içerisinde gerçekleştirilmesinde sağladığı avantajlar ise şöyledir ("Android Neural Networks API" 2019):

Gecikme: Bir ağ üzerinden sunucuya istek göndermeye ve cevap beklemeye gerek yoktur. Böylece, kameradan gelen art arda isteklere en hızlı şekilde cevap verilir.

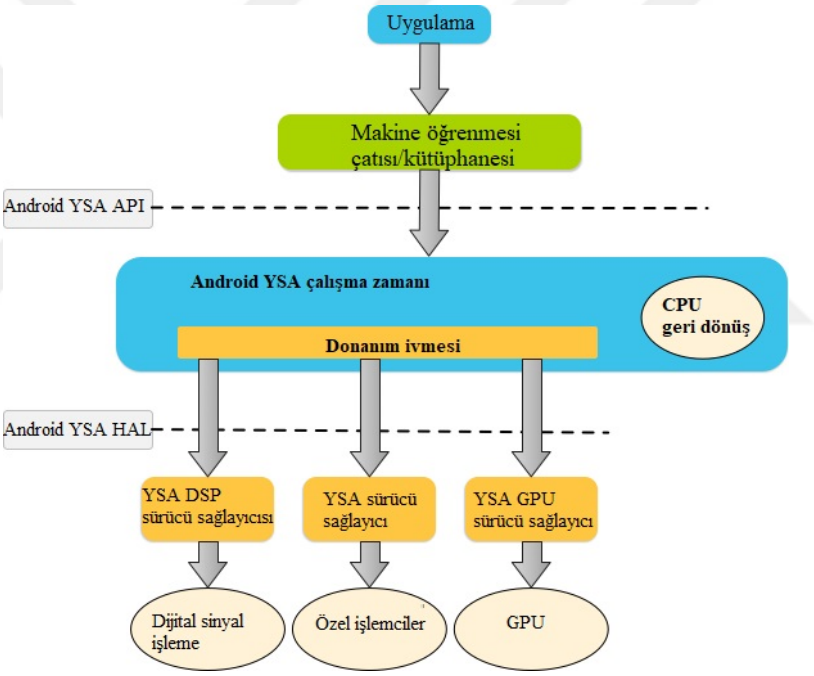
Kullanılabilirlik: Geliştirilen uygulama herhangi bir kapsama alanına ihtiyaç duyulmadan çalışabilmektedir.

Hız: Yapay sinir ağları için geliştirilen cihaz içi donanım alt yapısı sayesinde, tek CPU'nun gerçekleştirdiği işlemlerden daha hızlı çalışmaktadır.

Gizlilik: Verilerin cihaz içindeki uygulamaya gömülü olması güvenliği artırmaktadır.

Maliyet: Tüm işlemler cihaz içerisinde gerçekleştirildiği için sunucu gibi maliyeti yüksek cihazlara ihtiyaç bulunmamaktadır.

Uygulamalar, ASA'ı UPA'yı doğrudan kullanmazlar, onun yerine üst seviye makine öğrenme çerçevelerini (frameworks) kullanır. Bu çerçeveler, ASA'ı destekleyen cihazlarda, donanım temelli işleri hızlandırmak için kullanılırlar. Android sinir ağlarının çalışma zamanı, cihazdaki donanım özelliklerine ve uygulama gereksinimlerine, grafik işlem birimleri (GPUs) ve sinyal işlemcilerine bağlı olarak mevcut iş yükü en uygun şekilde dağıtılarak belirlenir. Şekil 3.14, Android sinir ağları mimarisini göstermektedir ("Android Neural Networks API" 2019).



Şekil 3.14. Android yapay sinir ağları uygulama programlama arayüzü

ASA programlama modeli dört soyut adımdan oluşur ("Android Neural Networks API" 2019):

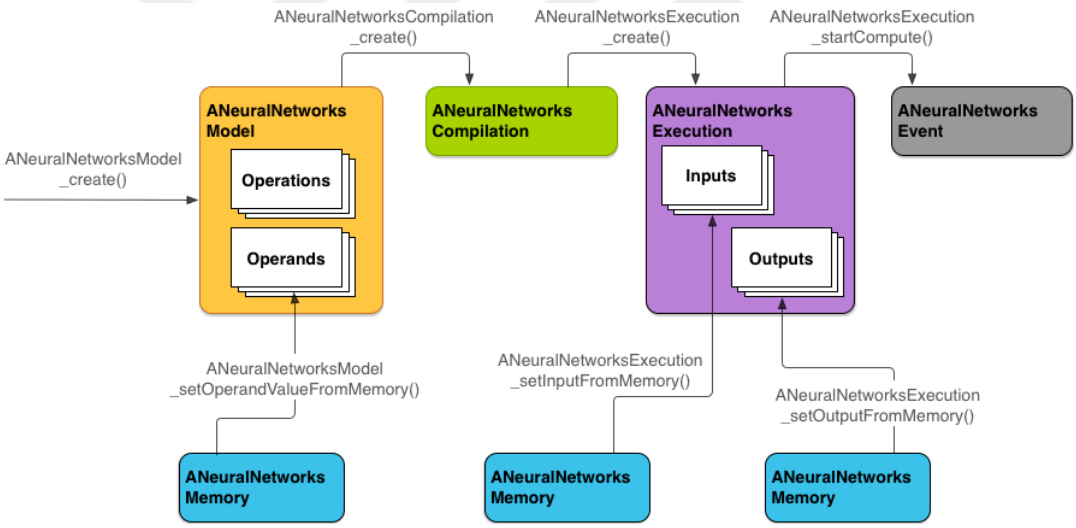
Model: Sinir ağları, matematiksel işlemlerin hesaplanan grafikleri ve eğitim süreci boyunca öğrenilen sabit değerlerden oluşur. Aynı zamanda sinir ağları, 2B konvolüsyon, sigmoid ve ReLU aktivasyon fonksiyonlarını içerir. Modeli oluşturan matematiksel işlemlerin açıklaması ANeuralNetworksModel ile ifade edilir.

Derleme: Oluşturulan bir modeli daha düşük seviyeli bir kodda derlemek için gerekli olan konfigürasyonu temsil eder. Her derleme işlemi "ANeuralNetworksCompilation" olarak temsil edilir.

Bellek: Bir uygulama genel olarak bir modeli tanımlamak için paylaşılan tampon bir hafıza oluşturur. Böylece, Android sinir ağlarının verileri sürücülere daha verimli bir şekilde aktarılması sağlanır. Bellek tamponu ise "ANeuralNetworksMemory" ile temsil edilir.

Yürütme: Bir Android sinir ağı, bir grup girişe uygulanarak sonuçların elde edildiği arayüzden oluşur. Yürütme, eş zamanlı olmayan bir işlemdir. Aynı yürütme işleminde birden fazla iş parçacığı bekleyebilir. Yürütme işlemi tamamlandığında, tüm iş parçacıkları serbest bırakılır. Her bir yürütme, "ANeuralNetworksExecution" olarak gösterilir.

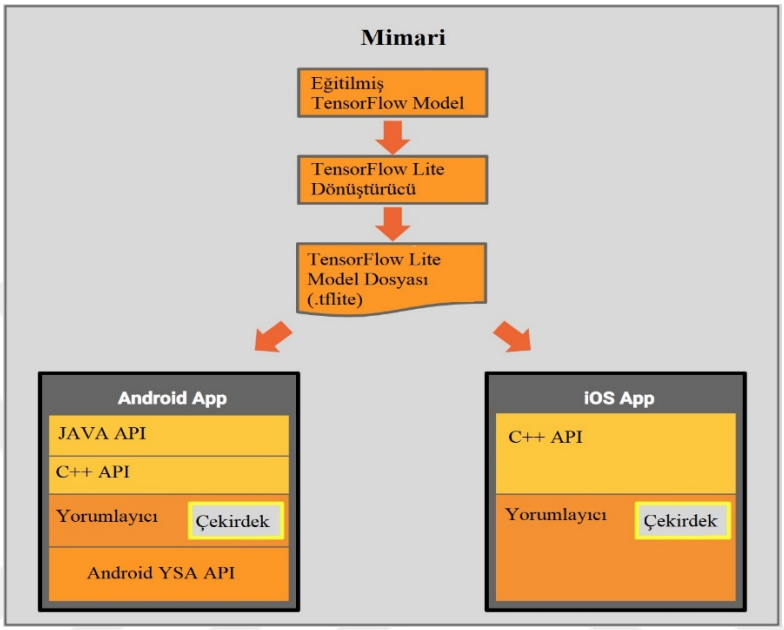
Şekil 3.15, temel Android sinir ağları programlama akışını göstermektedir:



Şekil 3.15. Android yapay sinir ağları uygulama programlama arayüzü akışı

Akıllı telefonlar, makine öğrenme görevleri için cihazdaki çıkarım işlemlerinden faydalanabilir. Genel olarak mobil cihazların bulut ortamında bulunan makine öğrenmesi modellerini kullanması yaygındır. Bunun sonucunda, bulut hizmeti maliyetlerinin yanı sıra, gecikme ve uygulama kullanılabilirliği sorunları oluşur. TensorFlow Lite ise mobil cihaz içerisinde çıkarım işlemlerinin gerçekleştirilmesini sağlar. TensorFlow Mobil'in bir sonraki gelişmiş hali TensorFlow Lite'ın, desteklenen cihazlarda donanım performansını yükselteceği belirtilmektedir. TensorFlow Lite, gömülü ve mobil cihazlar için TensorFlow'un hafif çözümüdür. Aynı zamanda, küçük ikili bir boyut ve düşük gecikme süresi ile mobil cihazda makine çıkarımını sağlar. TensorFlow Lite, Android ve iOS

platformlarını destekler. Şekil 3.16, TensorFlow Lite'ın desteklediği mobil platformları ifade etmektedir. İlk adım, TensorFlow Lite dönüştürücü (converter) kullanarak eğitimli bir TensorFlow modelinin, TensorFlow Lite dosya formatına (.tflite) dönüştürülmesini içerir. Dönüştürülen model dosyası uygulama içerisinde kullanılır (Arun Mani Sam 2019).



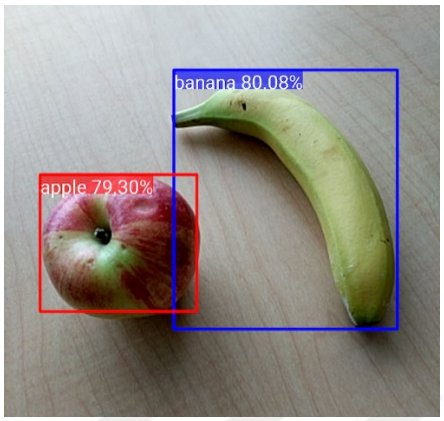
Şekil 3.16. TensorFlow Lite tarafından desteklenen mobil platformlar

Bu çalışmada, TensorFlow Lite Android uygulama içerisine eklenerek kullanılmıştır. TensorFlow Lite, doğrudan ilişkili olduğu Android Sinir Ağları UPA'sı ile donanımın hızlanmasını destekler.

Bu tez kapsamında geliştirilen; otonom Android mobil AG uygulamasına, daha önceden eğitilmiş COCO SSD MobileNet V1 modeli, uygulama içerisine eklenerek kullanılmıştır. Sağlanan bu model ile aynı anda bir görüntüde 10 nesne tanınabilir ve konumu tespit edilir. Bu model, 80 nesne sınıfını tanımak için eğitilmiştir ("TensorFlow Lite Object Detection" 2019).

Bir görüntü veya video akışı göz önüne alındığında, bir nesne algılama modeli, bilinen bir nesne grubunun hangisinin mevcut olabileceğini belirleyebilir ve görüntü içindeki konumları hakkında bilgi sağlayabilir. COCO SSD MobileNet V1 nesne algılama modeli, temsil ettiği sınıfın etiketleri ve her nesnenin nerede göründüğünü belirten birden fazla nesne sınıfının varlığını ve konumunu tespit etmek için eğitilmiştir. Daha sonra bu modele bir görüntü verildiğinde, algıladığı nesnelerin bir listesi, her

nesneyi içeren bir sınırlama kutusunun konumu ve algılamanın doğru olduğuna güvendiğini gösteren bir skor çıkartılır. Şekil 3.17, nesne tanıma işleminde bir örneği temsil etmektedir (“TensorFlow Lite Object Detection” 2019).



Şekil 3.17. TensorFlow Lite ile nesne tanıma işlemi

Model, bir görüntüyü giriş olarak alır ve görüntü, piksel başına üç kanal (kırmızı, mavi ve yeşil) olan 300x300 pikseldir. Model, 270.000 bayt değerinde (300x300x3) düzleştirilmiş bir tampon olarak veri ile beslenir. Tüm bu işlemlerin ardından tespit sonucu çıkarılır. Tespit sonucu nesnenin sınıfı, skor ve konum bilgisinden oluşur ve Çizelge 3.3’de gösterilmektedir (“TensorFlow Lite Object Detection” 2019).

Çizelge 3.3. Nesne tespiti dönüş değerleri

Sınıf	Skor	Konum
elma	0.92	[18, 21, 57, 63]
Muz	0,88	[100, 30, 180, 150]
çilek	0.87	[7, 82, 89, 163]
~~armut~~	~~0.23~~	~~[42, 66, 57, 83]~~
~~portakal~~	~~0.11~~	~~[6, 42, 31, 58]~~

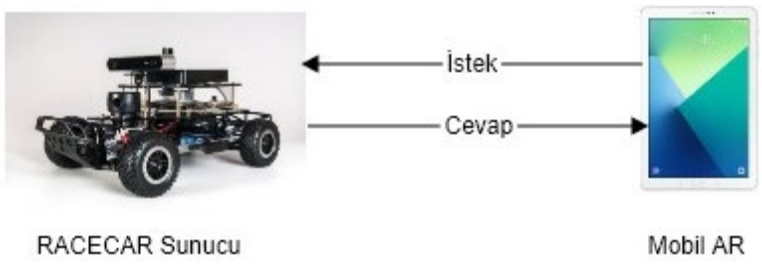
Skor, 0 ile 1 arasında, nesnenin gerçekten tespit edildiğine güvendiğini gösteren bir sayıdır. Tespit edilen skor ve konum sonuçlarına göre bir kesme eşiğine karar verilir. Çizelge 3.3’de yer alan verilere göre eşik değeri skor alanında 0.5 olarak belirlenirse, dizideki son iki eleman yok sayılır. Çünkü son iki elemanın güvenilirliğini belirleyen skor değeri 0.5’in altındadır. Şekil 3.17 ‘de gösterildiği gibi algılanan nesne için model, nesnenin konumunu çevreleyen sınırlayıcı dikdörtgeni temsil eden konum sayı dizisini döndürecektir. Konum verisi ise: [üst, sol, alt, sağ] şeklinde sıralanır. Üst değer,

dikdörtgenin üst kenarının görüntünün üstünden piksel cinsinden olan mesafesini gösterir. Sol değer, sol kenarın giriş görüntüsünün solundan uzaklığını temsil eder. Diğer değerler, alt ve sağ kenarları benzer şekilde temsil eder ("TensorFlow Lite Object Detection" 2019). Aynı anda 10 nesne tanınabilmektedir.

3.2.4 Otonom bir araç için mobil AR uygulaması geliştirilmesi

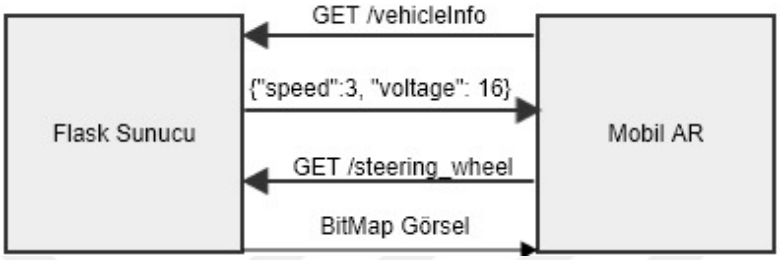
Otonom bir aracın, dış dünyayı nasıl algıladığını uzaktan veya araç içi geliştirilen uygulamalar ile kontrol edebilmek hedeflenmiştir. Otonom bir araç için sunulan yönlendirme ve bilgilendirme ekranları kullanıcıların yolculuk esnasında kendilerini daha güvende hissetmelerine yardımcı bir araç olarak düşünülmektedir. Otonom araçlarda, aracın anlık enerji tüketimi, hız bilgisi, etrafındaki nesneleri tanıma işlemi ve direksiyon döndürme kabiliyeti takip edilmesi gereken ihtiyaçların başında gelmektedir. Bu tez çalışması kapsamında geliştirilen Android mobil AG uygulaması, hem araç içi hem de araç dışı bilgi verebilmektedir.

Geleneksel mobil uygulamalar interaktif bir hizmet sunabilmek için çoğunlukla bir sunucu-istemci (server - client) mimaride işlem yapmaktadır. Çalışmada geliştirilen mobil AG uygulaması ile anlık araç hız değeri değişimi, eş zamanlı enerji tüketimi ve gerçek zamanlı direksiyon açı değişimini ekranda gösterebilmek için, araçtan veri alınması gerekmektedir. Araç üzerinden veri alma işlemi, geliştirilen bir sunucu ile sağlanmaktadır. Sunucu – istemci mimarilerinde, sunucular bağımsız (standalone) olarak hizmet vermektedir. Bu çalışmada, RACECAR üzerinde RİS ile birlikte çalışabilecek sunucu mimarisi geliştirilmiş olup araç üzerinden veri Mobil AG uygulamasına gönderilmektedir. Şekil 3.18, RACECAR üzerinde geliştirilen bir sunucu ve Mobil AG arasındaki iletişim mimarisini en temel şekilde ifade etmektedir.



Şekil 3.18. RACECAR sunucu – mobil AR iletişim

RACECAR sunucu mimarisi Python ile Flask web çerçevesi kullanılarak geliştirilmiştir. Flask, web uygulamaları geliştirmek için kullanılan sunucu mimarisidir. Flask üzerinde yazılan RESTful web servis iletişim alt yapısı ile Mobil AG uygulaması haberleştirilmiştir. Şekil 3.19, RACECAR üzerinde yazılan Flask Sunucu ve Mobil AG ile veri alışverişinde kullanılan Sunucu-İstemci RESTful mimari haberleşme teknolojisini göstermiştir.



Şekil 3.19. Mobil AR- Flask Sunucu RESTful Web servis mimari

Sunucu – İstemci arasında RESTful veri iletişim mimarisi kullanılmıştır. REST, temsili durum transferi olarak ifade edilir. HTTP protokolünü kullanarak, GET ve POST gibi isteklerde bulunup, bu isteklere çeşitli formatlarda yanıt aldığı esnek yapıya sahip iletişim yoludur. REST iletişim protokolünde istemci-sunucu arasında taşınan verilerde istemciye ait detaylar bulunmaz. Böylece, daha hafif çözümler sunmaktadır. REST ile transfer edilen verilerin formatı, HTML, JSON, XML veya farklı bir formatta olabilmektedir. RACECAR Sunucu ve Mobil AR arasında JSON veri transfer formatı seçilmiştir.

4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

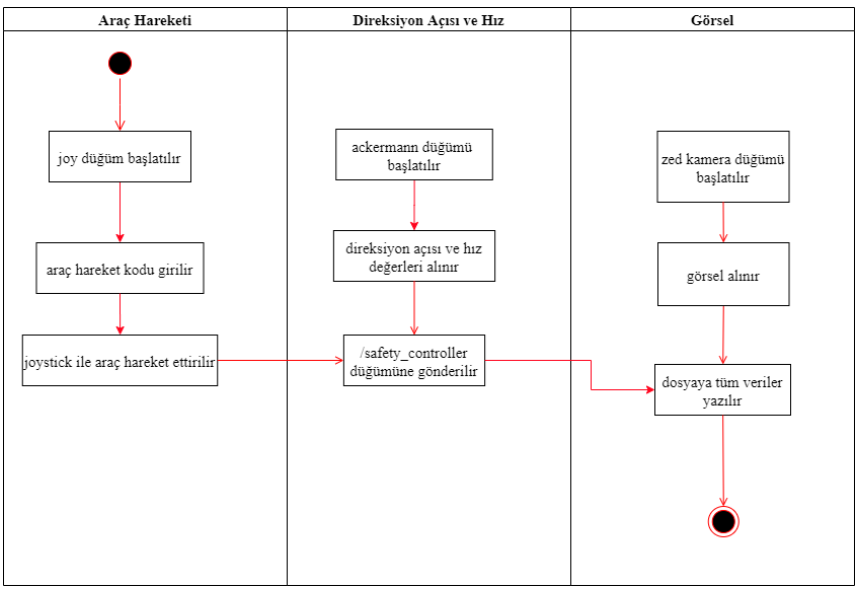
4.1 Otonom Araç Konfigürasyonu Sonuç Bulguları

Adım 1: Başlangıçta, otonom olarak bir sürüş eyleminin gerçekleştirilebilmesi için; 3.1.4. adımındaki veri toplama işlemi gerçekleştirilerek, alınan tüm görsel, hız ve direksiyon açısı değerleri ".csv" uzantılı bir dosyaya kaydedilir. Otonom bir araç için çevreden toplanılan veri setinin özelliği Çizelge 4.1' de gösterilmiştir. Kaydedilen tüm verilerle 3.1.5 adımındaki ağ yapısı ile Nvidia uçtan uca öğrenme modeli eğitilmiştir. Araç gerçek zamanlı hareket işlemini gerçekleştirirken kameradan almış olduğu görüntüyü modele göndermektedir. Model direksiyon açı değeri tahmini (regresyon) gerçekleştirmektedir.

Çizelge 4.1. Otonom bir araç için veri seti özellikleri

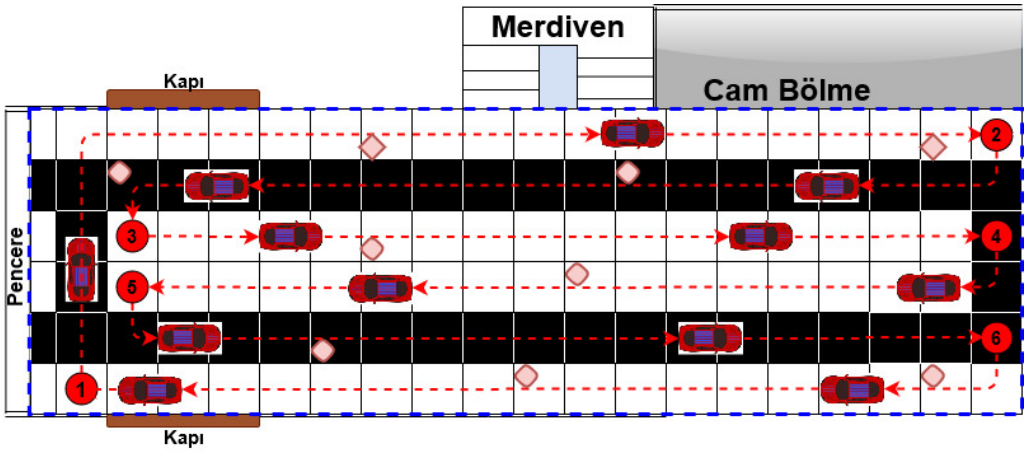
Eğitimde kullanılan veri setinin özellik ismi	**Veri setinin özellik tanımı**	**Verinin boyutu / birim olarak değeri / sayısı / zaman**
Görsel sayısı	Kaydetme işlemi esnasında, her bir görsel için sıra numarası verilir.	1,2, 3, ….., N
Görsel boyutu	Kaydedilen görselin piksel cinsinden değerini ifade eder.	533x312
Görüntünün dosya boyutu	Kayıtlı görüntünün megabayt/kilobayt olarak ölçülen dijital boyutudur.	108 KB
Araç hızı	Anlık olarak araç üzerinden alınan hız bilgisidir.	1-6 m/s
Direksiyonun açı sal değeri	Hareket halinde direksiyon açı değeri değişimidir.	radyan

Tüm işlem adımlarını içeren aktivite diyagramı Şekil 4.1'de verilmiştir. Böylece otonom sürüş işlemi için veri toplama adımı gerçekleştirilmiştir.



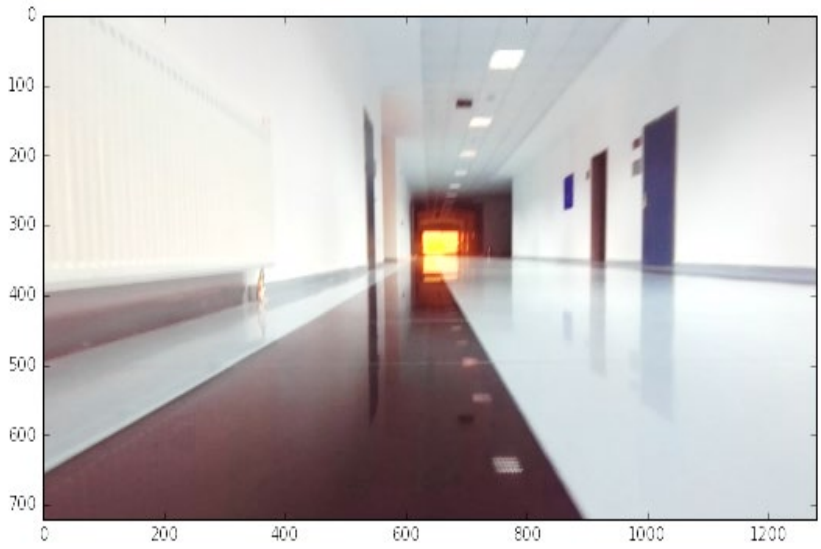
Şekil 4.1. Gerçek zamanlı veri toplama işleminde sistem aktivite diyagramı

Veri toplama işlemi için tasarlanmış ortamının krokisi Şekil 4.2'de verilmiştir.



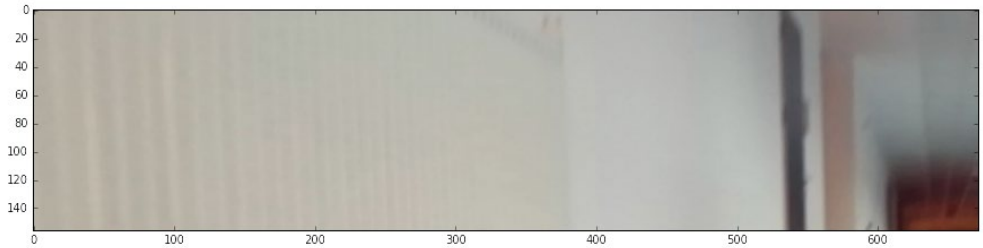
Şekil 4.2. Gerçek zamanlı veri toplama işleminde kullanılan ortamın krokisi

Adım 2: İkinci adımda, ağ yapısı oluşturulmadan önce dosyadan tüm veri seti okunmuş ve tüm görüntüden görüş alanını ifade eden üst kısım çıkarılmıştır. Bu işlem, ağ eğitimi ve eğitim sonrasında hızlı tahmin işlemleri yapabilmeyi sağlamıştır. Kameradan alınan orijinal görüntü Şekil 4.3'de gösterilmiştir.



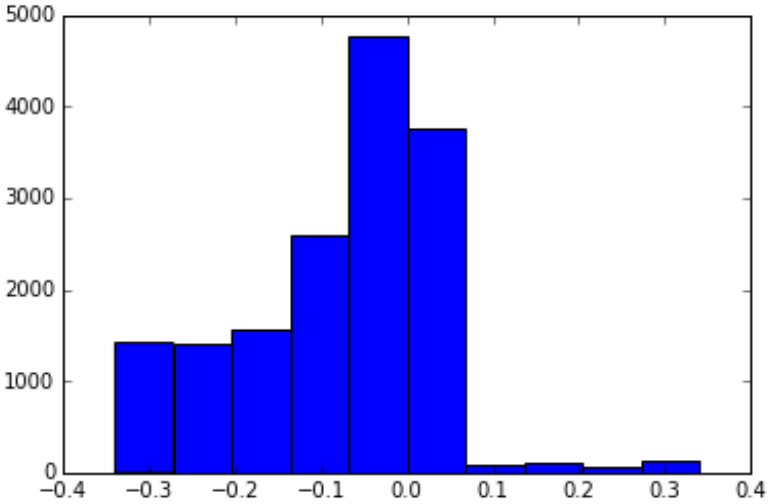
Şekil 4.3. Gerçek zamanlı veri toplama işleminde alınan orijinal boyutta bir görüntü

Ağ eğitimi ve eğitim sonrası tahmin işlemleri için, görüntüye uygulanan kırpma işlemi sonrası, görüntüdeki değişim Şekil 4.4'de verilmiştir.



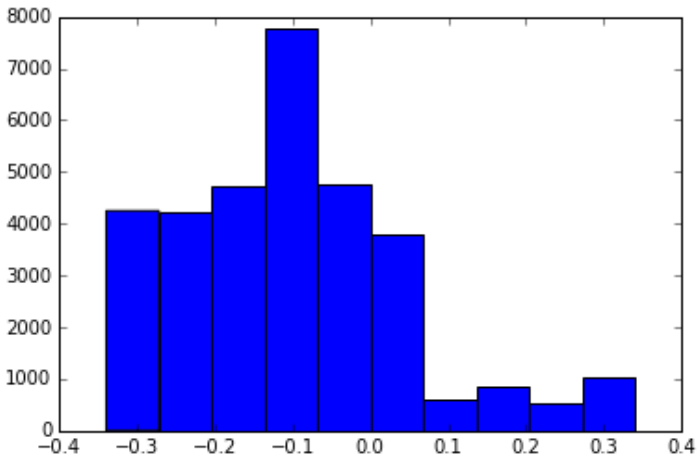
Şekil 4.4. Orijinal görüntüye kırpma işlemi uygulandığında elde edilen görüntü

Adım 3: Üçüncü adımda, okunan tüm veri seti içerisindeki direksiyon açı değerlerinin dağılımına bakılmıştır. Eğer açı değeri dengeli bir dağılıma sahip değilse, eğitim işleminin düzgün olmamasına neden olabilmektedir. Şekil 4.5'teki histogram grafiğinde elde edilen açısal değerlerin dağılımı gösterilmiştir.



Şekil 4.5. Direksiyon açı değerlerinin dağılımı grafiği

Açısal değerlerin sayıca az olanları, eğitim işleminde öğrenme problemleri oluşturacağı için bu verilerin dağılım üzerindeki etkisinin artırılması gereklidir. Bunun için veri zenginleştirme yöntemi kullanılarak, sayısı az olan veriler listeye yeniden eklenmiş (data augmentation) ve dağılımın daha düzgün hale gelebilmesi sağlanmıştır. Şekil 4.6 veri dağılımını Şekil 4.5'e göre daha dengeli bir hale geldiğini göstermiştir. Bu yöntem, en doğru çözüm olmamakla birlikte nispeten ilk duruma göre daha dengeli bir dağılım sağlamıştır.



Şekil 4.6. Daha dengeli bir dağılımı ifade eden histogram değerleri

Otonom bir aracın, güneşli, yağmurlu, karlı ve ışıkların farklı açılarla geldiği tüm ortamlarda sürüş eylemini tamamlaması beklenir. Dış ortamda veri toplama işlemi gerçekleştiriliyorsa, araç ilerlerken kameranın görüş alanına nesnelerin gölgelerinin düşmesi, güneş ışıklarının geliş açılarının değişimi gibi faktörler elde edilen görüntülerde bozulmalara neden olabilmektedir. Bu çalışmada, veri toplama işlemi kapalı bir ortam (bina koridoru) içerisinde gerçekleştirilmiş olup koridor boyunca spot ışıkları ortam aydınlatmasını sağlamaktadır. Bu çalışma özelinde, en temel problem ışıkların geliş açısıdır. Eğitim için toplanan görüntülerin hepsi aynı ışık değerine sahip değildir. Her durumda aracın hareket edebilmesi için toplanan görüntülerin %50'sine ışık artırma fonksiyonu uygulanmıştır. Bu fonksiyon, görüntüleri önce HSV'ye dönüştürerek, V kanalını aşağı veya yukarı ölçeklendirerek ve RGB kanalına dönüştürerek farklı parlaklıkta görüntüler üretecektir.

Çizelge 4.2. Işık değeri dönüşüm işlemi

Satır No	Kod
1	imageTemp=cv2.cvtColor(image,cv2.COLOR_BGR2HSV)
2	imageTemp = np.array(image1, dtype = np.float64)
3	random_bright = .5+np.random.uniform()
4	imageTemp [:,:,2] = imageTemp [:,:,2]*random_bright
5	imageTemp [:,:,2][imageTemp [:,:,2]>255] = 255
6	imageTemp = np.array(imageTemp, dtype = np.uint8)
7	imageTemp=cv2.cvtColor(imageTemp,cv2.COLOR_HSV2RGB)

Adım 4: Son olarak, tüm veri seti içerisinde karıştırma işlemi uygulanıp *%80*'i eğitim ve *%20*'si test verisi olarak ayrıştırılmıştır (Kocic´2019). Bazı uygulama geliştiriciler, tek bir doğrulama bölünmesine sahip olmak için k-katlamalı çapraz doğrulama yönteminden kaçınmayı tercih ederler (k-fold 2020), çünkü çapraz doğrulama yöntemi hesaplama açısından maliyetli bir işlem haline gelebilmektedir. Bu yüzden, k-katlamalı çapraz doğrulama yöntemi kullanılmamıştır.

Veriden öğrenen makine modelleri tasarlanırken, kullanılan algoritma ve parametreler probleme göre değişkenlik göstermektedir. Var olan verilere göre değişkenlik gösteren parametreler hiper-parametre olarak adlandırılmaktadır. Aşağıda

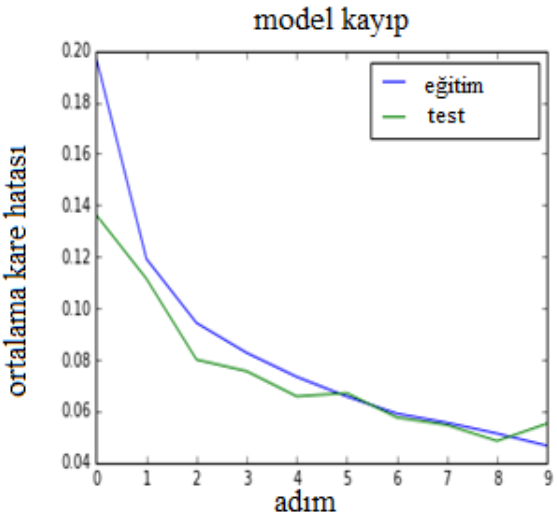
verilen hiper-parametreler ile oluşturulan öğrenme modeli en ideal modeli garanti etmemektedir.

Çalışmada kullanılan uygulama ve hiper-parametre ayarı aşağıdaki gibidir;

- Uygulama alt yapısında Tensor Flow ile Keras back-end kullanılmıştır.
- Sabit öğrenme oranı ile parametre optimizasyonu için Adam optimizasyon fonksiyonu kullanılmıştır (Kocic´2019).
- Adam optimizasyon fonksiyonu için öğrenme oranı 10^{-2}, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-08 olarak kullanılmıştır. Bu değerler Keras tarafından önerilen başlangıç değerleridir.
- Eğitim aşamasında performans değerlendirilmesinde ve modelin performansını izleyebilmek için ortalama karesel hata (mean square error (MSE)) kayıp fonksiyonu (loss function) olarak kullanılmıştır. Denklem 4.1 ile belirtilen MSE fonksiyonundaki y_i değeri beklenen çıktı ve y_i' ise modelin öngörüsüdür. MSE, tahminlerin ortalama karesel hatasını ölçer. Bu fonksiyon, her bir giriş verisi için beklenen değerle model öngörüsü arasındaki farkın karesini alarak ortalama hesaplar. Elde edilen değer ne kadar yüksek olursa, model o kadar kötü anlamına gelir. Bu durum, tüm ağın eğitim sonucu için tümüyle olumsuz bir durum teşkil etmez çünkü mükemmel bir model için sonuç sıfır olur. Burada dikkat edilmesi gereken durum verideki gürültüdür. Veri gürültülü ise mükemmel bir model olsa bile yüksek MSE değerleri elde edilir. Genel olarak MSE değerleri 1'den küçükse, modelin kötülüğü göz ardı edilebilir.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - y_i')^2 \tag{4.1}$$

Ağ eğitim işlemi için 3.1.6. 'da belirtildiği gibi bir ağ yapısı oluşturulmuştur. Ağ eğitim işlemi sonucunda oluşturulan modelin hata fonksiyonu Şekil 4.7'de gösterilmiştir. Şekil 4.7, eğitim ve doğrulama veri setleri için adım sayısı arttıkça, MSE kayıp fonksiyonunun değerlerinin nasıl değiştiğini göstermektedir. MSE kaybının, 1. adımdan sonra hızla azaldığı ve 5-8. adım arasında sabit kaldığı görülmektedir. 8-9. adımlarda değişim görülmektedir fakat yapılan test işlemleri sonucunda hata oranındaki değişim ihmal edilmiştir.



Şekil 4.7. MSE kayıp fonksiyon grafiği

Modelin doğruluk grafiği değişimine bakılarak ağ eğitim işleminin duracağı veya devam edileceği öngörülebilir. Eğitim işlemi sonucunda, her iki veri setinin her bir adımdaki doğruluk değeri Şekil 4.8'de verilmiştir.

Şekil 4.8. Eğitim ve test veri seti üzerinde modelin doğruluk değeri grafiği

Şekil 4.8'deki grafiğe göre, modelin her iki veri seti içinde karşılaştırılabilir bir beceri gösterdiği ve modelin eğitim veri setini daha fazla öğrenmemiş olduğu görülmektedir. En son adımdan sonra test ortamında yapılan gerçek araç sürüş eyleminden sonra model eğitim işleminin yeterli olduğu görülmüş ve ağ eğitimine devam edilmemiştir.

Ağ eğitim işlemi için, MIT RACECAR üzerindeki Jetson TX2 geliştirici kartı kullanılmıştır. Bu kart, NVIDIA Pascal, 256 CUDA GPU, HMP Dual Denver 2/2 MB L2

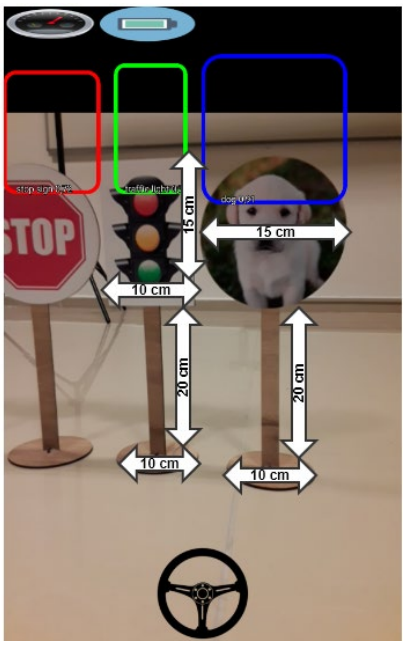
+ Quad ARM A57/2 MB L2 CPU, 8 GB 128 bit LPDDR4 58.3 GB/s hafıza özelliklerine sahiptir. Toplam ağ eğitimi için geçen süre ***4269.793*** saniye ‘dir.

4.2 Otonom Araç – Mobil AR Uygulaması Entegrasyon İşlemi ve Sonuçları

Mobil AR uygulaması için 3.2.3. adımında belirtildiği gibi bir alt yapı kurulmuştur. Araç üzerine gerçek zamanlı nesne tanıma işlemini gerçekleştirebilen bir tablet yerleştirilirmiştir. Tabletin araç üzerinde dik durabilmesi için iki parça tasarlanmış ve parçalar üç boyutlu yazıcıdan çıkarılmıştır. Tablet kamerası için en uygun görüş alanı, RACECAR üzerinde yer alan üstteki plakanın arka kısmıdır. 3B yazıcıdan çıkarılan parçalar, tahta bir plaka ile RACECAR üzerine sabitlenmiştir.

Uygulama içerisine eklenen COCO SSD MobileNet v1 modeli içerisinde yer alan 9 nesne modeli için test senaryosu oluşturulmuştur. Test senaryosu, 9 adet levhadan oluşan bir parkur ortamında nesne algılama işleminden oluşmaktadır. Levhalar ise, kamyon, kedi, köpek, bisiklet, motosiklet, yaya, tren, stop levhası, trafik lambası şekillerinden oluşturulmuştur. Araç bu levhalara göre bir sürüş eylemi gerçekleştirmemiştir (durma, yeniden hareket etme veya en uygun yol). Levhaların tamamı mobil AG uygulaması içerisinde nesne tanıma işlemi için kullanılmıştır.

Deneysel Çalışma 1: Birinci adımda, tahta levhalardan oluşan nesneler tasarlanmış ve 2B boyutlu görseller bu levhalar üzerine yerleştirilmiştir. Şekil 4.9, tahta levhaların tasarımını ve ölçülerini göstermektedir.

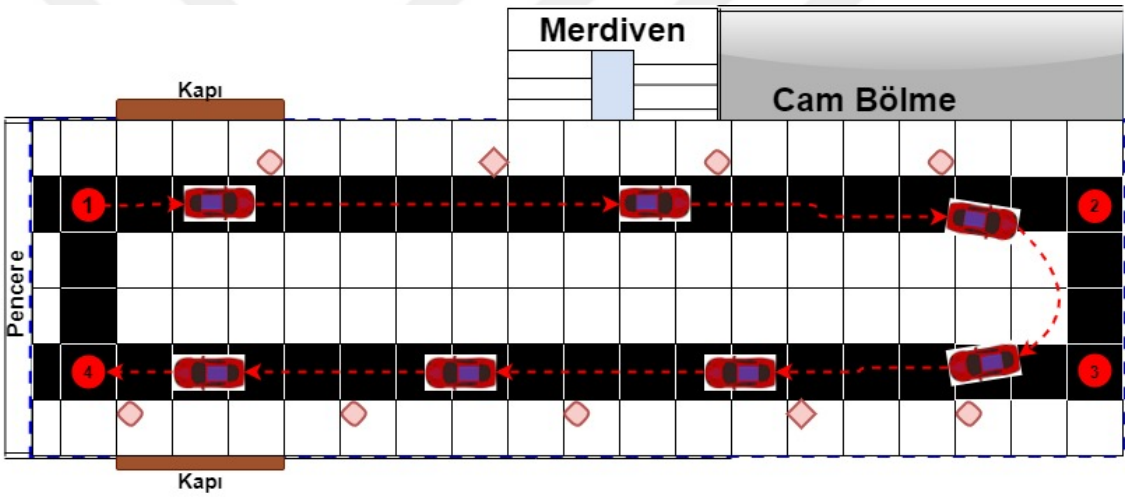


Şekil 4.9. Parkur ortamı için levha tasarımı ve ölçüleri

Şekil 4.9, temsili olarak hazırlanan test senaryosundaki yuvarlak ve dikdörtgen levhaları göstermektedir. Yuvarlak levhaların üst kısmındaki görsellerin olduğu kısmın çapı 15 cm'dir. Yuvarlak alan ile taban arasındaki uzunluk ise 20 cm'dir. Levhaların dik durmasını sağlayan taban çapı ise 10 cm'dir. Dikdörtgen levhaların, üst kısmının ölçüleri 10x15 cm'dir. Dikdörtgen alanın yerden yüksekliği 20 cm'dir ve taban çapı ise 10 cm'dir.

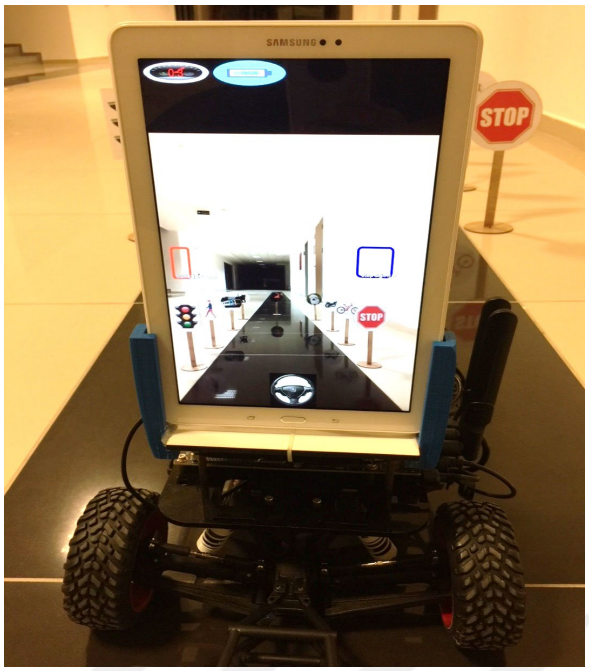
Şekil 4.9, üzerindeki sarı, yeşil ve mavi dört gensel gösterimlerden de görüldüğü üzere, bu ölçülere sahip bir levhanın mobil AR uygulaması ile algılanabildiği görülmektedir. Böylece, test senaryosu için tasarlanan levhaların parkur ortamında kullanılabilirliği kanıtlanmıştır.

Deneysel Çalışma 2: İkinci adımda, aracın kendi kendine sürüş eylemini gerçekleştireceği sürüş alanı ortamı hazırlanmıştır. Gerçek zamanlı sürüş işleminin gerçekleştirildiği ortam krokisi Şekil 4.10'da verilmiştir.



Şekil 4.10. Gerçek zamanlı sürüş ortamı krokisi

9 levha, eşit aralıklarla sürüş yapılacak alana yani aracın sürüş rotasının dışında kalan alana arka arkaya yerleştirilirmiştir. Sadece 1 nesne aracın hareket yoluna yerleştirilmiştir. Tüm nesneler kameranın algılayabileceği görüş alanı içerisine yerleştirilmiştir. Araç otonom olarak hareket ettiğinde, AG uygulamasının tüm nesnelerin aynı görüş alanı içerisinde olduğu durumda nasıl bir tanıma işlemi yaptığı gözlemlenmiştir. Şekil 4.11, ikinci adım için oluşturulan senaryoyu göstermektedir.

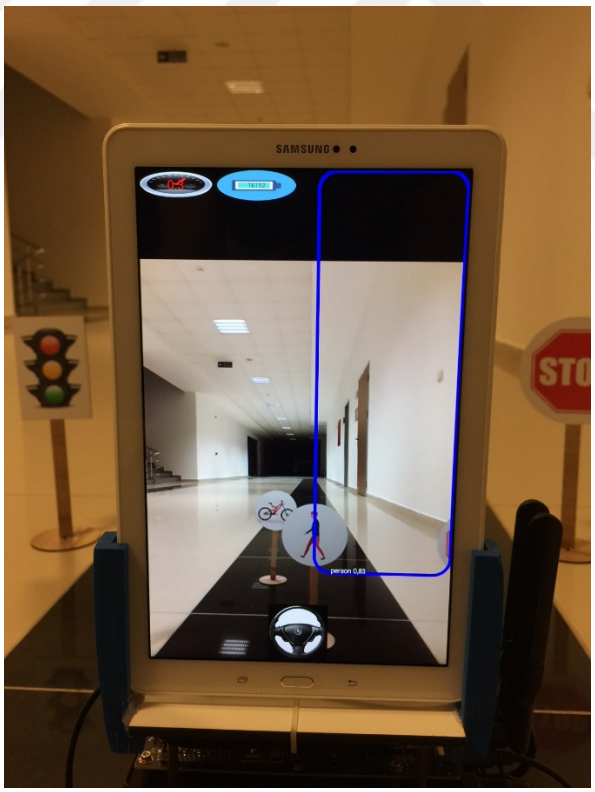


Şekil 4.11. Eşit aralıklarla yerleştirilen levhaların mobil AR ile ilk tanıma işlemi

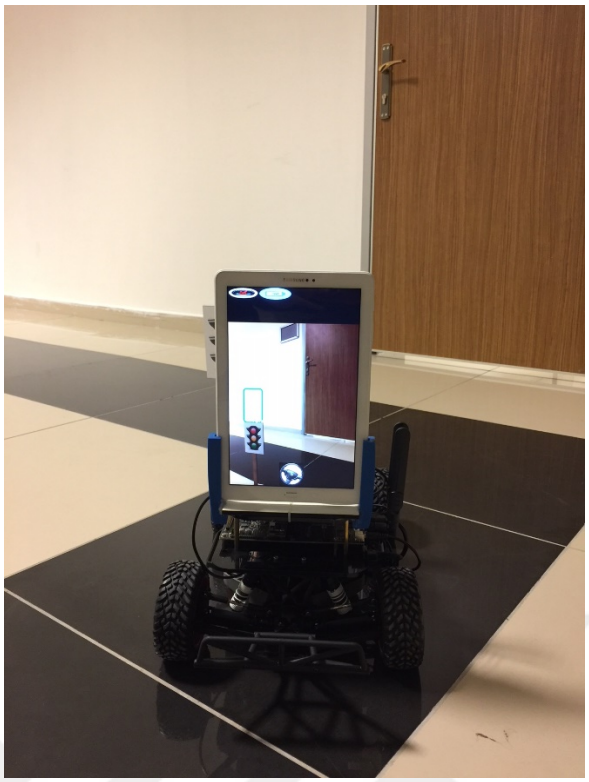
İkinci senaryo sonucunda, mobil AG uygulaması ile nesne algılama işleminin gerçekleştirilebileceği gerçek zamanlı olarak gösterilmiştir.

Deneysel Çalışma 3: Üçüncü adımda ise, aracın otonom sürüş işlemini gerçekleştireceği tüm rota alanına levhalar dağıtılarak bir senaryo oluşturulmuştur. Şekil 4.12'de stop levhası, Şekil 4.13'de yaya tanıma ve Şekil 4.14'de trafik tanıma işlemi gerçek zamanlı otonom sürüş işlemi esnasında mobil AG ile farklı konumlarda tanınan üç nesne gösterilmiştir.

Şekil 4.12. Stop levhası tanıma işlemi



Şekil 4.13. Yaya tanıma işlemi



Şekil 4.14. Trafik levhası tanıma işlemi

RACECAR'ın otonom olarak hareket edebilmesi için ortamdan veri toplanılmıştır. Toplanılan veriler eğitilmiştir ve sonucunda model elde edilmiştir. Ortamdan veri toplayabilmek ve nesne tanıma işlemleri için test senaryosu oluşturulmuştur. Bir sunucu uygulaması geliştirilmiştir ve araç üzerinden veriler anlık olarak alınmıştır. Sunucu uygulaması ile birlikte çalışabilen mobil AG uygulaması geliştirilmiştir. Mobil AG üzerinde, araç hızı, enerji tüketimi, direksiyon durumu ve nesne tanıma işlemi gösterilmiştir.

5. SONUÇLAR VE ÖNERİLER

5.1 Sonuçlar

Bu tez çalışmasında; otonom araçların hayatımızda yaygın olarak kullanılmasıyla ortaya çıkabilecek şu problemlere çözümler ortaya konmaya çalışılmıştır:

- Otonom bir araç insansız sürüş eylemini nasıl gerçekleştirir?
- Otonom bir araç dış dünyayı nasıl algılamaktadır?
- Sürücülü ve sürücüsüz hareket edebilen otonom bir aracın yapmış olduğu eylemler AG teknolojisi kullanılarak nasıl görüntülenebilir?
- Otonom araç içerisinde hareket eden yolcuların kendilerini daha güvende hissedebilmeleri için AG teknolojisi kullanılabilir mi?
- AG teknolojisi otonom bir araç içerisinde yardımcı ve daha güven verici bir teknoloji olarak kullanılabilir mi?
- Sayısız kanun ve yönetmeliklerle karşı karşıya kalacak olan insansız araçların eylemleri ileriye dönük nasıl kayıt altına alınır ve AG ile geçmişe dönük veriler görüntülenebilir mi?

Bu tez çalışmasında, geliştirme kiti olarak MIT üniversitesinin geliştirdiği RACECAR'ın donanım alt yapısı kullanılarak temel otonom araç yazılımı geliştirilmiştir. Bu araç kendi kendine sürüş eylemlerini gerçekleştirebilmektedir. Aracın dış dünyayı insan gözü gibi nasıl algıladığı ise sensörlerden ve araç üzerindeki çevre birimlerinden alınan verilerin işlenmesi ile elde edilmiştir. Aynı zamanda araç üzerinde tüm mobil cihazlarla haberleşebilen sunucu-istemci mimarisi geliştirilmiştir. Bu mimari, web servisler aracılığıyla istekte bulunan cihazlara hız, batarya, direksiyon hareketleri verilerini döndürmektedir. Toplanan tüm veriler mobil AG teknolojisi kullanılarak gösterilmiştir. Mobil AG uygulaması, Android işletim sistemine sahip cihazlarda kullanılmak üzere tasarlanmış ve geliştirilmiştir. Bu uygulama, araç içi ve dışı kullanılmak üzere bir alt yapıya sahiptir. Aynı şekilde araç içerisinde seyahat eden yolcuların kendilerini daha güvende hissedebilmeleri için aracın dış dünyayı nasıl algıladığı AG ile mobil cihaz üzerinde gösterilmiştir. Sürücü ve seyahat eden yolcu

olmadan, otonom sürüş işlemlerini yapan bir aracın, enerji, hız ve direksiyon değişimleri uzaktan mobil AG ile izlenebilmiştir.

TensorFlow Lite ile Android yapay sinir ağları kullanılarak nesne tanıma işlemleri gerçekleştirilmiştir. Bu aşamada, bazı problemlerle yüzleş ilmiştir, ilk olarak mobil AG uygulaması içerisinde tanınan nesnelerin dikdörtgen bir kutu içerisinde alınmasında, koordinat bilgilerinde sapmaların ve nesnelerin gerçek yerlerinin belirlenmesinde bazı yanlışlıklar görülmüştür. Bu durum, olumsuz gibi görünse de AG uygulamalarında istenilen bir sonuçtur. İkinci olarak mobil cihazın aracın arka tarafına aparatlarla monte edilmesinden dolayı, aracın görüş alanını dik bir açıyla kesmesi ve yüksek hızlarda (3m/sn) bazı nesnelerin tanınmasında zorluk yaşanmıştır. Bu kısımda ele alınması gereken diğer bir durum ise gerçek hayattaki nesnelere karşılık gelen plakaların boyutlarının ve gerçeğe tam yakınlarının oluşturulması gerektiği sonucu elde edilmiştir.

Yapılan testler sonucunda, aracın sürekli yüksek batarya değerlerine ihtiyaç duyması ve düşük voltaj değerlerinde kararsız davranışlar sergilediği gözlemlenmiştir (12.6V ve aşağı değerler için). Aracın güç tüketimine ekstra bir yük getirmemek için mobil AR uygulamasının çalıştığı mobil cihazın elektrik ihtiyacı araç üzerinden alınmamış ve mobil cihazın kendi bataryasından sağlanmıştır.

Literatür araştırmaları sonucunda otonom sistemler ve araçlar için çeşitli AG uygulamaların geliştirildiği görülmüştür. Bu tez çalışmasında geliştirilen sunucu-istemci mimari yapısı ve mobil AG uygulaması, birden fazla otonom sistemlerle birlikte kullanılabilirliği ve modülerliği gösterilmiş, mobil olması bir avantaj olarak sunulmuş ve gelecekte ele alınacak bazı problemlere ışık tutarak literatüre katkı yapması amaçlanmıştır.

5.2 Öneriler

Bu tez çalışması, sınırlı imkânlar dâhilinde otonom araç geliştirme kiti üzerinde gerçekleştirilmiştir. Ele alınan problemler, geliştirilen yazılım alt yapısı ve gerçek hayata uygulanabilmesi için gerçek bir araca ve gelişmiş donanıma ihtiyaç duyulmaktadır. Ayrıca test işlemlerinin, gerçek hayatı temsil eden bir otonom araç parkurunda ve farklı iklim koşullarında yapılması daha uygun olacaktır.

Android Sinir Ağları ve TensorFlow Lite ile Nesne Tanıma işlemi adımında aşağıdaki durumlara dikkat edilmesi gerekmektedir:

- Sistem kullanımı: Yapay sinir ağlarının çalışması, pil gücü kullanımını artırabilecek çok fazla hesaplama içermektedir. Uygulama tercihi, yoğun ve uzun işlemler gerçekleştiriyorsa, pil durumu kontrol edilmelidir.
- Uygulama boyutu: Uygulama boyutuna dikkat edilmelidir. Uygulama içerisine eklenen modellerin boyutu (megabayt) çok fazla yer kaplayabilir. APK'da büyük modelleri kullanmak, kullanıcıları aşırı derecede olumsuz etkiliyorsa, uygulama yüklendikten sonra modellerin indirilmesi, daha küçük modellerin kullanılması veya hesapların bulutta çalıştırılması düşünülmelidir.

KAYNAKLAR

"3D Graphics with OpenGL.", 2012, *https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html.* Erişim tarihi: 05.03.2020.

"Android Neural Networks API.", 2019, *https://developer.android.com/ndk/guides/neuralnetworks*. Erişim tarihi: 05.03.2020.

Arun Mani Sam, R&D Software Engineer., 2019, "Developing SSD-Object Detection Models for Android Using TensorFlow." https://www.inspirisys.com/objectdetection_in_tensorflowdemo.pdf. Erişim tarihi: 05.03.2020.

Bojarski, Mariusz, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, et al., 2016, "End to End Learning for Self-Driving Cars," 1–9.

Committee, On-Road Automated Driving., 2016. "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles."

"CS231n Convolutional Neural Networks for Visual Recognition.", 2019. *http://cs231n.github.io/convolutional-networks/#overview*. Erişim tarihi: 05.03.2020.

E.Sutherland, Ivan., 1968. "A Head-Mounted Three Dimensional Display." In Proceedings of the AFIPS Fall Joint Computer Conferance, 757–64.

Fredriksson, J, B Kulcsar, and J Sjöberg., 2015, "Proceedings of the 3rd International Symposium on Future Active Safety Technology Towards Zero Traffic Accidents," 411-418.

How, Jonathan P., Yu Fan Chen, John L. Vian, Ali-Akbar Agha-Mohammadi, Shayegan Omidshafiei, Rajeev Surati, and Nazim Kemal Üre., 2015, "MAR-CPS: Measurable Augmented Reality for Prototyping Cyber-Physical Systems," 1–13.

"Rospy.", 2017, http://wiki.ros.org/rospy. Erişim tarihi: 05.03.2020.

"ROS.", 2019, http://www.ros.org/. Erişim tarihi: 05.03.2020.

Jackel, Lawrence D., David Sharman, Charles E. Stenard, B. Ivan Strom, and Derek Zuckert., 1995, "Optical Character Recognition for Self-Service Banking." AT&T Technical Journal 74 (4): 16–24.

Joseph, Lentin., 2018, Robot Operating System for Absolute Beginners_ Robotics Programming Made Easy-Apress, 295.

Kocic´ Jelena, Jovicic Nenad, Drndarevic´ Vujo, 2019, " An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms " sensors, 1-26.

"k-fold", 2020, https://cs231n.github.io/classification/#val. Erişim tarihi: 05.05.2020.

Khandelwal, Pulkit, P. Swarnalatha, Neha Bisht, and S. Prabu., 2015, "Detection of Features to Track Objects and Segmentation Using GrabCut for Application in Marker-Less Augmented Reality." Procedia Computer Science 58 (January): 698–705.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton., 2015, "ImageNet Classification with Deep Convolutional Neural Networks." Journal of Geotechnical and Geoenvironmental Engineering 12: 04015009.

LeCun, Y., B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel., 1989, "Backpropagation Applied to Handwritten Zip Code Recognition." Neural Computation, 541–51.

Melo, José, and Aníbal Matos., 2017, "Survey on Advances on Terrain Based Navigation for Autonomous Underwater Vehicles." Ocean Engineering 139 (April): 250–64.

"MIT RACECAR Mobile Platform." 2017. http://racecar.mit.edu. Erişim tarihi: 05.03.2020.

Modi, Karankumar., 2018, "Evaluation of Holographic Head-up Display to Enhance Driving Safety" 248-249.

Mousazadeh, Hossein., 2013, "A Technical Review on Navigation Systems of Agricultural Autonomous Off-Road Vehicles." Journal of Terramechanics 50 (3): 211–32.

"Nvidia Cnn.", 2019, https://developer.nvidia.com/discover/convolutional-neural-network. Erişim tarihi: 05.03.2020.

"Openzeka.", 2019, https://openzeka.com/. Erişim tarihi: 05.03.2020.

Palmarini, Riccardo, John Ahmet Erkoyuncu, Rajkumar Roy, and Hosein Torabmostaedi., 2018, "A Systematic Review of Augmented Reality Applications in Maintenance." Robotics and Computer-Integrated Manufacturing 49 (February): 215–28.

Pan, Zengxi, Joseph Polden, Nathan Larkin, Stephen Van Duin, and John Norrish., 2012, "Recent Progress on Programming Methods for Industrial Robots." Robotics and Computer-Integrated Manufacturing 28 (2): 87–94.

Paulo Lima, João, Rafael Roberto, Francisco Simões, Mozart Almeida, Lucas Figueiredo, João Marcelo Teixeira, and Veronica Teichrieb., 2017, "Markerless Tracking System for Augmented Reality in the Automotive Industry." Expert Systems with Applications 82: 100–114.

Quigley, Morgan, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng., 2009, " ROS: an open-source Robot Operating System," in Proc. Open-Source Software workshop of the International

Conference on Robotics and Automation (ICRA), 679–86.

Rastogi, Vibhor., 2017, "Virtual Reality Based Simulation Testbed for Evaluation of Autonomous Vehicle Behavior Algorithms (Theses)." Clemson University, 48.

"Rqt_graph.", 2018, http://wiki.ros.org/rqt_graph. Erişim tarihi: 05.03.2020.

Rüßmann, Michael, Markus Lorenz, Philipp Gerbert, Manuela Waldner, Jan Justus, Pascal Engel, and Michael Harnisch., 2015, "Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries." Boston Consulting 62 (4): 40–41.

Sewak, Mohit, Md. Rezaul Karim, and Pradeep Pujari., 2018, Practical Convolutional Neural Networks : Implement Advanced Deep Learning Models Using Python, 220.

Stern, Raphael E., Shumo Cui, Maria Laura Delle Monache, Rahul Bhadani, Matt Bunting, Miles Churchill, Nathaniel Hamilton, et al., 2018, "Dissipation of Stop-and-Go Waves via Control of Autonomous Vehicles: Field Experiments." Transportation Research Part C: Emerging Technologies 89, 205–21.

T.Azuma, Ronald., 1997, "Survey of Augmented Reality," 355–85.

"TensorFlow Lite Object Detection.", 2019, https://www.tensorflow.org/lite/models/object_detection/overview. Erişim tarihi: 05.03.2020.

Tönnis, Marcus, Christian Sandor, Gudrun Klinker, Christian Lange, and Heiner Bubb., 2005, "Experimental Evaluation of an Augmented Reality Visualization for Directing a Car Driver ' s Attention." In Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society, 56–59.

Uzun Y., Bilban M., 2019, "Autonomous Vehicle and Augmented Reality Usage" International Journal of Engineering and Management Research, 39-43.

Walsh, Corey H., 2018, "Localizing, Modeling, and Drifting an Autonomous RACECAR." Massachusetts Institute of Technology.

Wen, Ming-Chang, and Shih-Chung Kang., 2014, "Augmented Reality and Unmanned Aerial Vehicle Assist in Construction Management," 1570–77.

Wetzstein, Gordon., 2018, "The Graphics Pipeline and OpenGL III." EE267 Virtual Reality, 39.

Zaccone, Giancarlo, and Md. Rezaul Karim., 2018, Deep Learning with TensorFlow - Second Edition, 484.

Zhu, J., S. K. Ong, and A. Y.C. Nee., 2013, "An Authorable Context-Aware Augmented Reality System to Assist the Maintenance Technicians." International Journal of Advanced Manufacturing Technology 66 (9–12): 1699–1714.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	MEHMET BİLBAN
Uyruğu	**:**	T.C.
Doğum Yeri ve Tarihi	**:**	Beyşehir 26.06.1991

EĞİTİM

Derece		**Adı, İlçe, İl**	**Bitirme Yılı**
Lise	**:**	Enis Şanlıoğlu Anadolu Lisesi, Seydişehir, Konya	2009
Üniversite	**:**	İnönü Üniversitesi, Malatya	2014
Yüksek Lisans:		Necmettin Erbakan Üniversitesi, Konya	**-**
Doktora	**:**		

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
2014-2016	Magis Teknoloji	Yazılım Mühendisi
2017-	Necmettin Erbakan Üniversitesi	Öğretim Görevlisi

UZMANLIK ALANI

Back-End sistemler
Mobil Uygulama
Otonom Sistemler
Artırılmış Gerçeklik

YABANCI DİLLER

İngilizce

BELİRTMEK İSTEĞİNİZ DİĞER ÖZELLİKLER

YAYINLAR

Uzun Y., Bilban M., 2019, "Autonomous Vehicle and Augmented Reality Usage" International Journal of Engineering and Management Research, 39-43.