T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

METRO CAR SİSTEMLERİNDE TAKİP VE BİRLEŞME ALGORİTMALARININ İYİLEŞTİRMESİ VE PYTHON İLE SİMÜLE EDİLMESİ

YÜKSEK LİSANS TEZİ

MUHAMMED SAİD PİLEVNELİ

DENİZLİ, MART - 2025

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

METRO CAR SİSTEMLERİNDE TAKİP VE BİRLEŞME ALGORİTMALARININ İYİLEŞTİRMESİ VE PYTHON İLE SİMÜLE EDİLMESİ

YÜKSEK LİSANS TEZİ

MUHAMMED SAİD PİLEVNELİ

DENİZLİ, MART - 2025

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

MUHAMMED SAİD PİLEVNELİ

ÖZET

METRO CAR SİSTEMLERİNDE TAKİP VE BİRLEŞME ALGORİTMALARININ İYİLEŞTİRMESİ VE PYTHON İLE SİMÜLE EDİLMESİ

YÜKSEK LİSANS TEZİ

MUHAMMED SAİD PİLEVNELİ

PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

(TEZ DANIŞMANI:PROF. DR. ABDULLAH TAHSİN TOLA)

DENİZLİ, MART - 2025

Günümüzde bireysel araç sayısındaki artış, kent içi ulaşımı sürdürülebilir hale getirmeyi zorlaştırmakta ve şehirlerde daha esnek, hızlı ve çevre dostu ulaşım çözümlerine olan ihtiyacı artırmaktadır. Bu bağlamda, söz konusu gereksinimlere cevap verebilecek yenilikçi bir ulaşım modeli olarak MetroCar Sistemi (MCS) geliştirilmiştir.

MCS, kent içi ulaşımda hız, güvenlik ve esneklik sağlayan, tek kişilik araçlarla donatılmış çok etmenli ve çift modlu bir sistemdir. Bu sistem, kullanıcılara bir noktadan diğerine taşımacılık imkânı sunarak bireysel araç kullanımına benzer bir rahatlık sağlarken, aynı zamanda otonom sürüş teknolojisi ile sürücüye ihtiyaç duymadan güvenli bir yolculuk yapma avantajı ve konforu sunmaktadır.

Bu tez çalışması kapsamında ise, MCS'nin daha güvenli ve verimli çalışmasını sağlamak amacıyla "Birleşme" ve "Takip" algoritmalarında çeşitli iyileştirmeler yapılmış; bu iyileştirmeler Python tabanlı özel bir simülasyon ortamında test edilmiştir. Sonuçlar, yeni algoritmaların MCS'nin mevcut algoritmalarına kıyasla performans artışı sağladığını göstermiştir.

Özellikle, MCS sisteminde bir aracın öndeki aracı takip etmesi durumunda gerçekleştirebileceği hızlanma veya yavaşlama hareketlerine ek olarak hızını koruma yeteneği de kazandırılmıştır. Bu sayede, hızdaki ardışık artış ve azalışları dengelemek için kullanılan alçak geçiren filtre veya histerezis gibi yöntemlere artık ihtiyaç kalmadığı görülmüştür.

Ayrıca mevcut birleşme algoritmasından farklı bir algoritma geliştirilmiş ve performans testleri yapılarak bu algoritmalar karşılaştırılmıştır. Sonuçlar, yeni algoritmanın mevcut algoritmaya kıyasla performans açısından üstünlük sağladığını ortaya koymaktadır.

Sonuç olarak, geliştirilen yeni algoritmalar, MCS'nin güvenli, verimli ve sürdürülebilir bir şehir içi ulaşım çözümü olarak potansiyelini arttırmakta; mevcut algoritmalara kıyasla daha yüksek performans sağladığını ortaya koymaktadır.

ANAHTAR KELİMELER: MetroCar Sistemi, Otonom Araçlar, Çift Modlu Ulaşım, Python

ABSTRACT

IMPROVEMENT OF TRACKING AND MERGE ALGORITHMS IN METRO CAR SYSTEMS AND SIMULATION WITH PYTHON
MSC THESIS
MUHAMMED SAİD PİLEVNELİ
PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE
ELECTRICAL AND ELECTRONICS ENGINEERING

(SUPERVISOR:PROF. DR. ABDULLAH TAHSİN TOLA)

DENİZLİ, MARCH 2025

The increasing number of individual vehicles today complicates the sustainability of urban transportation and heightens the need for more flexible, faster, and environmentally friendly transportation solutions in cities. In this context, the MetroCar System (MCS) has been developed as an innovative transportation model that addresses these requirements.

MCS is a multi-agent, dual-mode system equipped with single-seater vehicles that provides speed, safety, and flexibility in urban transportation. This system offers users the convenience of transportation from one point to another, resembling the comfort of personal vehicle use, while also providing the advantage of a safe journey without the need for a driver through autonomous driving technology.

Within the scope of this thesis, various improvements have been made to the "Merging" and "Tracking" algorithms to ensure that MCS operates more safely and efficiently; these improvements have been tested in a Python-based specialized simulation environment. The results demonstrates that the new algorithms provide a performance increase compared to the existing algorithms of the MCS.

Specifically, the system has been enhanced to allow a vehicle tracking another vehicle to not only accelerate or decelerate but also maintain it’s speed. Consequently, it has been observed that methods such as low-pass filters or hysteresis, which were previously necessary to dampen the sequential increases and decreases in speed during straight path tracking, are no longer required.

In addition, an algorithm different from the existing merging algorithm has been developed, and performance tests have been conducted to compare these algorithms. The results indicate that the new algorithm offers superior performance compared to the existing one.

In conclusion, the newly developed algorithms enhance the potential of MCS as a safe, efficient, and sustainable urban transportation solution, demonstrating higher performance relative to the existing algorithms.

KEYWORDS: MetroCar System, Autonomous Vehicles, Dual-Mode Transportation, Python

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

SEMBOL LİSTESİ

$\mathbb{R}$	:	Reel Sayılar Kümesi
m	:	Metre
km	:	Kilometre
ms	:	Milisaniye
δ	:	Güvenli takip mesafesi
φ	:	Sönümleyici Alan
φ_{min}	:	Minimum Sönümleyici Alan
τ	:	Yenileme Zamanı
α_{dec}	:	Yavaşlama İvmesi
α_{acc}	:	Hızlanma İvmesi
i,j,m,n	:	İndis Numaraları
v	:	Hız
x	:	Mesafe
a	:	İvme
f_s	:	Örnekleme Frekansı
f_{max}	:	Maksimum Frekans

AÇIKLAMALAR LİSTESİ

MCS : MetroCar Sistemi
TYS : Trafik Yönetim Sistemi
ADAS : Advanced Driver-Assistance System
SAE : Society of Automotive Engineers
CAV : Connected and Autonomus Vehicles
OTY : Otonom Trafik Yönetimi
GNSS : Global Navigation Satellite System
IMU : Inertial Measurement Unit
SLAM : Simultaneous Localization and Mapping
V2V : Vehicle to Vehicle
V2I : Vehicle to Infrastructure
DSRC : Dedicated Short-Range Communications
AASHTO : American Association of State Highway and Transportation Officials

ÖNSÖZ

Bu süreçte, beni destekleyen, bilgi ve deneyimlerini esirgemeyen değerli hocam Prof. Dr. Abdullah T. Tola'ya en içten teşekkürlerimi sunuyorum. Sayın hocam, çalışmamın her aşamasında sağladığınız rehberlik ve teşvik, bu tezin başarıya ulaşmasında büyük bir rol oynamıştır. Ayrıca, beni araştırma alanında cesaretlendirmeniz ve bilimsel düşünme yeteneğimi geliştirmem konusunda sunduğunuz destek ve bulunduğunuz katkılar için minnettarım.

Tezin hazırlanmasında emeği geçen tüm arkadaşlarıma, aileme ve bana ilham veren herkese teşekkür ederim. Umarım bu çalışma, şehir içi ulaşım sorunlarına yönelik yeni perspektifler ve çözümler sunar.

1. GİRİŞ

1.1 Tez Konusu ve Amacı

Günümüzde yoğun kent yaşamı, artan trafik hacmi ve çevresel kirlilik gibi sorunların şehir yapısı üzerindeki olumsuz etkilerini sürekli olarak ortaya koymaktadır.

Bu bağlamda, toplu taşıma sistemleri; güvenlik, maliyet ve zaman yönetimi gibi birçok alanda şehir içi ulaşımı iyileştirme potansiyeline sahip etkin çözümler arasında yer almaktadır. Özellikle yoğun trafiğe duyarsız çalışması nedeniyle metro sistemleri, diğer toplu taşıma seçeneklerine kıyasla verimliliğiyle öne çıkmaktadır. Trafik sıkışıklığından bağımsız rotaları ve yüksek taşıma kapasiteleri sayesinde metro gibi sistemler, güvenilir, hızlı ve sürdürülebilir ulaşım çözümleri sunmaktadır.

Ancak, toplu taşımanın sunduğu bu avantajlara rağmen, bireysel araç kullanımı, sunduğu kişisel konfor, mahremiyet ve hijyen avantajları nedeniyle tercih edilmeye devam etmektedir. Bireysel araçların kullanımındaki artış şehir içi trafiğinde sıkışıklığa ve sorunlara neden olmaktadır. Bu etkilerin en aza indirgenmesi için Trafik Yönetim Sistemi (TYS) kavramına ihtiyaç duyulmaktadır.

TYS'ler, artan nüfus ve bireysel araç sayısıyla birlikte modern kentlerde ulaşım taleplerini güvenli, sürdürülebilir ve düzenli bir şekilde karşılamak amacıyla geliştirilmektedir. Kent içi ulaşımın mevcut altyapısının sınırlı kapasitesi, günümüzde artan ulaşım talebi karşısında yetersiz kalmakta ve sıkışıklık, güvenlik sorunları ve çevresel etkiler gibi pek çok sorunu beraberinde getirmektedir. Bu bağlamda, TYS'ler geleneksel ulaşımın sınırlarını aşacak şekilde, hız ve güvenliği sağlayan, çevresel etkileri azaltan ve yolculuk süresini optimize eden yenilikçi çözümler sunmayı hedefler.

Bu sistemler, geleneksel ulaşım ağına entegre olan sistemler ve bağımsız ve özgün bir altyapıya sahip sistemler olarak iki alt başlıkta incelenebilir. Bu iki sistem

türü, şehir içi ulaşımda güvenli, hızlı ve sürdürülebilir çözümler sunma yolunda farklı yaklaşımlar sergilemektedir.

Entegrasyon üzerine yoğunlaşan sistemler, şehir içi trafik akışına dahil olarak geniş bir kullanım alanı sağlar, ancak mevcut yol yapısı ve trafik dinamikleriyle uyum gereksinimi nedeniyle karmaşık bir kontrol ve güvenlik yapısı gerektirir. Bağımsız sistemler ise, özel hatlara sahip olup, tıpkı metro sistemleri gibi geleneksel trafiğe karışmadan çalışır. Bu yapılar, özel olarak ayrılmış rotalarda sabit hız ve otomatik kontrol avantajları sunarak trafik sıkışıklığına duyarlı olmadan kesintisiz bir yolculuk imkânı tanır.

Mevcut altyapıya doğrudan entegre olmak yerine, bağımsız ve özel hatlara sahip ulaşım sistemlerinin sunduğu alternatif çözümler, dış etkenlerin getirdiği öngörülemez belirsizlikleri en aza indirmektedir. Bu tür bir bağımsız altyapı ile ulaşım yapan yolcular, trafik akışından etkilenmeden yolculuk yapabilmekte, kapıdan kapıya ulaşım kolaylığından faydalanabilmektedir. Böylece, yoğun kent trafiğine olan bağımlılığı azaltan ve kent içi ulaşım üzerindeki baskıyı hafifleten bu tür sistemler, çevresel etkileri azaltan, güvenli ve esnek bir ulaşım modeli olarak öne çıkmaktadır.

MetroCar Sistemi (MCS), bu bağlamda tıpkı bir metro sistemi gibi özgün bir altyapısı olan, otonom ve manuel kullanımın olduğu, kişiye ait tek kişilik araçların bulunduğu, çift modlu, çok etmenli, çok şeritli, noktadan noktaya ulaşım sağlayan yenilikçi ve akıllı bir sistemdir. Normal bir metro sisteminin aksine olarak MCS yapısına dahil olan araçlar, sistemdeyken otonom bir davranış sergilerken araçların sistemden ayrılmaları durumunda manuel olarak kullanılabilmektedir.

Ancak MCS yenilikçi bir yaklaşım olmasının getirdiği avantajlarla birlikte tam olgunlaşmış bir sistem değildir. Bunun için bazı çalışmalar yapılmış olup halen geliştirilmeye devam edilmektedir.

Bu kapsamda bu tez çalışması ile MCS'de bulunan düz yolda araç takip etme algoritmasında iyileştirmeler yapılmış, iki şerit arasında yeni bir birleşme algoritması geliştirilmiştir. Bu çalışmalar Python ortamında sıfırdan tasarlanan bir simülasyon aracılığıyla test edilmiştir.

1.2 Literatür Çalışmaları

Artan trafik hacimleri, şehir içi ulaşımı daha verimli ve sürdürülebilir hale getirme ihtiyacını doğurmuştur. Literatürde yer alan ilgili çalışmalar incelendiğinde, gelişen otonom araç teknolojileri bu doğrultuda önemli bir alternatif olarak öne çıkmaktadır. Ayrıca, MCS gibi özel hatlarla sağlanan ulaşım çözümleri, kent içi ulaşımın geleneksel kalıplarının ötesine geçerek farklı bir yaklaşım sunmaktadır. Otonom araç teknolojileri ve bu teknolojilerin trafik üzerindeki etkileri, akıllı TYS'ler gibi çeşitli alanlarda araştırmalar yapılmıştır.

Hayalet trafik sıkışıklıkları, sürücü davranışlarının ve araç etkileşimlerinin bir sonucu olarak trafik akışında ani dalgalanmalar ve tıkanmaların meydana geldiği, önemli bir olgu olarak literatürde yer bulmaktadır. Bu fenomen, özellikle yoğun trafik akışlarında, dışsal bir sebep olmaksızın oluşan sıkışıklıkların dinamiklerini anlamak açısından dikkat çekmektedir. Yapılan literatür çalışmalarında, otonom araçların ve akıllı sistemlerin bu tür sıkışıklıkları azaltmadaki etkisi incelenmiştir.

Bu bağlamda, literatürdeki araştırmalar otonom ulaşım teknolojilerinin tarihsel gelişimini ele alırken, trafik yönetiminde nasıl devrim niteliğinde çözümler sunduğunu ve bu teknolojilerin hayalet trafik sıkışıklıkları gibi karmaşık problemlerin çözümüne yönelik katkılarını ele almaktadır. Otonom araçların sınıflandırma sistemleri ve bu sistemlerin şehir içi ulaşımda sunduğu yenilikler, çağdaş literatürün önemli bir odağı olmuştur ve bu tez kapsamında detaylı olarak değerlendirilecektir.

1.2.1 Otonom Araçların Gelişimi

Otonom ulaşım sistemlerinin başlangıcı 100 yıla yakın olmasına rağmen yeni milenyumda internetin de hızlanarak gelişmesiyle birlikte özellikle son 20 yıldır hızla geliştiği görülmektedir. Yakın gelecekte ise kişisel otomobiller ve hatta toplu taşıma sistemleri için otonom taşımanın geçerli olacağı öngörülebilmektedir.

Otonom araçlarının günümüze kadar olan bazı gelişimleri şu ise şekilde sıralanabilir;

- 2000 Öncesi Dönem
- 2000’den Sonrası ve Günümüz

1.2.1.1 2000 Öncesi Dönem

Modern yarı otonom araçların başlangıcı, 1926’da *Houdina Radio Control* tarafından tanıtılan radyo kontrollü *Linriccan Wonder* adlı araçla atılmıştır. Bu araç, arka antenleriyle başka bir arabadan gelen radyo sinyallerini alıp, küçük motorlar yardımıyla yönlendirilmekteydi. Bu basit sistem, otonom araçların ilk örneklerinden biri olarak gösterilmektedir. *Linriccan Wonder*'ın geliştirilmiş bir modeli *Achen Motors* tarafından “*Phantom Auto”* olarak adlandırılmış ve *Milwaukee*’de sergilenmiştir (Bimbraw 2015).

1939 Dünya Fuarı’nda General Motors, Norman Bel Geddes'in "*Futurama*" sergisini destekleyerek yol içine gömülü devrelerle çalışan elektrikli araçları tanıtmıştır. Bu sistem, yoldaki devrelerin radyo sinyalleriyle arabaları yönlendirmesi esasına dayanmaktadır (Bimbraw 2015).

1953’te RCA Labs, laboratuvar zeminine yerleştirilmiş tellerle yönlendirilen bir mini otonom araba geliştirmiştir. 1958'de Nebraska’da, mühendisler Leland Hancock ve L. N. Ress, bu sistemi gerçek bir karayolu üzerinde yaklaşık 122 metre uzunluğundaki bir şeritte denedi. Yeraltına gömülü devreler ve yol kenarına yerleştirilen ışıklar, arabaları yönlendirmek için sinyaller gönderiyor ve araçların varlığı ile hızını tespit edebiliyordu. General Motors da bu projeye katılarak radyo alıcıları, sesli ve görsel uyarı sistemleriyle otonom sürüş, hızlanma ve frenlemeyi simüle eden iki araç üretmiştir (Bimbraw 2015).

1960’larda, General Motors’un “*Motorama*” fuarında, sürücüye ihtiyaç duymayan bir elektronik yönlendirme sistemi olan *Firebird* sergilendi (Burgan 1999; Cranswick 2013; Temple ve Dennis Adler 2006). Ohio State Üniversitesi ve İngiltere'nin Taşıma Araştırma Laboratuvarı da benzer sürücüsüz araba projelerini test etti; bunlardan biri *Citroen DS* modeliydi ve yola gömülü manyetik kablolarla iletişim kurarak 130 km/s hızda etkili bir sürüş gerçekleştirdi (Bimbraw 2015; Pressnell 1999; Temple ve Adler 2006).

Amerika Birleşik Devletleri Karayolları Bürosu, 1960'larda elektronik kontrollü deneysel bir otoyol inşası girişiminde bulundu. Ohio, Massachusetts, New York ve California, bu projeyi üstlenmek için başvurdu ve Ohio Valisi DiSalle bu otomasyon deneylerinin geleceği için destek verdi (Bimbraw 2015).

1980'lere gelindiğinde, Almanya'da Bundeswehr Üniversitesi'nden Ernst Dickmanns ve ekibinin geliştirdiği görsel kontrollü sürücüsüz Mercedes-Benz minibüs, trafiksiz yollarda 63 km/s hıza ulaşmayı başardı (Bimbraw 2015).

Otonom araç teknolojisindeki ilerlemelerle birlikte çeşitli ulusal ve uluslararası projeler başlatıldı. 1987-1995 yılları arasında, EUREKA tarafından yürütülen ve 1 milyar ABD dolarından fazla yatırım yapılan Prometheus Projesi bu alandaki önemli çalışmalardan biridir (Flyte 1995, Xie ve diğ. 1993). ABD Savunma Bakanlığı'na bağlı DARPA'nın otonom kara aracı (*Autonomous Land Vehicle*) projesi de Carnegie Mellon Üniversitesi, Michigan Çevre Araştırma Enstitüsü, Maryland Üniversitesi gibi kuruluşların katkısıyla geliştirildi. Bu proje kapsamında, bilgisayar görüşü, LIDAR ve otonom kontrol sistemleri kullanılarak 31 km/s hızla otonom yol izleme başarıyla gösterildi (Bimbraw 2015).

1991'de ABD Kongresi, otonom araç ve otoyol sistemleri geliştirilmesi için ISTEA Taşımacılık Yetkilendirme Kanunu'nu kabul etti ve Ulusal Otomatik Otoyol Sistemi Konsorsiyumu kuruldu. Bu proje, General Motors ve UC-Berkeley gibi ortakların da katılımıyla 1997'de San Diego'da tanıtıldı ancak bütçe yetersizliği nedeniyle iptal edildi. Aynı yıl, Daimler-Benz ve Bundeswehr Üniversitesi'nin geliştirdiği VaMP ve Vita-2 ikiz robot araçlar, Paris'te yoğun trafikte 130 km/s hıza ulaştı ve şerit değiştirme gibi otonom manevraları insan müdahalesiyle gerçekleştirdi (Bimbraw 2015).

1995 yılında, Dickmanns'ın otonom S-Class Mercedes-Benz'i Münih'ten Kopenhag'a gidip dönen 1.6 km'lik bir yolculuk yaptı. Araba, 180 km/s hızla ve insan müdahalesi olmadan yüzde 95 oranında otonom sürüş sağladı ve gerçek zamanlı karar alarak diğer araçları sollayarak ilerleyebildi. Bu gelişmeler, otonom araçların insan sürücülerden daha hızlı ve güvenli olma potansiyelini göstermekteydi (Bimbraw 2015).

1.2.1.2 2000'den Sonrası ve Günümüz

2000'li yıllarda otonom teknoloji alanında yalnızca sıradan bir ilerleme değil, araç bağımsızlığına dair toplumsal bakışı köklü bir şekilde değiştiren büyük bir dönüşüm yaşandı. DARPA'nın *"Grand Challenge"* projeleri, Stanford ve Carnegie Mellon gibi üniversitelerin yenilikçi çabaları ve Google ile Tesla gibi sektör devlerinin tamamen sürücüsüz araç girişimleri, otonom araç alanında yeni bir çağın kapılarını araladı. Stanford ve Carnegie Mellon gibi üniversitelerin bu yarışmalardaki başarıları, otonom teknolojinin sınırlarını genişletirken, 2009'da Google'ın başlattığı sürücüsüz araç projesi (daha sonra Waymo olarak bilinecek) sektörde bir mihenk taşı oldu. Tamamen otonom bir araç geliştirme hedefiyle yola çıkan Waymo, yoğun test süreçleriyle teknolojinin ilerlemesine büyük katkılarda bulundu. Aynı dönemde Tesla Motors, Otopilot özelliği ile radar, kamera ve ultrasonik sensörleri birleştirerek adaptif hız sabitleyici ve şerit takip asistanı gibi yarı otonom sürüş özelliklerini kullanıcı deneyimine sundu. Bu gelişmeler, otonom araç teknolojisinin geleceğini yeniden şekillendirmeye yönelik ortak bir vizyon ve yaratıcılıkla zengin bir teknolojik manzara oluşturmuştur (Alabyad ve diğ. 2024).

İnternet ve yapay zekâ teknolojilerindeki gelişmeler de bu alandaki ilerlemeyi hızlandırdı. Google, Tesla, Apple, Nissan ve Mercedes gibi dev şirketler otonom araç geliştirme çalışmalarına ciddi yatırımlar yaptı. Özellikle Google, yapay zekâ temelli çalışmalarıyla otonom araç testlerinde milyonlarca kilometre yol kat etti ve Tesla ise ilk üretim araçlarına bu teknolojiyi entegre ederek sektördeki konumunu güçlendirdi. Bu sırada, Volvo ve Samsung gibi diğer büyük sektör aktörleri de otonom araç güvenliği ve test çalışmaları için kapsamlı projeler başlattılar. Çin'de ise Baidu ve Tencent gibi teknoloji devleri, yapay zekâ ve derin öğrenme teknolojilerini kullanarak kendi otonom sürüş sistemlerini geliştirmeye odaklandılar (Alabyad ve diğ. 2024).

Otonom sürüş alanındaki bu hızlı ilerleyiş, güvenlik standartlarının belirlenmesi ve teknolojilerin sınıflandırılmasını da gerekli kılmıştır. Bu bağlamda, Gelişmiş Sürücü Destek Sistemleri (ADAS, *Advanced Driver-Assistance System*) kapsamına giren otonom araç sistemlerinin bir standart altında değerlendirilebilmesi için Otomotiv Mühendisleri Topluluğu (*Society of Automotive Engineers*, SAE)

tarafından ilk defa 2014 yılında sınıflandırılmıştır. Ancak bu tanımlama her geçen gün güncellenmektedir (SAE 2021[a]).

SAE tarafından sınıflandırılan ADAS, altı seviyeye ayrılmış olup, Seviye 0'da sistem aracın kontrolünü üstlenmez; yalnızca sürücüye bilgi sunar ve bu bilgilerin değerlendirilmesi tamamen sürücünün sorumluluğundadır. Bu seviyedeki işlevler, park sensörleri, çevresel görüş, trafik işareti tanıma, şerit ihlali uyarısı, gece görüş, kör nokta bilgi sistemi, geri çapraz trafik uyarısı ve önden çarpışma uyarısı gibi özellikleri içerir. Seviye 1 ve 2'de araç, hâlâ sürücünün kararlarına büyük ölçüde bağımlıdır, ancak Seviye 1'de sistem yalnızca tek bir işlevi yönetirken, Seviye 2'de birden fazla işlevde destek sağlayabilir. Örneğin, Seviye 1'de adaptif hız sabitleyici, acil fren desteği, otomatik acil fren yardımı, şerit tutma ve şerit ortalama gibi fonksiyonlar bulunur. Seviye 2 sistemleri ise otoyol yardımı, otonom engel kaçınma ve otonom park etme gibi gelişmiş işlevleri barındırır. Seviye 3 ile 5 arasındaki seviyelerde araç kontrolü kademeli olarak artar. Seviye 3 belirli ortamlarda çalışabilir ayrıca sistem belli koşullarda insan sürücüden devralmasını talep edebilir. Seviye 4 araç tamamen otonom sürüş yapar insan müdahalesi olmaz ancak belli ortam koşulları gerektirir. Seviye 5 tam otonom sürüş anlamına gelir ve hiçbir bir ortamda müdahale gerektirmez. Günümüzde, bazı ileri düzey sistemler hâlâ yaygın olarak ticari kullanımda değildir; örneğin, otoyol sürüş asistanı Seviye 3, otomatik vale park etme ise Seviye 4 kapsamındadır (SAE 2021[b]).

Ancak bu tanımlama, standartlar günden güne değişmektedir. SAE 2014 yılında yaptığı ilk otonom sürüş seviye standartlarını en son 2021 yılında güncellemiştir (SAE 2014, SAE 2021[b]).

1.2.2 Hayalet Trafik Sıkışıklığı

Hayalet trafik sıkışıklığı (*phantom traffic jam*), araçların yol koşulları veya dış bir engel olmaksızın, yalnızca sürücü davranışları ve trafik akışı dinamiklerinden kaynaklanarak yolda bir yoğunluk ve dur-kalk hareketleri oluşturduğu bir trafik türüdür. Bu tür sıkışıklıklar, genellikle insan kaynaklı sürüş hataları, hız değişiklikleri veya takip mesafesinin yeterince korunmaması gibi nedenlerle ortaya çıkar.

Bu fenomenin temel mekanizması, bir aracın hızını aniden azaltması ve arkasındaki araçların buna tepki vermesi ile başlar. Tepki süresi ve frenleme davranışları nedeniyle bu dalgalanma, geriye doğru bir zincirleme etki yaratır ve hiçbir dış sebep olmaksızın trafiğin yoğunlaşmasına ve hatta durmasına dahi neden olabilir. Böyle bir duruma genellikle “dur-kalk dalgaları” ya da “trafik dalgaları” denir.

Hayalet trafik sıkışıklığı, literatürde mikro ve makro trafik modelleriyle açıklanmakta ve özellikle araç takip sistemleri, otonom araçlar, adaptif hız sabitleyiciler gibi teknolojilerin bu tür sıkışıklıkları azaltmada nasıl etkili olabileceği üzerinde durulmaktadır. Özellikle otonom araçların, insan davranışından kaynaklanan bu dalgaları absorbe edebilme potansiyeli önemli bir araştırma alanıdır. Literatürdeki belirli çalışmalar şu şekilde sıralanabilir:

Trafik sıkışıklığı, şehirlerde sıkça karşılaşılan bir sorun olup, bu durumun önemli nedenlerinden biri olan yasa dışı şerit değiştirme davranışlarının analiz edilmesi gereklidir; yapılan bir çalışmada, kentsel ekspres yollarda "hayalet trafik sıkışıklıkları" senaryosuna odaklanılarak, yasa dışı şerit değiştirme davranışlarının uzun vadede yol üzerindeki etkileri incelenmiş, sıkışıklığın gerçek zamanlı olarak yayılma kalıpları belirlenmeye çalışılmış ve sonuç olarak, bu tür davranışların 4,5 kilometrelik bir yol kesiminde yoğun trafik sıkışıklıklarına neden olduğu ve bu durumun yaklaşık 40 dakika devam ettiği tespit edilmiş; ayrıca, sıkışıklık dalgasının yayılma hızı ile her bir gözlem noktasındaki hız düşüşünün dalgalanmalarla birlikte sürekli azaldığı gözlemlenmiş olup, elde edilen bulguların trafik yönetimi alanında geniş bir referans kaynağı olabileceği ve ulaşım simülasyon sistemlerinde parametre ayarlamalarını desteklemek için ideal bir temel sunduğu ifade edilmiştir (Jiang ve diğ. 2018).

Bağlantılı ve Otonom Araçlar (*Connected and Autonomus Vehicles,* CAV), araç trafiğinin verimliliğini artırma potansiyeline sahiptir ve yapılan bir çalışmada, CAV'lerin otoyollarda karma trafik sistemlerinin dinamik davranışını, doğrusal olmayan dinamikler teorisi perspektifinden nasıl olumlu etkileyebileceği ele alınmaktadır; ilk olarak, insan sürücülü trafiğin, bireysel bir sürücünün fren yapması gibi küçük bir etkiyle ya akıcı bir şekilde ilerleyebileceği ya da sıkışıklığa neden olabileceği iki kararlı durum (bistabilite) olgusunu sergilediği ve bu durumun istenmeyen hayalet trafik sıkışıklıklarına yol açabileceği ortaya konmaktadır; doğrusal

olmayan dinamik modellerin analiziyle, iki kararlı durumun mekanizması açıklanmakta ve bu duruma neden olabilecek insan sürücü parametreleri belirlenmektedir; ikinci olarak, hem insan sürücülerin hem de CAV'lerin yer aldığı karma trafiğin incelenmesiyle, CAV'lerin doğrusal olmayan dinamik davranış üzerindeki etkileri analiz edilmekte ve CAV'lerin yeterince yüksek bir nüfuz oranına ulaşması halinde iki kararlı durumu ortadan kaldırabileceği, bu koşulu sağlayabilecek CAV kontrol parametrelerinin tespit edildiği belirtilmektedir (Molnár ve Orosz 2024).

Yapılan başka bir çalışmada, trafik akışının kararlılığını sağlamak amacıyla hayalet trafik sıkışıklıklarının meydana geldiği bir trafik akışına entegre edilmiş akıllı bir araç modeli tanıtılmaktadır. Akıllı araç, öndeki aracın hız ve mesafe bilgilerini paylaşma özelliğine sahiptir. Ayrıca, paylaşılan bilgiler sayesinde öndeki araçlardaki değişiklikleri önceden öngörebilmekte ve insan sürücülü araçlardan daha hızlı bir şekilde hızlanmaya başlayabilmektedir. Bu çalışmada, öndeki araçlarla bilgi paylaşımına izin veren, genelleştirilmiş bir *Nagel-Schreckenberg* modeli temelinde geliştirilen bir akıllı araç modeli önerilmektedir. Genelleştirilmiş *Nagel-Schreckenberg* modeli, bilgi paylaşımı yapılacak öncü araç sayısının keyfi olarak belirlenebilmesini sağlamaktadır ve yapılan analizler, öndeki iki veya daha fazla araçla bilgi paylaşımı gerçekleştiren bir akıllı aracın hayalet trafik sıkışıklıklarını ortadan kaldırabildiğini ortaya koymaktadır (Ishikawa ve Arai 2016).

1.2.3 TYS’lerde Otonom Araçların Yeri

TYS’lerde otonom araçların yeri, akıllı şehirler ve sürdürülebilir ulaşım çözümleri kapsamında giderek daha önemli bir hale gelmektedir. Özellikle son yıllarda, otonom araçların trafikteki varlığı, güvenliği artırma, trafik akışını iyileştirme ve çevresel etkileri azaltma gibi çeşitli hedefler doğrultusunda incelenmektedir. Otonom araç teknolojileri, TYS’lerde operasyonel verimliliği artırarak, trafik yoğunluğunu yönetme, acil durum yanıt sürelerini iyileştirme ve enerji kullanımını optimize etme konularında önemli avantajlar sunmaktadır. Bu kısımda, otonom araçların trafik yönetiminde nasıl konumlandığına, altyapı gereksinimlerine ve mevcut sistemlerle entegrasyonuna dair temel yaklaşımlar ve güncel araştırmalar ele alınacaktır.

MATEC Web of Conferences'da 2017 yılında yayımlanan *"Review on Driverless Traffic from Management Perspective"* başlıklı makalede, otonom araçların güvenlik ve verimlilik açısından sunduğu büyük avantajların yalnızca gelişmiş teknoloji ile değil, aynı zamanda uyumlu trafik yönetimi yaklaşımlarıyla da desteklenmesi gerektiği vurgulanmıştır. Mevcut trafik altyapısı, yasalar ve düzenlemelerin, otonom araçların potansiyel güvenlik ve verimlilik kapasitelerini kısıtladığı ifade edilmektedir. Gelecekte, otonom araçların yaygınlaşmasıyla birlikte ulaşım altyapısının biçimleri, kullanım kuralları ve kaza sorumluluğu ile ilgili düzenlemelerin de bu yeni sisteme uygun olarak güncellenmesi gerektiği belirtilmiştir (Chen ve diğ. 2017).

2017'de yayımlanan bu çalışmada, otonom ve bağlantılı araçların kullanımına yönelik bir simülasyon tabanlı TYS fikri önerilmiştir. Bu yaklaşım, araçların konum bilgilerini toplama ve iletme kapasiteleri, ayrıca rotalarını önceden belirleyebilme yetenekleri sayesinde oldukça umut vaat etmektedir. Simülasyon tabanlı sistem, derin sinir ağları (*Deep Neural Networks*), bulut bilişimi (*Cloud Computing*), grafik işlem birimi (Graphics Processing Unit, GPU) teknolojisi ve büyük veri (*Big Data*) işleme gibi yenilikçi teknolojilerden yararlanmaktadır. Başlangıçta, çevrimdışı trafik yönetiminde kullanılabilecek olumlu sonuçlar elde edilmiştir. Ancak gerçek zamanlı trafik yönetimini mümkün kılmak için, gelecek araştırmaların simülasyonların hızlandırılmasına, sinir ağlarının öğrenme ve çıkarım süreçlerinin geliştirilmesine ve meta-sezgisel yöntemlerin verimliliğinin artırılmasına odaklanması gerektiği belirtilmiştir (Gora 2018).

2018 yılında yapılan bir çalışmada, otonom sürüş teknolojisinin sunduğu güvenlik, konfor ve enerji verimliliği vaatlerini ele alırken, özellikle diğer yol kullanıcılarının bilinmeyen niyetlerinin yarattığı zorluklara odaklanmaktadır. Araçlar arası ve altyapıyla iletişim, farkındalığı artırarak iş birliğini sağlama potansiyeline sahiptir. CAV'ler otomasyon ve bağlantılılık alanında tek başına mümkün olanın ötesine geçerek mobiliteyi dönüştürebilir. Uygulama alanları; anlık kontrol ve planlama, mikro ölçekte trafik verileriyle yönlendirme, trafik sinyalleriyle koordineli konvoy sürüşü ve park garantili eko-mobiliteyi içerir. Bu makale, CAV'lar için bir kontrol ve planlama mimarisi sunmakta ve her işlevsel blokta mevcut durumları incelemektedir; özellikle enerji verimliliğini artıran tekniklere odaklanılmaktadır.

Mevcut algoritmaların genel bir özetini sunarak, birbirleriyle etkileşimlerini, CAV kontrolüne yönelik umut vadeden optimizasyon tabanlı yaklaşımları ve gelecekteki zorlukları ortaya koymaktadır. Bu çalışma, sürüş otomasyonu ve araç bağlantısının gelişiminde kontrol ve planlama algoritmalarının kritik önemini ele alıyor. Yapılan literatür taramasında, bağlantılı ve otonom araçlar için mevcut kontrol ve planlama algoritmaları incelenmiş, hiyerarşik bir yapı içinde bu teknolojiler değerlendirilmeye alınmıştır. Çalışmada, bu sistemlerin genel çerçevede nasıl rol oynadığı ve mevcut yaklaşımların hangi alanlarda eksik olduğu belirlenmiştir. Mevcut teknolojilerin büyük kısmında deneysel doğrulama eksikliği gözlenmiş; bu durum, özellikle gerçek araçlar üzerinde test yapılmasının önemini vurgulamaktadır. Araştırmacılar, kamu kurumları ve özel sektörün bu yönde ilerlediği belirtilirken, sahada doğrulama ve gerçek dünyayı temsil eden geniş çalışmalarının da gerekli olduğunu vurgulamıştır (Guanetti ve diğ. 2018).

2018'de yayımlanan çalışmada, Akıllı Ulaşım Sistemleri'nin (*Intelligent Transportation Systems*), otonom araçların kullanıma hazır hale geldiği dönemde trafik problemlerini çözmek için Otonom Trafik Yönetimi'ne (OTY) dayandığı belirtilmiştir. Bu sistemin, trafik sıkışıklığını ve ekonomik maliyetleri azaltma gibi hedefleri olduğu vurgulanmıştır. Bununla birlikte, OTY'nin genellikle otonom araçların kusursuz bir şekilde çalışmasını, basit yol altyapılarını, belirlenmiş trafik senaryolarını, sınırsız kablosuz iletişim olanaklarını ve yalnızca otonom araçların yol kullanıcısı olduğu bir ortamı varsaydığı ifade edilmiştir. Gerçek dünya koşullarında bu varsayımların OTY uygulamalarını sınırlayabileceği, bu nedenle OTY'nin iyileştirilmesine yönelik altı ana başlıkta araştırma yönlerinin ele alındığı belirtilmiştir. Bu sistemin trafik sıkışıklığını ve ekonomik maliyetleri azaltma hedefleri olduğu vurgulanmıştır. Bununla birlikte, OTY'nin genellikle otonom araçların kusursuz çalışması ve belirli yol altyapısı koşullarını varsaydığı ifade edilmiştir. Çalışmada, OTY'nin iyileştirilmesi adına akıllı grafik ışığı kontrol Sistemlerinin iyileştirilmesi, dur levhalarının değiştirilmesi, yol kapasitesinin artırılması, kavşak yönetimi, güvenliğin artırılması, trafik sıkışıklığının azaltılması başlıkları altında altı ana öneri sunulmuştur (El Hamdani ve Benamar 2018).

2019'da yayımlanan bir çalışmada, yalnızca otonom araçların kullanacağı yol ağlarında güvenlik ve performans optimizasyonu sağlamak amacıyla geliştirilmiş bir

TYS mimarisi tanıtılmıştır. Bu TYS, ağ düzeyinde en uygun rotaları belirlerken, yerel düzeyde çarpışmaları önlemek ve araçların en kısa sürede varış noktasına ulaşmasını sağlamak için en uygun güzergâhları planlamaktadır. Literatürdeki çoğu çalışmadan farklı olarak yalnızca tek bir kavşak yerine geniş bir yol altyapısını dikkate alarak optimizasyon sağlayan bu sistem, matematiksel programlama problemleriyle modellemeler yaparak araç güzergâhlarını gerçek zamanlı belirlemektedir. Yapılan deneyler, yöntemin uygulanabilirliği açısından umut verici sonuçlar sunmuştur. Çalışmanın gelecekteki hedefleri arasında daha karmaşık yol düzenlerinin denenmesi, sistemin en uygun rotaları belirlemek için makroskopik bir modelle entegrasyonu ve otonom araçların daha az gelişmiş bağlı otonom araçlarla etkileşimlerinin incelenmesi bulunmaktadır (Di Febbraro ve diğ. 2020).

2020 yılında yayımlanan bir editoryal yazıda, CAV teknolojilerinde kaydedilen büyük ilerlemelerin, bu sistemlerin trafik kontrolü üzerindeki potansiyel etkilerini ve uygulama yöntemlerini önemli ölçüde derinleştirdiği belirtilmiştir. Bu yazıda, CAV tabanlı trafik kontrol yöntemlerinin kavşak ve otoyol kontrolü, CAV sürüş davranışları ve karar süreçleri, enerji yönetimi, takip mesafeleri, kapalı otopark yönlendirme ve toplu taşıma entegrasyonu gibi alanlardaki çalışmalar özetlenmiştir. Ancak, daha sistematik ve bütünleşik yöntemlerin geliştirilmesine ihtiyaç olduğu, özellikle araç kontrol seviyesinden ağ ve bölgesel seviyelere kadar kapsamlı yaklaşımlar gerektirdiği vurgulanmaktadır. Ayrıca, güvenli iletişim altyapısı, gizlilik ve güvenlik gibi konularda multidisipliner bir iş birliğine duyulan gereksinim de ifade edilmiştir. Son olarak, simülasyon ve kapalı alan testlerinin gerçek yol koşullarına taşınmasının, bu teknolojilerin yaygın uygulamaya geçişini hızlandırmada kritik olduğu kaydedilmiştir (Ban ve diğ. 2020).

2020 yılında yayımlanan bir çalışmada, yol ile araç arasındaki iletişim kullanılarak kavşak güvenliğini artırma üzerine yoğunlaşılmıştır. Kavşaklar, yayalar, bisikletler ve araçlar arasındaki karmaşık etkileşimler, şerit işaretlerinin eksikliği, yol hakkının belirlenememesi, araçların görünmeden yaklaşmaları ve hatalı davranışlar nedeniyle tehlikeli alanlar olarak tanımlanmıştır. Çalışma, bu zorlukların yalnızca yol tasarımı veya *"Vision Zero"* projeleriyle ya da sensörlerle donatılmış otonom araçlarla tamamen çözülemeyeceğini vurgulamaktadır. Sürücüler, bisikletliler ve yayalar, yanlış kararlar almalarına yol açabilecek eksik bilgilere sahip olduklarından, bu

bilgilerin akıllı kavşak altyapısı ile sağlanabileceği önerilmiştir. Bu akıllı kavşak sistemi, mevcut sinyal fazlarını, faz değişim süre tahminlerini ve sürücü veya otonom araç için kör noktaların doluluk bilgilerini iletmeyi amaçlamaktadır. Çalışma, Arizona'da gerçekleşmiş bir kaza örneğinden hareketle, bu tür bir akıllı kavşak altyapısının nasıl geliştirilebileceğini detaylandırmaktadır (Grembek ve diğ. 2019).

2021 yılında yayımlanan bir çalışmada, akıllı şehirlerin ihtiyaçlarına uygun yeni bir TYS önerilmektedir. Geleneksel üç renkli trafik ışıklarının, özellikle çevresel koşullara karşı duyarlılık ve acil durum araçlarına öncelik tanıyamama gibi zayıf yönleri ele alınmıştır. Araştırmacılar, bu sorunlara çözüm olarak Araçsal Tasarsız Ağlar (*Vehicular ad hoc network*, VANET) ve Nesnelerin İnterneti (*Internet of Things*) teknolojilerinden yararlanarak araçların çevreleriyle kablosuz iletişim kurabileceği bir sistem geliştirmiştir. Önerilen Akıllı TYS ve Akıllı Trafik Sinyal denetleyicisi, yerel trafik yönetimini optimize ederek trafik akışında adalet, bekleme süresinin azalması, trafik sıkışıklığının hafifletilmesi ve acil durum araçlarına öncelik sağlama hedefleriyle tasarlanmıştır. Simülasyon sonuçları, bu yeni sistemin geleneksel yönetim sistemlerine kıyasla daha etkili olduğunu ve gelecekteki akıllı şehirlerde kullanım için potansiyel taşıdığını göstermektedir (Elsagheer Mohamed ve Alshalfan 2021).

2022 yılında Boğaziçi Üniversitesinde yapılan bir araştırmada, trafik yönetimi için bağlantılı otonom araçları (CAV) kullanarak yeni bir yöntem olan Şok Dalga Yumuşatmasıyla CAV Metodunu (*"Shockwave Smoothing CAV Method"*) öneriyor. CAV'ların, geleneksel şerit kontrol sinyalleri ve değişken hız sınırları sistemlerine alternatif olarak trafik kontrol aktüatörleri olarak kullanımı incelenmiştir. Geliştirilen bu metot, bir olay nedeniyle oluşan şok dalgası tespit edildiğinde, CAV'ların dalganın hızına kadar yavaşlamasını sağlayarak kuyrukların büyümesini önlemeyi amaçlar. SUMO simülasyon ortamında yapılan 4800 farklı senaryoda, önerilen metodun trafik yoğunluğunu %12,68'e kadar azalttığı ve düşük CAV penetrasyon oranlarında bile şerit kontrol sinyalleri sistemine kıyasla olay bölgesinde trafik yoğunluğunu iki kat daha fazla düşürdüğü gözlemlenmiştir (Gökaşar ve diğ. 2023).

1.2.4 MCS Üzerine Yapılan Araştırmalar

2019'da yayımlanan bir çalışmada, kent içi taşımacılık için öngörülen yeni bir sistem olan MCS tanıtılmıştır. MCS, kişisel araçlarla metro sistemlerini birleştiren bir ulaşım konsepti sunmaktadır: Yolcular, MCS'ye özgü metro tipi yollarda araçlarının otonom olarak sürülmesini sağlar, ardından diğer yollarda manuel sürüşe geçer. Küçük, tek kişilik ve tamamen elektrikli olan bu sistem araçları, MCS altyapısında dış kaynaklı enerjiyle çalışırken, diğer yollarda batarya ile devam eder. Bu sistem için, mevcut trafik sıkışıklıklarını azaltarak şehir içi hareketliliği artıran ve enerji tüketimini düşük tutacak şekilde özel, dar ve seviyeli yollar önerilmektedir. MCS, şehirlerin ana arterlerinde yeni bir altyapı gerektirirken, havaalanları, üniversiteler ve alışveriş merkezleri gibi geniş özel alanlarda da kullanılabilecek bir çözüm sunar. Bu, tamamen otonom araçlara geçiş öncesi, trafik kontrolünü basitleştiren ve ulaşım güvenliğini artıran ara bir çözüm olarak sunulmuştur. *NetLogo* yazılımı kullanılarak çoklu ajan modeli ile doğrulanan bu sistem, şehir içi ulaşımın geleceğine dair umut verici bir yaklaşım olarak görülmektedir (Bozuyla ve Tola 2018).

2019 yılında yayımlanan bir çalışmada, MCS araçları için güvenli ve verimli akışı sağlamak amacıyla geliştirilmiş takip ve birleşme algoritmaları sunulmuştur. Ana ve tali yol senaryosu üzerine kurgulanan algoritma, *NetLogo* simülasyon platformunda test edilmiştir. Çeşitli araç hızlarında trafik akışının kolayca kontrol edilebildiği bu algoritmada, tali yoldan ana yola katılım yapan araçlar normal dağılıma göre düzenlenmiştir. Simülasyon sonuçları, önerilen algoritmanın birleşme bölgesi de dahil olmak üzere herhangi bir çarpışma riski olmadan güvenli bir birleşme sağladığını göstermiştir. Çalışmanın başarılı sonuçları, MCS algoritmalarına yönelik gelecekteki araştırmalar için de umut vadetmektedir (Bozuyla ve diğ. 2018).

2021 yılında yayımlanan bir çalışmada, özellikle mega kentlerdeki ulaşım sorunlarına çözüm üretmek amacıyla geliştirilen yeni bir çift modlu ulaşım sistemi olan MCS tanıtılmıştır. Önerilen sistem, modern ve geleneksel ulaşım sistemlerinin bazı avantajlarını birleştirerek daha etkili, sağlam ve güvenli bir ulaşım modeli oluşturmayı hedeflemektedir. MCS, çok etmenli bir yaklaşımla geliştirilmiş mikroskobik bir trafik simülasyon modeli olarak tasarlanmış ve *NetLogo* simülasyon ortamında test edilmiştir. Sistem kapsamında giriş, takip, birleşme, yönlendirme ve

çıkış gibi süreçler detaylı şekilde tanımlanmış ve dağıtık bir yapı içinde üç farklı etmen tarafından yürütülmüştür. Simülasyon sonuçları, önerilen modelin beklentileri karşıladığını ve trafik akışının çarpışma olmaksızın kontrol edilebildiğini göstermiştir. Bu doğrultuda, MCS'nin geleneksel ulaşım sistemlerine kıyasla daha pratik, ileri seviye ulaşım sistemlerine göre ise daha uygulanabilir bir alternatif sunduğu ortaya konmuştur (Bozuyla ve Tola 2021).

2022 yılında yayımlanan bir çalışmada, son yirmi yılda giderek artan akıllı şehir araştırmaları ve çözümleri ele alınmıştır. Akıllı şehir felsefesinin önemli bir bileşeni olan akıllı mobilite kapsamında, trafik sıkışıklığına çözüm getiren ulaşım sistemlerinin önemi artmıştır. Bu bağlamda, trafik yoğunluğunu önlemek amacıyla geliştirilen yeni çift modlu ulaşım sistemlerinden biri olan MCS incelenmiştir. Çalışmada, MCS'nin *Autoshuttle, TriTrack System, Loop Project, GTS Foundation, BiModal Glideway, CarTube* ve *Rapid Urban Flexible* gibi diğer çift modlu ulaşım sistemleriyle karşılaştırılması yapılmış ve avantajları değerlendirilmiştir. Ayrıca, MCS'nin altyapısı ayrıntılı olarak tanıtılmış ve Denizli kenti için hazırlanan örnek bir proje sunulmuştur. Son olarak, mikroskobik trafik modeli kullanılarak, diğer çift modlu ulaşım sistemlerinde ele alınmayan bir şerit değiştirme problemine yönelik özgün bir çözüm önerilmiştir. Simülasyon sonuçları, MCS'nin diğer sistemlere kıyasla daha güvenli ve uygulanabilir olduğunu, ancak maliyet açısından daha yüksek olduğunu göstermiştir (Bozuyla ve Tola 2022).

1.3 Yöntem ve Materyaller

Bu tez çalışmasında, MCS için takip ve birleşme algoritmalarının performansını artırmayı hedefleyen bir metodoloji izlenmiştir. İlk olarak, sistemin mevcut yapısı ve algoritmaları detaylı bir şekilde incelenmiş, ardından sistemin ihtiyaçları doğrultusunda yenilikçi algoritmalar geliştirilmiş ve gerekli materyallerle test edilmiştir. Çalışma aşağıdaki adımları kapsamaktadır:

Mevcut Durum Analizi

- MCS'de kullanılan mevcut takip ve birleşme algoritmaları analiz edilerek performans sınırlamaları belirlenmiştir. Özellikle, düz yol

takip algoritmasının hız dalgalanmalarına karşı duyarlılığı ve birleşme algoritmasının etkinliği incelenmiştir.

Yeni Algoritmaların Tasarımı

- Takip Algoritması: Araçların hızlarını korumasını sağlamak amacıyla alçak geçiren filtre ve histerezis gibi ek yöntemlere ihtiyaç duymayan bir hız kontrol algoritması tasarlanmıştır.
- Birleşme Algoritması: Mevcut birleşme algoritması iyileştirilmiş, alternatif bir birleşme algoritması önerilmiştir.

Simülasyon Ortamının Oluşturulması

- Python tabanlı bir simülasyon ortamı geliştirilerek algoritmaların test edilmesi sağlanmıştır. Bu simülasyon ortamında araçların davranışlarını modellemek için ayrık zamanlı algoritmalar kullanılmıştır.

Performans Değerlendirmesi

- Yeni algoritmalar, mevcut algoritmalar ile performans açısından karşılaştırılmıştır. Bu süreçte kullanılan metrikler arasında hız stabilitesi, çarpışma riski, algoritma verimliliği ve yol kapasitesi yer almaktadır.

Bu tezde gelecek bölümlerde ele alınacak konular aşağıdaki gibidir.

İkinci bölüm, tez ile alakalı bazı temel kavramlar ele alınacaktır. Bu kavramlar Otonom Araçlar, Sürekli Zamanlı ve Ayrık Zamanlı Sistemler, Python, Normal Dağılım ve MCS şeklinde beş ana başlık altında incelenecektir.

Üçüncü bölümde, tez çalışmasında kullanılacak olan materyaller ve yöntemler bahsedilecek ve çalışma bütün hatlarıyla bu bölümde anlatılacaktır. Son olarak çalışmadan elde edile test sonuçları bu bölümde sunulacaktır.

Dördüncü bölüm, tez çalışmasının sonuçlarının ve önerilerinin anlatıldığı son bölümdür.

2. TEMEL KAVRAMLAR

Bu bölümde tezin de temelini oluşturan bazı temel kavramlar tanıtılacaktır. Bu kavramlar Otonom Araçlar, MCS, Sürekli ve Ayrık Zaman Sistemleri, *Python* ve Normal Dağılım olmak üzere beş ana başlıkta incelenecektir.

2.1 Otonom Araçlar

Otonom araçların geliştirilmesi, modern toplumun karşılaştığı birçok ulaşım sorununa çözüm arayışından doğmuştur. Yoğun trafik, sürücü hatalarına bağlı trafik kazaları, çevre kirliliği ve artan enerji tüketimi gibi küresel ölçekte yaşanan problemler, otonom sistemlere olan ilgiyi tetiklemiştir. Özellikle büyük şehirlerdeki ulaşım talebinin artması, trafik sıkışıklığı ve kaza risklerini azaltacak yeni teknolojiler geliştirme ihtiyacını doğurmuştur. Aynı zamanda, sürüş esnasında güvenliği artıracak, verimliliği destekleyecek ve çevresel etkiyi azaltacak sistemlerin varlığı, ulaşımda daha akıllı ve sürdürülebilir çözümler sunulması gerekliliğini ortaya koymuştur. Bu koşullar, insan müdahalesine ihtiyaç duymadan çevreyi algılayabilen ve otonom kararlar alabilen araçların geliştirilmesi için güçlü bir zemin hazırlamıştır.

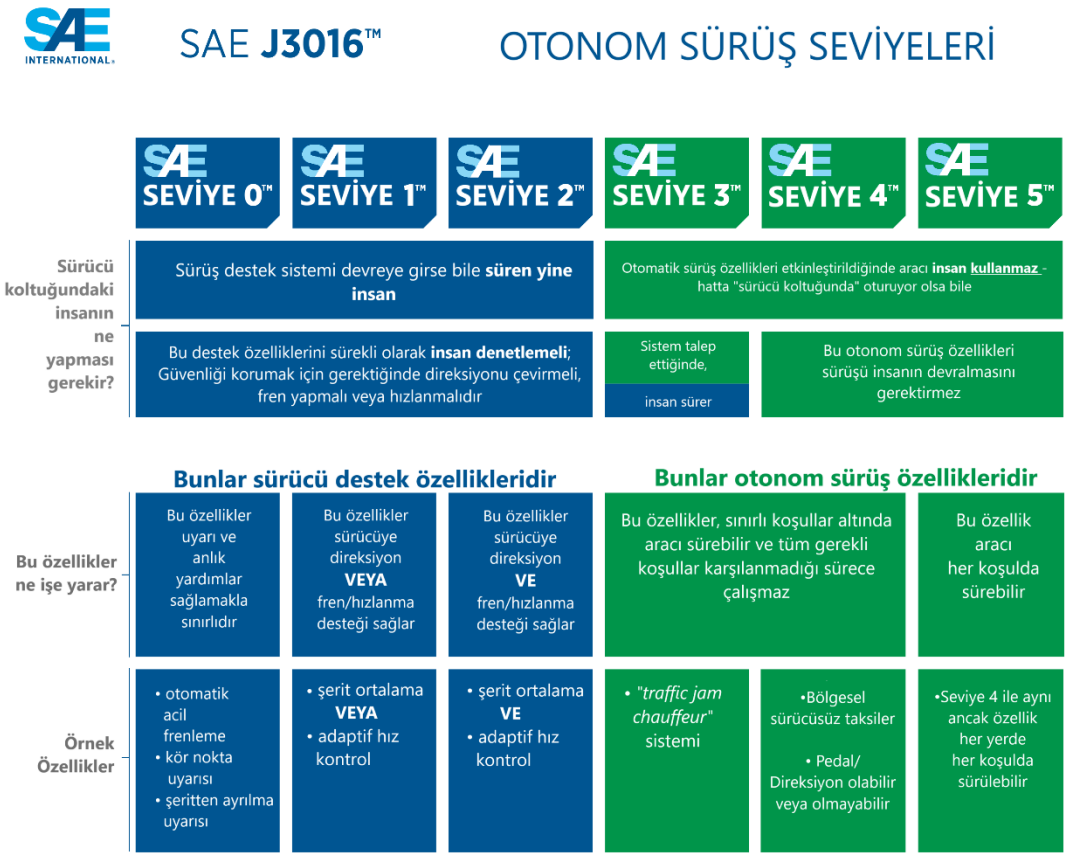
2.1.1 Otonom Araç Tanımı

Otonom araçlar, insan müdahalesine gerek kalmadan çevrelerini algılayarak, analiz yaparak ve karar alarak yol alabilen araçlardır. Sensörler, yapay zekâ, makine öğrenimi ve algoritmalar gibi yöntemler kullanılarak çevredeki nesneleri tanıma, rotayı belirleme, hız ayarlama ve gerektiğinde durma gibi işlemleri bağımsız olarak gerçekleştirebilmektedirler. Otonom araçlar, sürüş görevlerini farklı seviyelerde yerine getirerek çeşitli otonomi derecelerine sahip olabilirler.

Otonom araçlar, ulaşımda güvenliği artırmak, trafik akışını iyileştirmek, yakıt tasarrufu sağlamak ve sürücü hatalarını minimize etmek gibi amaçlarla önem kazanmaktadır. Trafik kazalarının büyük bir bölümü insan hatasından kaynaklandığından, otonom araçlar potansiyel olarak daha güvenli bir sürüş sağlayabilir. Ayrıca, otonom araç teknolojileri engelli veya yaşlı bireylerin bağımsız

hareket edebilmesine imkân tanıyarak toplumsal faydalar da sunar. Enerji verimliliği, emisyonların azaltılması ve akıllı şehir altyapılarının desteklenmesi gibi sürdürülebilirlik hedefleri açısından da otonom araçlar kritik bir rol üstlenmektedir.

2.1.2 Otonom Sürüş Düzeyleri



Şekil 2.1: J3016 SAE uluslararası standardı (SAE 2021[b]).

Otonom sürüş düzeyleri, araçların sürüş işlevlerini hangi derecede bağımsız olarak gerçekleştirebildiğini tanımlayan ve SAE tarafından belirlenen bir sınıflandırma sistemidir. Seviye 0, acil frenleme ve kör nokta uyarısı gibi uyarı ve anlık yardım işlevlerini içerir. Seviye 1 ise şerit ortalama ve adaptif hız sabitleme gibi destekleyici sürüş özelliklerine sahiptir. Tesla'nın *Autopilot* özelliği ve Cadillac'ın *Super Cruise* sistemleri Seviye 2 olarak sınıflandırılmaktadır. Bu seviyede araç, otonom hareket edebilse de sürücünün sürüşü sürekli izlemesi ve gerektiğinde hızlıca kontrolü devralmaya hazır olması gerekir (SAE 2021[b]). Audi A8L'in trafik sıkışıklığı pilotu, otonom sürüş görevini yerine getiremediği durumlarda sürücünün kontrolü

devralmasını gerektiren Seviye 3 otonom sunmaktadır. Seviye 4 ve 5 otomasyon sistemleri, her koşulda aracı kendi başına sürebilir ve herhangi bir sistem arızasında müdahale edebilir. Ancak, bu tam otomasyon seviyeleri önemli altyapı geliştirmeleri gerektirmektedir; bu yüzden Cruise ve Waymo gibi araçlar, otonom sürüş modunda düşük hızda ve sınırlı kentsel alanlarda faaliyet göstermektedir. Örneğin, Cruise, kendi kendine sürüş yapan araçlarının tam veya sınırlı erişim sağladığı coğrafi bölgelerin bir listesini yayımlamıştır (Cruise 2024).

Şekil 2.1'de SAE tarafından yayımlanan otonom araç sınıflandırmasında J3016 uluslararası standardı görülmektedir.

Otonom sürüş düzeyleri, otonom araç teknolojisinin gelişim aşamalarını anlamada önemli bir rehberdir ve gelecekte tam otonom sürüşe ulaşmak için gereken adımları göstermektedir.

2.1.3 Otonom Araç Teknolojileri ve Bileşenleri

Otonom araç teknolojileri ve bileşenleri, araçların çevresini algılayıp yorumlayabilmesi, güvenli ve verimli bir şekilde yol alabilmesi için birbirleriyle entegre çalışan çok sayıda gelişmiş sistemden oluşur. Bu sistemler temel olarak sensörler, veri işleme modülleri, karar alma algoritmaları ve kontrol mekanizmalarını içermektedir. Aşağıda, bu teknolojiler ve bileşenleri ayrıntılı olarak ele alınmıştır.

2.1.3.1 Sensör Sistemleri

Otonom araçlar, çevrelerindeki yol, trafik, yayalar ve diğer engelleri algılayabilmek için çeşitli sensörlerden veri toplar. Otonom araçlarda kullanılan temel sensör türleri şunlardır:

- GNSS (*Global Navigation Satellite System,* Küresel Navigasyon Konumlandırma Sistemi): GNSS, aracın tam konumunu belirleyerek harita üzerinde yerini tespit eder. Otonom araçlarda hassas konumlandırma için GNSS sistemleri, diğer sensörlerle birlikte çalışır. GNSS, özellikle şehir dışı

yollar gibi daha geniş alanlarda konum takibi sağlar ancak kentsel alanlarda binalardan dolayı sinyal zayıflaması yaşanabilir.

- Lidar: Lidar, ışık dalgalarını kullanarak çevredeki nesnelerin mesafesini ölçer ve yüksek doğrulukta bir 3 boyutlu harita oluşturur. Lidar, lazer darbeleriyle gönderdiği sinyallerin nesnelerden yansıma süresini hesaplayarak araç etrafındaki cisimlerin konumunu belirler. Özellikle karmaşık kentsel ortamlarda detaylı bir görüş sağlayarak, araç etrafındaki nesnelerin şekil ve pozisyonlarını doğru bir şekilde tanımlar.
- Radar: Lidar'ın aksine, radar mikrodalgaları kullanarak nesnelerin mesafesini, hızını ve yönünü tespit eder. Radar sensörleri kötü hava koşullarında dahi güvenilir bir görüş sağlayarak, Lidar ve GNSS gibi diğer sensörlere kıyasla daha dayanıklıdır. Otonom araçlarda kısa ve uzun menzilli radarlar, nesne algılama ve çarpışma önleme için kullanılır.
- Kameralar: Görsel veri sağlamak amacıyla kameralar, araçların çevresindeki objeleri algılayarak nesne tanımlama ve şerit takibi gibi işlevlerde kritik rol oynar. Görüntü işleme algoritmaları ile birlikte kullanıldığında kameralar, trafik işaretleri, yaya ve bisiklet gibi çeşitli nesneleri tanımlayabilir. Ancak kameralar aydınlatma koşullarına ve hava şartlarına daha duyarlıdır, bu nedenle Lidar ve radar gibi diğer sensörlerle desteklenir.
- IMU (*Inertial Measurement Unit*, Ataletsel Ölçüm Birimi): IMU, aracın ivmesini, hızını ve yönünü ölçerek aracın hareketini ve yönelimini anlamada kritik bir bileşendir. GNSS ile birlikte çalışarak, aracın konum ve hareket doğruluğunu artırır.

2.1.3.2 Veri İşleme ve Yapay Zekâ

Otonom araçlarda toplanan büyük miktardaki verinin hızlı ve doğru bir şekilde işlenmesi için güçlü bir veri işleme altyapısı gereklidir. Bu altyapı, araç içi bilgisayarlar ve bulut tabanlı işlem gücünden oluşur:

- Algoritmalar ve Makine Öğrenimi: Algoritmalar, sensörlerden gelen veriyi anlamlandırmak, nesneleri tanımlamak ve çevre haritalarını çıkarmak için kullanılır. Makine öğrenimi ve derin öğrenme teknikleri, kameralar ve diğer

sensörlerle birlikte çalışarak nesne algılama, yaya tespiti, hız sınırı ve trafik işareti tanıma gibi işlevleri yerine getirir.

- Çevre Algılama ve Haritalama (SLAM): (*Simultaneous Localization and Mapping*, SLAM), aracın konumunu harita üzerinde güncelleyerek etrafındaki ortamın haritasını oluşturur. Lidar ve kamera gibi sensörler tarafından elde edilen veriler işlenerek aracın çevresi hakkında sürekli güncellenen bir harita oluşturulur.
- Karar Alma ve Planlama Algoritmaları: Karar alma algoritmaları, aracın çevredeki nesnelerle güvenli bir etkileşim içinde yol almasını sağlamak için kullanılır. Bu algoritmalar, aracın hızını ve yönünü düzenleyerek, trafik kurallarına uyma, yayalardan kaçınma ve şerit değiştirme gibi işlemleri gerçekleştirir.
- Eylem Kontrol Modülleri: Eylem kontrol modülleri, aracın hızlanma, frenleme ve direksiyon kontrolünü düzenler. Bu modüller, karar alma algoritmalarının verdiği talimatları fiziksel hareketlere dönüştürerek aracın güvenli ve akıcı bir şekilde hareket etmesini sağlar.

2.1.3.3 İletişim Sistemleri

Otonom araçlar, diğer araçlar, altyapı ve çevredeki kullanıcılarla iletişim kurarak daha güvenli ve verimli bir sürüş deneyimi sağlar. Bu iletişim sistemleri, araçtan araca (*Vehicle to Vehicle,* V2V) ve ayrıca araçtan altyapıya (*Vehicle to Infrastructure,* V2I) iletişim protokollerini içerir:

- V2V İletişimi: Araçtan araca iletişim, araçların birbirleriyle hız, konum ve yön bilgisi gibi verileri paylaşmasını sağlar. Bu, çarpışmaların önlenmesi ve trafik akışının optimize edilmesi için önemlidir.
- V2I İletişimi: Araçtan altyapıya iletişim, araçların trafik ışıkları, yol işaretleri ve diğer altyapı unsurlarıyla iletişim kurmasını sağlar. Bu sayede otonom araçlar trafik ışıklarının durumu, yol durumu gibi bilgilere erişebilir.
- 5G ve DSRC Teknolojileri: Hızlı ve kesintisiz veri akışı sağlamak için 5G ve DSRC (*Dedicated Short-Range Communications*) gibi teknolojiler kullanılır.

Bu iletişim teknolojileri, gerçek zamanlı veri alışverişini mümkün kılarak otonom araçların güvenli sürüş kabiliyetini artırır.

2.1.3.4 Güvenlik ve Hata Tolerans Sistemleri

Otonom araçlar, güvenli sürüş için hata toleransı ve siber güvenlik sistemlerine de sahiptir.

- Yedek Sistemler: Sistem hataları durumunda aracın güvenli bir şekilde çalışmaya devam etmesi için yedek sensörler ve işlemciler kullanılır. Bu, sistemde bir hata oluştuğunda aracın kendini korumasını sağlar.
- Siber Güvenlik: Otonom araçlar, veri ihlalleri ve kötü amaçlı saldırılardan korunmak için siber güvenlik önlemleriyle donatılması gerekmektedir. Araç içi ağlar ve iletişim protokolleri şifrelenerek dış müdahalelere karşı koruma sağlanır.

Bu bileşenlerin tümü, otonom araçların çevresel veriyi toplaması, analiz etmesi ve güvenli bir şekilde hareket etmesi için bir arada çalışır. Teknolojinin gelişimiyle birlikte, bu sistemlerin doğruluk, güvenilirlik ve güvenlik düzeylerinin artırılması gerekmektedir.

2.1.3.5 Haritalama ve Konumlandırma

Otonom araçlar, doğru bir yol tarifi ve çevre haritası elde etmek için çeşitli haritalama ve konumlandırma sistemlerini kullanır:

- HD Haritalar: Yüksek çözünürlüklü haritalar, standart dijital haritalardan daha ayrıntılı bilgiler içerir ve otonom araçların yoldaki her detayı daha hassas bir şekilde algılamasını sağlar. Bu haritalar, şerit genişliği, yol eğimi ve yaya geçitleri gibi verileri içerir.
- Gerçek Zamanlı Güncelleme Sistemleri: Haritalar, trafik akışı ve çevre koşullarındaki değişimlere uyum sağlamak için gerçek zamanlı olarak

güncellenir. Bu güncellemeler, aracın hareketini çevredeki dinamik unsurlara göre ayarlamasına yardımcı olur.

2.1.4 Şehir İçi Otonom Araç Türleri ve Uygulama Alanları

Şehir içi ulaşım, lojistik ve altyapı bakımında kullanılan otonom araçlar, farklı işlevlere göre sınıflandırılmaktadır. Bu bölümde, yolcu taşımacılığı, yük ve paket dağıtımı ile temizlik ve bakım hizmetlerinde kullanılan otonom araç türleri incelenmektedir.

2.1.4.1 Kentsel Alanlarda Kullanılan Otonom Yolcu Araçları

Genellikle otonom taksi, minibüs ve yolcu otobüsü olarak sınıflandırılan bu araçlar, yolcu kapasitesi, sürüş hızı ve güzergâh planlama gibi özelliklere göre çeşitlenir. Yolcu taşımacılığında otonom sistemler, özellikle kent merkezlerinde sık görülen trafik sıkışıklığını azaltma ve karbon emisyonunu düşürme potansiyeline sahiptir. Bu araçların kullanımında güvenlik, yolcu memnuniyeti ve verimlilik gibi temel performans ölçütleri incelenir. Amerika'da Tesla, Şekil 2.2'de görselleri verilen Cybercab ve Cybervan gibi otonom sistemlerle şehir içinde tamamen sürücüsüz bir deneyimi 2026'da piyasaya sürmeyi planlamaktadır.

Şekil 2.2: Tesla Cybervan ve Cybercab.

2.1.4.2 Şehir İçi Yük ve Paket Dağıtımında Kullanılan Otonom Araçlar

Otonom dağıtım araçları, paket taşımacılığında insan faktörünü en aza indirerek hızlı ve etkili bir lojistik çözümü sunar. Dronlar, kargo robotları ve küçük ölçekli dağıtım araçları bu gruba dâhildir. Şehir planlamasında bu tür araçların güzergâhları, enerji tüketimi ve teslimat verimliliği gibi metrikler, sistemin işlevselliğini doğrudan etkiler.

2.1.4.3 Şehir İçi Temizlik ve Bakım Araçları Olarak Otonom Sistemler

Kentsel yaşam alanlarının temizliği ve altyapı bakımı için kullanılan otonom araçlar, belediyeler ve kamu kuruluşları tarafından tercih edilmektedir. Şekil 2.3'te verilen bu tip otonom araçlar, cadde temizliği, çöp toplama, park düzenlemeleri ve kar küreme gibi rutin bakım ve temizlik işlerinde insan gücünü azaltarak kentsel sürdürülebilirliği desteklemektedir.

Şekil 2.3: IdriverPlus otonom süpürge aracı.

2.2 Sürekli Zamanlı ve Ayrık Zamanlı Sistemleri

Sürekli zaman ve ayrık zaman sistemleri, özellikle dijital ve analog sistemlerdeki modelleme, analiz ve tasarım süreçlerinde önemli bir yere sahiptir.

Sürekli zamanlı ve ayrık zamanlı sistemlerin temel özellikleri, avantajları ve sınırlamaları detaylı bir şekilde ele alındıktan sonra, bu sistemlerin gerçek dünyadaki fiziksel büyüklükleri (örneğin, hız ve ivme) nasıl modellediği üzerinde durmak gereklidir. Sürekli zaman sistemlerinde hız ve ivme gibi büyüklükler türev ve integral kavramlarıyla ifade edilirken, ayrık zaman sistemlerinde fark denklemleri ve ayrık türevlerle modellenir. Aşağıdaki bölümlerde bu denklemlerin matematiksel ifadeleri ve temel uygulama örnekleri ele alınacaktır.

2.2.1 Sürekli Zaman Sistemleri

Sürekli zaman sistemleri, zamana bağlı bir sürecin tüm zaman boyunca kesintisiz olarak tanımlandığı sistemlerdir. Bu sistemlerde zaman değişkeni *t*, genellikle reel sayılar kümesinden (**ℝ**) alınır ve bu nedenle zaman ekseni üzerinde sonsuz sayıda değer alabilir. Sürekli zaman sistemleri, fiziksel dünyadaki birçok sürecin matematiksel olarak modellenmesi için doğal bir temel oluşturur.

2.2.1.1 Matematiksel modelleme

Sürekli zaman kavramı, fiziksel süreçlerin modellenmesinde analog bir yaklaşımdır. Bir odadaki sıcaklık değişimi, bir elektrik devresindeki akım veya voltaj dalgalanması, ya da bir cismin hareketi sürekli zaman sistemleri ile ifade edilebilir.

$$\frac{d^n y(t)}{dt^n} + a_{n-1}\frac{d^{n-1} y(t)}{dt^{n-1}} + \cdots + a_0 y(t) = b_m \frac{d^m x(t)}{dt^m} + \cdots + b_0 x(t) \qquad (2.1)$$

Yukarıda ifade edilen Denklem (2.1)'de:

- $y(t)$, sistemin çıkış sinyalini,
- $x(t)$, sistemin giriş sinyalini,
- a_i ve b_j $(i = 0, 1 \ldots n)(j = 0, 1 \ldots m)$, sistemin parametrelerini ifade eder.

Diferansiyel denklemler ve integraller, sürekli zaman sistemlerinin dinamiklerini analiz etmek için kullanılır. Örneğin, bir mekanik sistemde yay ve kütlenin davranışı

ya da bir elektrik devresindeki RLC elemanlarının etkileşimi bu tür denklemlerle modellenir.

2.2.1.2 Temel Özellikler

Kesintisizlik: Sinyaller zamanın her anında tanımlıdır ve kesintisiz bir yapıya sahiptir.

Frekans Alanı Analizi: Fourier ve Laplace dönüşümleriyle analiz yapılabilir.

Enerji ve Güç Hesaplama: Sürekli zaman sinyallerinin enerji ve güç değerleri integral hesaplarıyla belirlenir.

Doğrusal ve Doğrusal Olmayan Sistemler: Hem doğrusal hem de doğrusal olmayan sistemler modellenebilir.

Sonsuz Çözünürlük: Zaman ekseni üzerindeki tüm değerler kullanılabilir.

Analog Doğa: Fiziksel dünyadaki birçok süreçle uyumludur.

Matematiksel Temel: Diferansiyel denklemlerle ifade edilir.

2.2.1.3 Avantajlar ve Sınırlamalar

Avantajları:

- Doğal Süreçlerle Uyumlu: Sürekli zaman sistemleri, fiziksel süreçleri doğal olarak modelleyebildiği için mühendislik ve bilimsel çalışmalarda sıkça kullanılır.
- Kesintisiz Analiz: Zamanın her anında tanımlı sinyaller, sürekli analiz yapılmasını sağlar.
- Geniş Uygulama Alanı: Elektrik mühendisliği, mekanik, termodinamik gibi birçok alanda kullanılabilir.

Sınırlamaları:

- Dijital İşlemeye Uygun Değil: Sürekli zaman sistemleri dijital ortamda doğrudan işlenemez. Dijital işleme için bu sistemlerin ayrık zamana dönüştürülmesi gerekir.
- Matematiksel Karmaşıklık: Diferansiyel denklemlerle analiz ve çözümleme, özellikle doğrusal olmayan sistemlerde oldukça karmaşık olabilir.

2.2.1.4 Hareket Denklemleri

Hız (v), ivme (a), yol (x) ve zaman (t) arasındaki ilişkiler genellikle türev ve integral kullanılarak ifade edilir. Aşağıda bu terimler arasındaki ilişkileri temsil eden bazı temel formüller bulunmaktadır:

Hız (v) değişkeninin yol – zaman – ivme değişkenleri ile ilişkisi:

$$v(t) = v(t_0) + \int_{t_0}^{t} a(\tau)d\tau \tag{2.2}$$

$$v(t) = \frac{dx(t)}{dt} \tag{2.3}$$

İvme (a) değişkeninin yol – zaman – hız değişkenleri ile ilişkisi:

$$a(t) = \frac{dv(t)}{dt} \tag{2.4}$$

$$a(t) = \frac{d^2x(t)}{dt^2} \tag{2.5}$$

Yol (x) değişkeninin ivme – zaman – hız değişkenleri ile ilişkisi:

$$x(t) = x(t_0) + \int_{t_0}^{t} v(\tau)d\tau \tag{2.6}$$

$$x(t) = x(t_0) + \iint_{t_0}^{t} a(\tau)d\tau \tag{2.7}$$

2.2.2 Ayrık Zaman Sistemleri

Ayrık zaman sistemleri, sinyallerin ve olayların yalnızca belirli zaman anlarında tanımlandığı bir yapıyı ifade eder. Sürekli zaman sistemlerinin aksine, ayrık

zaman sistemlerinde sinyaller belirli aralıklarla örneklenir ve dijital sistemlerde sıklıkla kullanılır. Bu sistemlerin temel kavramları, avantajları ve analiz yöntemleri aşağıda ayrıntılı olarak ele alınmıştır.

Ayrık zaman sistemleri, bir sinyalin sürekli bir zaman ekseni yerine yalnızca belirli zaman noktalarında tanımlandığı sistemlerdir. Bu sinyaller genellikle $x[n]$ formunda ifade edilir ve n, tam sayılar kümesinde yer alır.

Örneğin:

- Ses sinyalinin dijital bir cihazda kaydedilmesi.
- Bir finansal zaman serisinin günlük olarak izlenmesi.

Ayrık zaman sinyalleri genellikle sürekli zaman sinyallerinin örneklenmesiyle elde edilir. Bu süreç, belirli bir örnekleme frekansı ile gerçekleştirilir.

2.2.2.1 Matematiksel Modelleme

Ayrık zaman sistemlerinin matematiksel temelleri fark denklemleri ve ayrık sinyal işlemleri üzerine kuruludur:

- Fark Denklemleri: Ayrık zaman sistemleri, sürekli zaman sistemlerinde kullanılan diferansiyel denklemlerin ayrık bir karşılığı olan fark denklemleriyle modellenir.

  Örneğin:

$$y[n] = a_1 \cdot y[n-1] + a_2 \cdot y[n-2] + b_1 \cdot x[n] + b_2 \cdot x[n-1] \qquad (2.8)$$

Denklem (2.8), sistemin geçmiş ve mevcut girişlerine dayanarak çıkışını hesaplar.

- Örnekleme Süreci: Sürekli bir sinyalin örneklenmesi, *Nyquist* oranına uygun şekilde yapılmalıdır. Örnekleme frekansı (f_s) sinyaldeki en yüksek frekansın (f_{max}) Denklem (2.9) gibi iki katından fazla olmalıdır. Aksi taktirde katlanma

(*Aliasing*) adı verilen bir bozulma meydana gelir. Bu durumda sinyal, yanlış frekans bileşenleri içerir ve yeniden doğru bir şekilde oluşturulamaz.

$$f_s > 2\, f_{max} \tag{2.9}$$

2.2.2.2 Temel Özellikler

Ayrıklık: Sinyaller yalnızca belirli zaman anlarında tanımlıdır ve aradaki değerler göz önünde bulundurulmaz.

Frekans Alanı Analizi: Z-dönüşümü ve Ayrık Fourier Dönüşümü kullanılarak analiz yapılabilir.

Enerji ve Güç Hesaplama: Ayrık zaman sinyallerinin enerji ve güç değerleri toplamlarla hesaplanır.

Doğrusal ve Doğrusal Olmayan Sistemler: Ayrık zamanlı doğrusal ve doğrusal olmayan sistemler modellenebilir.

Sonlu Çözünürlük: Sinyaller yalnızca belirli zaman anlarında tanımlandığı için çözünürlük örnekleme frekansına bağlıdır.

Dijital Doğa: Fiziksel süreçlerin dijital temsilinde kullanılır ve dijital sistemlerle uyumludur.

Matematiksel Temel: Fark denklemleriyle ifade edilir ve analiz edilir.

2.2.2.3 Avantajlar ve Sınırlamalar

Avantajlar:

- Depolama ve İşleme Kolaylığı: Dijital sistemlerde saklama ve işleme daha kolaydır.
- Hata Toleransı: Dijital sinyaller, analog sistemlere kıyasla daha az parazit ve bozulma gösterir.

- Uyarlanabilirlik: Ayrık zaman sistemleri, modern yazılım ve donanım araçlarıyla kolayca uyarlanabilir.

Dezavantajlar:

- Bilgi Kaybı: Örnekleme sırasında sürekli sinyalin bazı bilgilerinin kaybolma riski vardır.
- Ekstra İşlem Gereksinimi: Örnekleme ve yeniden yapılandırma işlemleri gibi ek hesaplama gerekliliği doğurur.

2.2.2.4 Hareket Denklemleri

Ayrık zaman terimi, birbirini takip eden belirli zaman aralıklarında ölçülen değerlerin kullanıldığı bir kavramdır. Bu durumda hız (v), ivme (a), yol (x) ve zaman (t) terimleri arasındaki ilişkileri formülle göstermek için aşağıdaki ilişki kullanılabilir:

Hız (v) değişkeninin yol – zaman – ivme değişkenleri ile ilişkisi:

$$v[n+1] = v[n] + a[n] \cdot \tau \tag{2.10}$$

$$v[n+1] = \frac{x[n+1]-x[n]}{\tau} \tag{2.11}$$

İvme (a) değişkeninin yol – zaman – hız değişkenleri ile ilişkisi:

$$a[n+1] = \frac{v[n+1]-v[n]}{\tau} \tag{2.12}$$

$$a[n+1] = \frac{2.(x[n+1]-x[n\,]) - v[n]\cdot\tau}{\tau^2} \tag{2.13}$$

Yol (x) değişkeninin ivme – zaman – hız değişkenleri ile ilişkisi:

$$x[n+1] = x[n] + \ v[n] \cdot \tau \tag{2.14}$$

2.3 Python

Python, yazılım geliştirme ve bilimsel hesaplamada geniş bir uygulama alanına sahip, esnek ve güçlü bir programlama dilidir. Bu bölümde, *Python*'ın temel özellikleri, yapıları ve algoritmalar üzerindeki etkisi ele alınacaktır. *Python*'ın açık kaynaklı yapısı, kullanıcı dostu olması ve geniş kütüphane desteği, bilimsel araştırmalardan günlük yazılım geliştirme projelerine kadar birçok alanda tercih

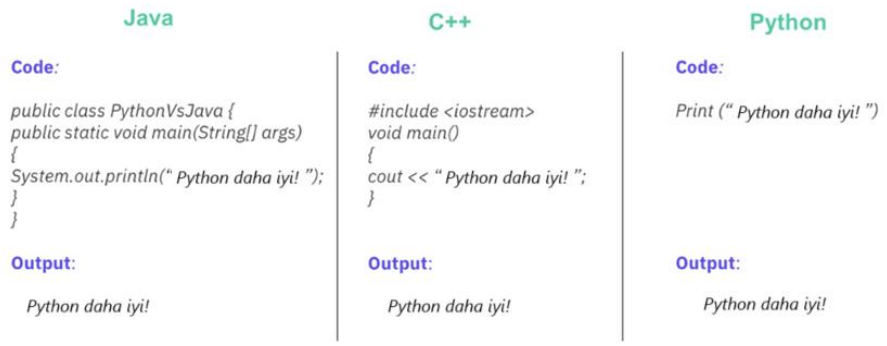
edilmesini sağlar. *Python*'ın sağladığı bu avantajlar, simülasyonlar ve ayrık zamanlı algoritmaların uygulanması gibi konularda oldukça kullanışlıdır.

2.3.1 Python'un Genel Tanımı ve Özellikleri

Python, 1991 yılında Guido van Rossum tarafından geliştirilen, nesne yönelimli, yüksek seviyeli ve genel amaçlı bir programlama dilidir. Dilin genel özellikleri şu şekilde özetlenebilir:

2.3.1.1 Kolay Öğrenilebilirlik

Python, okunabilir, anlaşılır ve sade bir sözdizimine sahiptir. Bu özellik, programlamaya yeni başlayanlar için ideal bir ortam sağlar. Şekil 2.4'te *Java, C++* ve *Python* karşılaştırılarak *Python*'ın hem yazılış biçiminden hem de anlaşılma açısından daha tercih edilebilir olduğu görülmektedir.



Şekil 2.4: Java, C++ ve Python karşılaştırması.

2.3.1.2 Açık Kaynak ve Çok Platformlu Yapı

Python açık kaynaklıdır, bu da herkesin kaynak kodunu inceleyebilmesine ve geliştirebilmesine olanak tanır. Ayrıca, *Windows, macOS* ve *Linux* gibi birçok platformda çalışmaktadır.

2.3.1.3 Geniş Kütüphane Desteği

Bilimsel hesaplama (*NumPy*, *SciPy*), veri analitiği (*Pandas*), yapay zekâ (*TensorFlow*, *PyTorch*), görselleştirme (*Matplotlib*, *Seaborn*) gibi birçok alanda kullanılabilen zengin bir kütüphane koleksiyonuna sahiptir. Şekil 2.5'te en çok bilinen kütüphanelerden bir kısmı bulunmaktadır.

Şekil 2.5: Python kütüphaneleri (Rodrigues 2019).

2.3.1.4 Dinamik Tip Sistemine Sahip Olması

Python, değişkenlerin tiplerini otomatik olarak belirler. Şekil 2.6'daki gibi üst üste aynı değişken üzerine tanımlama yapılabilmektedir. Bu özellik, kod geliştirme sürecini hızlandırır ve kullanıcıya esneklik sağlar.

```
X=10
Print(x)
X=''Hello World"
Print(x)
```

(a)

```
10
Hello World
```

(b)

Şekil 2.6: Dinamik veri tipi a) girdi b) çıktı.

2.3.1.5 Nesne Yönelimli ve İşlevsel Programlama Desteği

Python, nesne yönelimli ve işlevsel programlamayı bir arada destekleyen bir dil olarak öne çıkar. Bu esneklik, algoritma tasarımı ve geliştirme süreçlerinde büyük bir avantaj sağlar.

2.3.1.6 Topluluk Desteği ve Belgelendirme

Python, geniş bir kullanıcı topluluğuna sahiptir. Bu topluluk, *Python*'un geliştirilmesine katkıda bulunurken, kullanıcılar için kapsamlı bir dokümantasyon ve kaynak sağlamaktadır.

2.3.1.7 Hızlı Prototip Geliştirme

Python, yüksek seviyeli yapısı sayesinde hızlı prototip geliştirmeyi mümkün kılar. Bu da özellikle algoritmaların test edilmesi ve simülasyonlar için ideal bir ortam sağlar.

Bu temel özellikler, *Python*'ın modern yazılım geliştirme ve bilimsel araştırmalardaki önemini artırmaktadır. İlerleyen bölümlerde *Python*'ın yapıları ve bilimsel algoritmalar üzerindeki etkisi detaylandırılacaktır.

2.3.2 Python'un Temel Yapıları

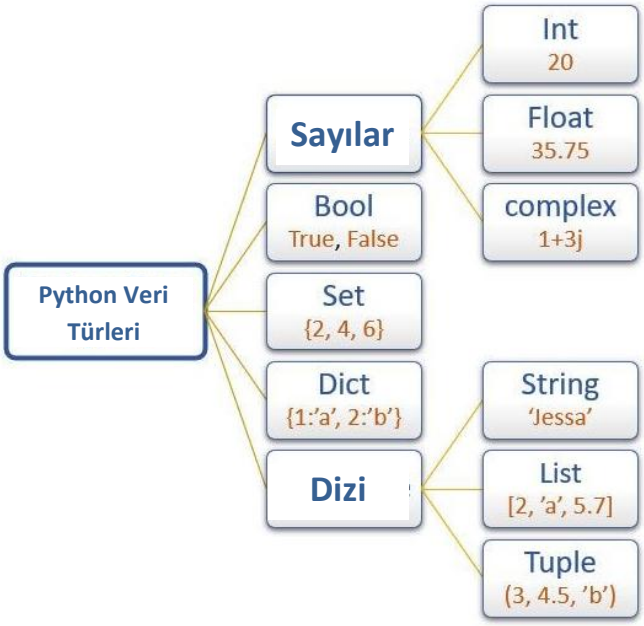
Python, kullanıcı dostu ve esnek yapısıyla temel programlama kavramlarını kolayca uygulamayı mümkün kılar. *Python*'ın temel yapıları hem basit hem de karmaşık algoritmaların geliştirilmesine olanak sağlar. Bu yapılar aşağıdaki gibi sınıflandırılabilir:

2.3.2.1 Değişkenler ve Veri Türleri

Python, dinamik tip sistemine sahiptir ve değişkenlerin tipleri otomatik olarak belirlenir. Şekil 2.7'de görsel olarak verilen temel veri türleri ise şunlardır:

- Tamsayılar (*int*): Pozitif veya negatif tam sayılar.
- Ondalıklı Sayılar (*float*): Ondalıklı ve reel sayılar.
- Karmaşık Sayılar (complex): Reel ve imajiner barındıran sayılar.
- Metinler (*str*): Karakterlerden oluşan dizeler.
- Mantıksal (*bool*): Doğru (*True*) veya yanlış (*False*) değerlerini ifade eder.

- Liste (*list*), Demet (*tuple*) ve Sözlük (*dict*): Veri yapıları ile karmaşık veriler saklanabilir.



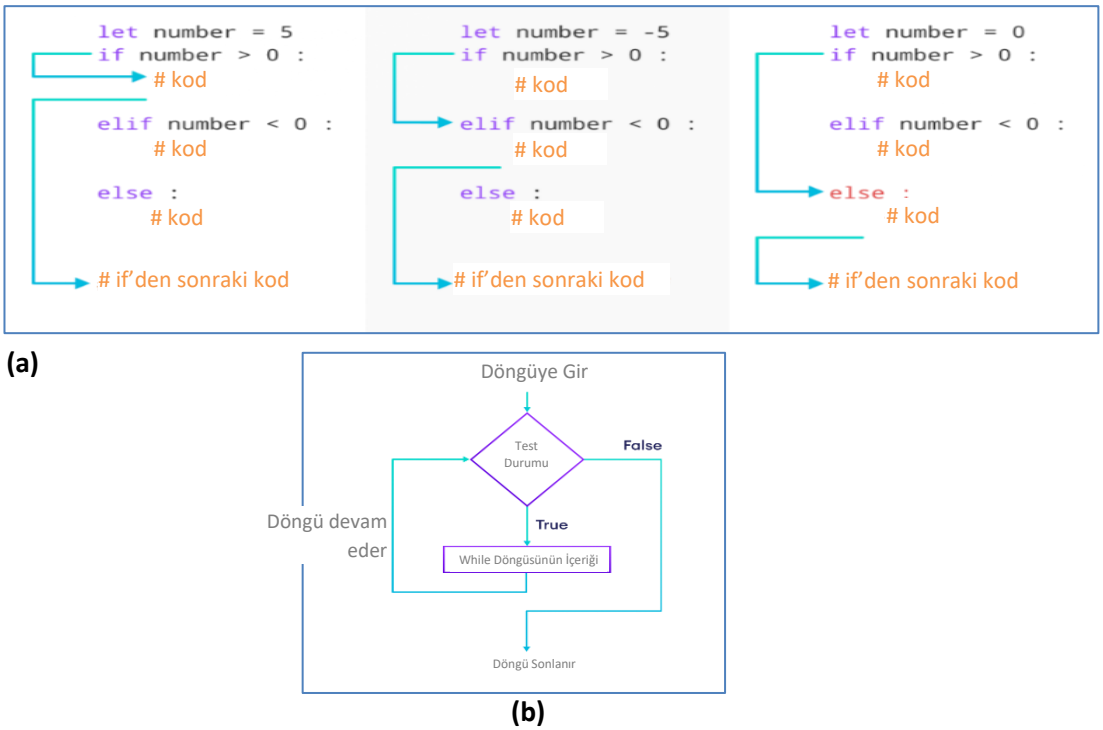
Şekil 2.7: Python veri türleri.

2.3.2.2 Kontrol Yapıları

Python, kontrol akışını yönetmek için temel yapılara sahiptir:

- Koşul Yapıları (*if, elif, else*): Belirli bir koşula bağlı olarak kodun belirli bölümlerinin çalıştırılmasını sağlar. Bu sayede program, farklı durumlara dinamik olarak tepki verebilir.
- Döngüler (*for, while*): Bir işlemin belirli bir sayı kadar veya belirli bir koşul sağlandığı sürece tekrarlanmasını sağlar. Bu yapı, özellikle veri işleme ve otomasyon görevlerinde büyük kolaylık sunar.

Şekil 2.8'de bu döngülerin işleyiş biçimleri ve nasıl kullanıldıkları görsel olarak ifade edilmiştir.

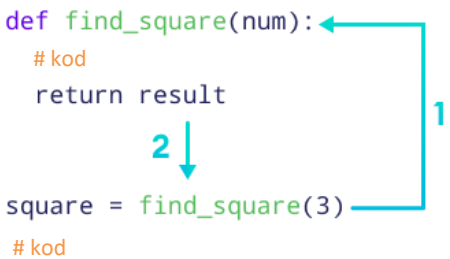


Şekil 2.8: Kontrol yapıları a) koşullar b) döngüler.

2.3.2.3 Fonksiyonlar

Python'da fonksiyonlar, belirli bir işlemi gerçekleştirmek için kullanılan yeniden kullanılabilir kod bloklarıdır. Bunun bir örneği Şekil 2.9'da görülmektedir.

- Tanımlama: *def* anahtar kelimesiyle fonksiyonlar tanımlanır.
- Parametreler ve Dönüş Değerleri**:** Fonksiyonlar veri alabilir ve işlenmiş verileri döndürebilir.



Şekil 2.9: Python fonksiyon yapısı.

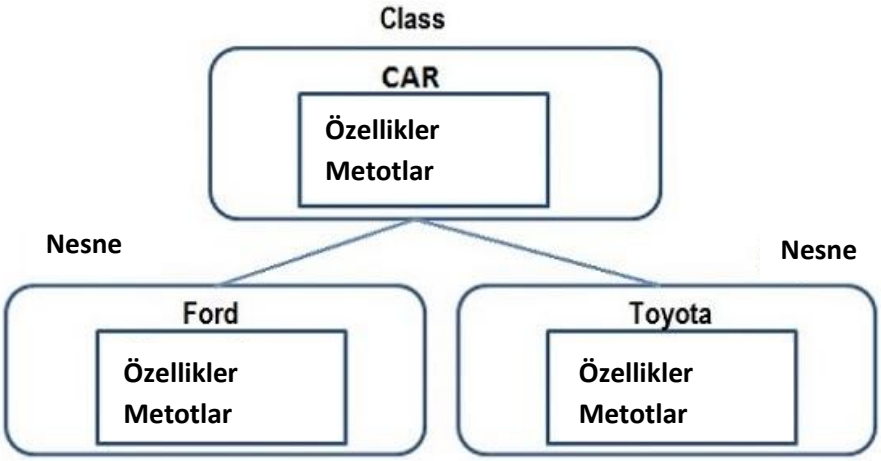
2.3.2.4 Hata Yönetimi

Python, hata yönetimi için *try, except* ve *finally* bloklarını kullanır. Bu yapı, kodun beklenmedik durumlarda stabil çalışmasını sağlar.

2.3.2.5 Nesne Yönelimli Programlama

Python, sınıf ve nesne yapıları üzerinden nesne yönelimli programlamayı desteklemektedir. Şekil 2.10'da basit bir şekilde ifade edilen sınıf, metot ve nesneler şu şekilde açıklanabilir:

- Sınıflar (*Class*): Nesnelerin şablonlarını tanımlar.
- Metotlar: Sınıflara ait işlevlerdir.
- Nesne: Bir sınıftan türetilen ve o sınıfın özellikleri ile metotlarını kullanan örnektir.



Şekil 2.10: Python'da Class ve nesne ilişkisi.

2.3.2.6 Dosya İşlemleri

Python, dosya okuma ve yazma işlemleri için gerekli fonksiyonları sunabilmektedir:

open(), read(), write() gibi fonksiyonlarla dosyalar üzerinde rahatlıkla işlem yapılabilmektedir.

2.3.2.7 Modüller ve Kütüphaneler

Modüller, *Python*'da kodun organizasyonu ve yeniden kullanımı için kullanılan dosya veya paketlerdir. Modüller sayesinde büyük projeler daha yönetilebilir hale gelir.

Yerleşik Modüller: *Python*, matematiksel işlemlerden dosya yönetimine kadar birçok işlemi kolaylaştıran yerleşik modüllerle gelir. Örneğin:

- *math*: Matematiksel işlemler
- *datetime*: Saat ve tarih işlemleri
- *os:* Sistemsel araçlara erişim

Üçüncü Parti Kütüphaneler: Topluluk tarafından oluşturulmuş ve kullanıcılar tarafından indirilip kurulabilen modüllerdir. Örneğin:

- *NumPy*: Sayısal hesaplama
- *Pandas*: Veri analizi
- *Seaborn*: Veri görselleştirme

2.3.2.8 Çevrim ve Tür Dönüşümü

Python, veri türleri arasında kolayca dönüşüm yapabilir:

- *int(), float(), str(), list()* gibi fonksiyonlar bu işlemleri gerçekleştirmek için kullanılır.

Bu temel yapılar, *Python*'ın algoritma geliştirme ve uygulama süreçlerinde ücretli yazılımlardan bağımsız aynı zamanda anlaşılabilir ve de topluluk içeriği bakımından zengin olması aynı zamanda simülasyon yapma ve bu verileri işleme açısından kolay bir platform olması *Python* yazılımını tercih için cazip kılmaktadır.

2.4 Normal Dağılım

Normal dağılım, istatistikte ve mühendislik uygulamalarında sıkça kullanılan bir olasılık dağılımıdır. Bu dağılım, birçok doğal ve insan kaynaklı olayın istatistiksel özelliklerini temsil etmekte önemli bir rol oynar. Normal dağılımın temel özelliği, veri değerlerinin simetrik bir şekilde ortalamaya yakın bir yoğunluk göstermesidir. Matematiksel olarak, normal dağılım bir olasılık yoğunluk fonksiyonu ile ifade edilir ve Denklem (2.15)'teki gibi tanımlanır.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{2.15}$$

Burada:

- μ, dağılımın ortalamasını,
- σ, standart sapmasını temsil eder.

Normal dağılımın çan eğrisi şeklindeki grafiği, merkezde ortalamaya (μ) odaklanır ve uçlara doğru azalan bir eğilim gösterir. Bu özellik, özellikle trafik modelleme ve simülasyonlarında gerçekçi senaryolar oluşturmak için son derece yararlıdır.

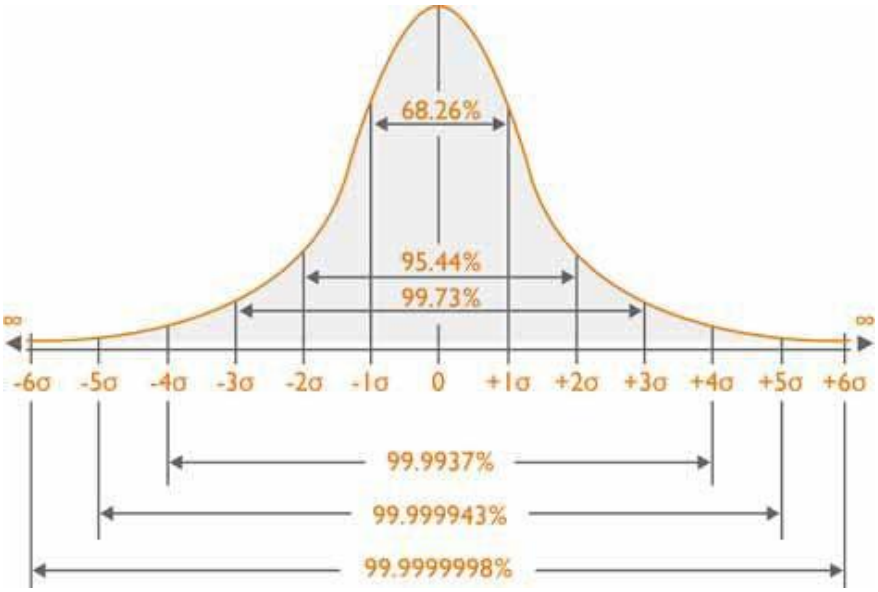
MCS'nin simülasyonunda, trafik yoğunluğunun gerçekçi bir şekilde modellenmesi amacıyla normal dağılımdan faydalanılmıştır. Özellikle, yoğun trafik saatleri gibi durumlarda araç yoğunluğunun zamana ve konuma bağlı olarak değişkenlik göstermesi, normal dağılımın bu simülasyonlarda kullanılmasını sağlamıştır. Bu yaklaşım, simülasyondaki araç akışını gerçek trafik senaryolarına daha yakın bir şekilde yansıtarak, algoritmaların performansını gerçeğe uygun koşullarda değerlendirmek için kritik bir rol oynamaktadır.

2.4.1 Altı Sigma ve Çan Eğrisi

Altı Sigma, 1980'lerde bir Motorola mühendisi olan Bill Smith tarafından Motorola süreçlerine uygulanan ve öncülüğünü yapılan bir toplam kalite yönetimi tekniği olarak ortaya çıkmıştır (Barney 2002, Toma 2008).

Altı Sigma metodu, normal dağılımda verilerin yaklaşık %99,9999998'inin ortalamadan ±6σ değeri kadar içinde yer aldığını ifade etmektedir. Şekil 2.11'de görsel olarak ifade edilen normal dağılım grafiğinde, her bir sigma değeri için verilerin hangi aralıkta yoğunlaştığı aşağıda belirtilmiştir:

- Verilerin yaklaşık %68'i μ ± σ aralığındadır.
- Verilerin yaklaşık %95'i μ ± 2σ aralığındadır.
- Verilerin yaklaşık %99,7'si μ ± 3σ aralığındadır.
- Verilerin yaklaşık %99,9937'si μ ± 4σ aralığındadır.
- Verilerin yaklaşık %99,99994'ü μ ± 5σ aralığındadır.
- Verilerin yaklaşık %99,9999998'i μ ± 6σ aralığındadır.



Şekil 2.11: Normal dağılım (PSU 2024).

Altı Sigma, süreçlerin yalnızca merkezdeki performansını değil, aynı zamanda uç durumlarındaki hatalarını da minimize etmeyi hedefler. Çan eğrisi (normal dağılım) ile ilişkilendirilen bu yöntem, istatistiksel olarak süreçlerin ne kadar tutarlı olduğunu ölçmek için kullanılır. Altı Sigmanın hedefi, süreçlerde meydana gelebilecek hataların milyonda 3,4 oranına kadar düşürülmesidir. Bu yaklaşım, üretim süreçlerinde yüksek kalite ve düşük maliyet dengesini sağlamanın yanı sıra, müşteri memnuniyetini artırmayı ve iş süreçlerinde sürdürülebilir bir iyileştirme sağlamayı amaçlayarak yaygın olarak kullanılmaktadır (Pestorius 2007).

2.4.2 Standart Sapma

Standart sapma, bir veri kümesindeki değerlerin aritmetik ortalamaya olan uzaklıklarının ölçüsüdür ve veri dağılımının ne kadar yayıldığını anlamak için kullanılır. Bu ölçü, veri setinin merkezi bir değer etrafında nasıl dağıldığını anlamaya yardımcı olur ve istatistikte önemli bir yer tutar.

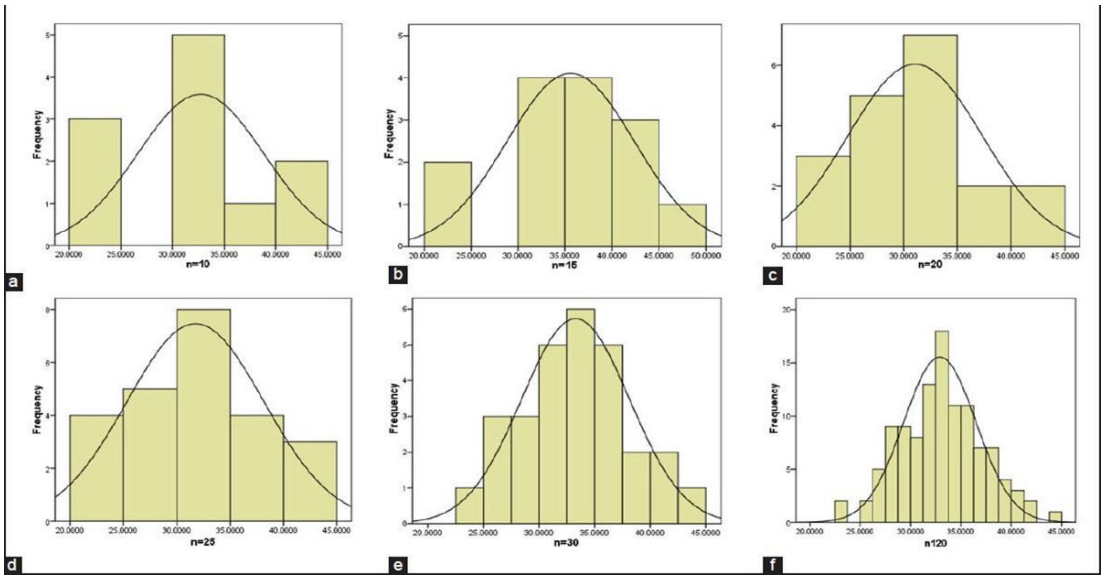
Matematiksel olarak standart sapma, bir veri kümesindeki her bir değerin ortalamadan farkının karesinin alınması, bu karelerin toplamının veri sayısına (ya da örneklemde bir eksiğine) bölünmesi ve ardından karekök alınması ile hesaplanır. Bu formül şu şekilde ifade edilir:

Belirli bir veri aralığının altıda biri olarak standart sapmanın sabitlenmesi, normal dağılımın özelliklerine dayanır. Normal dağılımda, verilerin yaklaşık %99,7'si ortalamanın ±3σ standart sapma aralığında yer alır. Bu, toplam veri aralığının yaklaşık 6 standart sapma genişliğinde olduğunu gösterir.

Dolayısıyla, veri aralığının altıda biri, bir standart sapmaya (σ) karşılık gelir. Bu yaklaşım, özellikle veri setinin aralığının önceden belirlenip standart sapmanın bilinmediği durumlarda simülasyonlarda ve istatistiksel analizlerde, kolaylık sağlamaktadır.

2.4.3 Örneklem Sayısı

Veri sayısının azlığı, normal dağılım varsayımının geçerliliğini olumsuz etkileyebilir. Özellikle küçük örneklem büyüklüklerinde, verilerin dağılımı normalden sapma eğilimi gösterebilir. Bu durum, verilerin yayılımının yetersiz tahmin edilmesine yol açarak, frekans dağılımının normal eğriyi yansıtmasını engeller (Krithikadatta 2014).



Şekil 2.12: Farklı örneklem (n) büyüklüklerinde frekans dağılımı (Krithikadatta 2014).

Örneklem büyüklüğünün artmasıyla birlikte, örneklem dağılımının normal dağılıma yaklaşma eğilimi gösterdiği bilinmektedir (PSU 2024). Ancak, tali yol girişleri gibi spesifik durumlarda veri sayısının sınırlı olması, tam anlamıyla normal dağılım elde etmeyi zorlaştırabilir. Bu nedenle, küçük örneklem büyüklüklerinde normal dağılım varsayımının geçerliliğini dikkatlice değerlendirmek önem taşır.

İstatistiksel analizlerde, örneklem büyüklüğünün 30'dan büyük olması, Merkezi Limit Teoremine göre, örneklem ortalamalarının yaklaşık olarak normal dağılım göstermesini sağlar (Ganti 2024). Şekil 2.12'de farklı örneklem büyüklüklerine göre verilen grafikler bu olguyu desteklemektedir. Bu sebeplerden dolayı veri seti oluşturulurken bu kriterler önem arz etmektedir.

2.5 MCS

MCS, şehir içi ulaşım sistemlerini yeniden tanımlayan, trafik izole edilmiş ve otonom araç teknolojileriyle desteklenen bir ulaşım çözümüdür. Şehirlerdeki trafik sıkışıklığı, enerji verimliliği, çevre kirliliği ve güvenlik sorunlarını ele alarak çözüm getirmeyi hedefleyen yeni bir bakış açısıyla tasarlanmış bir sistemdir.

MCS, çevre dostu ve uygulanabilir bir çözüm sunan yeni bir çift modlu ulaşım sistemidir. Şehir içi ulaşım sorunlarını çözmeyi hedefleyen bu sistem, iki süreçten oluşur:

1. Kullanıcı Kontrollü Süreç:
    - Araçlar, ev/ofis ile MCS istasyonu arasında kişisel araç mantığında kullanıcı tarafından kontrol edilir.
    - Kullanıcı, aracı protokollerle sisteme entegre eder.
2. MCS Kontrollü Süreç:
    - Araçlar, tamamen otonom olarak hareket eder.
    - MCS yollarında, trafik sorunlarından bağımsız ve çarpışmasız bir ulaşım sağlanır.

Sistemin Temel Özellikleri:

- Hat Özerkliği: MCS yolları yalnızca sisteme ait araçlara tahsis edilmiş olup insan, hayvan ve trafik gibi dış etkenlerden arındırılmıştır.
- Trafiksiz ve Işıksız Ulaşım: Trafik sıkışıklığı ve ışık problemleri ortadan kaldırılır.
- Konfor ve Hijyen: Araçlar tek kişilik, özel tasarlanmış ve hijyeniktir.
- Esnek Kullanım: Kapıdan kapıya ulaşım sağlanır, aktarma gerektirmez.
- Zaman Tasarrufu: Kullanıcı, yolculuk sırasında farklı aktiviteler gerçekleştirebilir.

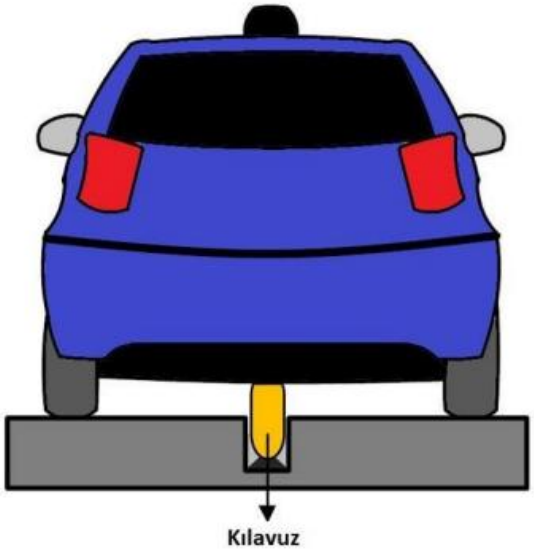
Avantajlar**:**

- MCS, metro sistemlerinden daha esnek; kişisel ve sürücüsüz araçlardan daha güvenli ve konforludur.
- Altyapı maliyeti, metro sistemlerinden düşük; diğer sistemlere göre daha yüksek olmasına rağmen avantajlarıyla öne çıkar.

2.5.1 MCS Temel Bileşenleri

MCS'yi oluşturan temel unsurlar iki ayrı başlık altında incelenebilir. Bunlardan birincisi, MCS'ye ait özgün yapıya sahip olan araçlar ve bu özel araçların takip ettiği yollardır.

2.5.1.1 MCS'nin Araç Yapısı



Şekil 2.13: MCS aracının arkadan görünümü (Bozuyla 2019).

MCS araçları, çift modlu ulaşım sistemine uygun olarak hem manuel hem de otonom kullanım için tasarlanmış, kişisel araçlardan farklı özelliklere sahip özel araçlardır (Bozuyla 2019). Şekil 2.13'te MCS araç yapısının arkadan görünümü verilmiştir. Normal bir kişisel araca benzemesine rağmen bazı özellikler bakımından farklılıklar içermektedir.

Temel Özellikler:

- Kapasite: Tek kişilik.
- Enerji: Yalnızca elektrikle çalışır.
- Boyutlar: Standart uzunluk, genişlik ve yükseklik ölçüleri.
- Tasarım: Bireysel araç görünümlü.
- Mekanik: Tüm araçlar aynı mekanik özelliklere sahiptir.

Kullanım Özellikleri:

- Hem sürücülü hem otonom hareket kabiliyeti.
- Sabit hızlanma ve yavaşlama ivmesine sahip.
- Kılavuz yoluna tekerlek ile bağlanır.

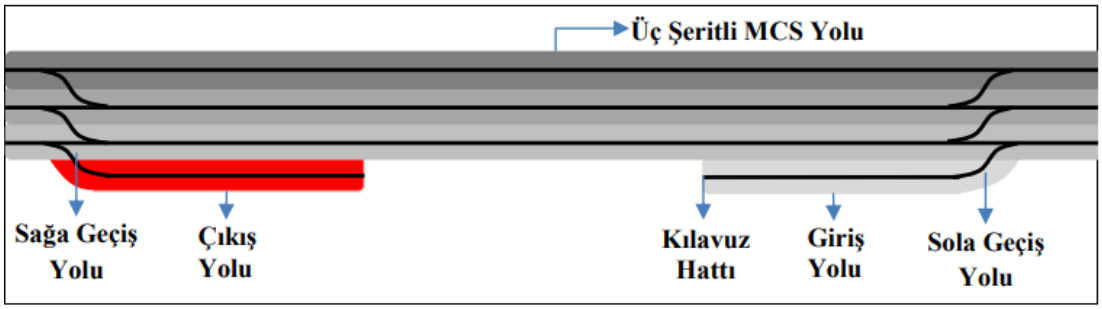
Teknolojik Özellikler:

- Hareketli kılavuz ile enerji ve veri transferi sağlar.
- DRSC, GPS gibi haberleşme ve güvenlik teknolojilerini içerir.
- Diğer sistem bileşenleriyle iletişim kurabilen donanımlara sahiptir.

Konfor ve İşlevsellik:

- Bireysel çalışma ve dinlenme için uygundur.
- Güvenli, konforlu ve hijyenik seyahat imkânı sunar.

2.5.1.2 MCS Yol Ortamı



Şekil 2.14: 3 Şeritli MCS yol ortamı (Bozuyla 2019).

MCS yol protokolü, Şekil 2.14'te görülen giriş, çıkış ve ana yol bileşenleri ile bu yolların kullanım kurallarını belirler (Bozuyla 2019).

Giriş Yolları:

- Araçların sisteme giriş yaptığı, ana yola paralel inşa edilmiş tali yollardır.
- Tüm giriş yolları aynı boyutlarda ve maksimum hızlanma için yeterli uzunlukta tasarlanmıştır.
- Araçların enerji ve veri transferi için kılavuz hattı bulunur.

- Giriş yoğunluğu ve sırasını düzenleyen bekleme istasyonları mevcuttur.

Çıkış Yolları:

- Araçların sistemi terk ettiği yollardır ve kontrolün kullanıcıya devredildiği alanlardır.
- Ana yola paralel ve tüm çıkış yolları aynı boyuttadır.
- Araçların yavaşlayarak durabileceği şekilde tasarlanır.
- Kılavuz hattı sayesinde enerji ve veri transferi sağlanır.

Ana Yollar

- MCS'nin omurgasını oluşturan en az üç şeritli yollar. Araçlar, girişten itibaren
- Tüm link yolları düz, kavisler ise güvenli ve konforlu dönüş sağlayacak şekilde tasarlanır.
- Şerit değişimi yalnızca sağa/sola geçiş yolları ile yapılır.
- Araçların savrulmasını önlemek ve enerji/veri transferi sağlamak için kılavuz hattı bulunur.
- Düğüm noktaları sabit aralıklarla ve geçiş yollarıyla uyumlu olacak şekilde konumlandırılmıştır.

Genel Özellikler

- MCS yolları dış etkenlerden arındırılmış özel tasarımlara sahiptir.
- Tüm yollar, araçların güvenli, konforlu ve verimli bir şekilde hareket edebilmesini sağlamak üzere optimize edilmiştir.

2.5.1.3 İstasyonlar

İstasyonlar, MCS'nin çift modlu işleyişinin anahtar noktalarıdır. Sistem kullanıcılarının araçlarını otonom sürüş moduna geçirdiği veya otonom sürüşten manuel sürüşe geçtiği alanlardır (Bozuyla 2019).

2.5.1.3.1 Giriş İstasyonları:

Otonom Geçiş Noktası: Araçlar, giriş yolları yardımıyla istasyona gelir ve burada sürücü kontrolünden otonom sürüşe geçer.

Trafik Yönetimi: Bekleme alanları sayesinde araç giriş öncelikleri düzenlenir, yoğunluk kontrol edilir.

Standartlaştırılmış Yapılar: Her giriş istasyonu, araçların hızlanma ivmesi ve kılavuz bağlantısı için gerekli fiziksel standartlara sahiptir.

2.5.1.3.2 Çıkış İstasyonları

Manuel Geçiş Noktası: MCS yollarında otonom olarak hareket eden araçlar, çıkış istasyonunda manuel sürüşe geçer.

Yavaşlama Alanları: Araçların hızlarını güvenli bir şekilde azaltabileceği tasarım özellikleri içerir.

Düğüm Öncesi Konumlandırma: Çıkış istasyonları, yol düğümlerinden önceki stratejik noktalara yerleştirilir.

2.5.1.3.3 Bekleme Alanları

Yoğunluk Yönetimi: Kullanıcıların giriş ve çıkışta bekleme sürelerini optimize etmek için tasarlanmış alanlardır.

Araç Sıralama: Kullanıcı önceliklerine göre araçlar sıraya alınır, bu da sistem akışını hızlandırır.

2.5.1.4 Link ve Düğümler

Linkler, MCS'nin yol altyapısının temel yapı taşlarını oluşturur ve iki düğüm noktası arasındaki hat parçalarını ifade eder. Bu yapı, sistemin modülerliğini ve esnekliğini sağlamaktadır (Bozuyla 2019).

Her bir link, aşağıdaki özelliklere sahiptir:

- Bağımsız Veri Tabanı: Her link, kendine ait bir veri tabanına sahiptir ve bu veri tabanı, linkteki trafik akışı, araç geçişleri, enerji transferi ve şerit kullanımı gibi bilgileri saklar. Bu veriler, düğüm etmenleri tarafından kullanılarak sistemin yönetilmesine olanak tanır.
- Şerit ve Hattın Görevleri: Linkler, en az üç ana şerit ve bir giriş veya çıkış hattını içerir. Şerit değiştirme işlemleri yalnızca link sonundaki sağa ve sola bağlanma yollarında gerçekleşir. Aynı link üzerinde hem giriş hem çıkış bulunmaz. Bu düzen, araçların hareketlerini optimize eder ve çakışmaları önler.
- Sorumluluk Alanı: Linkler, bağlı oldukları düğüm etmeninin kontrolündedir ve bu düğüm, ilgili linkin istatistiksel verilerini yönetir.

Düğümler: Düğüm noktaları, linkler arasındaki bağlantıyı sağlayan, MCS'nin koordinasyon ve yönetim ağının önemli bir parçasıdır. Bu noktalar, sistemin hem fiziksel hem de dijital altyapısında kritik bir rol oynar (Bozuyla 2019).

- Düğüm Etmenleri:
    - Her düğüm noktası, o bölgeye özgü bir düğüm etmeni içerir. Düğüm etmenleri, sistem etmenine bağlı olmakla birlikte, daha dar bir kontrol ve yönetim alanına sahiptir.
    - Düğüm etmenleri, sorumlu oldukları linklerin istatistiksel bilgilerini depolar ve bu bilgileri sistem etmeni ile paylaşır. Örneğin, araç yoğunluğu, hız ortalamaları ve şerit kullanım oranları gibi veriler, düğüm etmeni tarafından kaydedilir.
    - Araçların yön değişimi ve şerit değiştirme kararları, düğüm etmenleri tarafından alınır. Bu kararlar hem güvenliği sağlamak hem de trafiğin akışını optimize etmek amacıyla verilir.

- Düğümlerin Konumlandırılması:
    - Düğüm noktaları, sistem yöneticisi tarafından tanımlanan sabit aralıklarla yerleştirilir. Bu aralıklar, sistemin ihtiyaçlarına göre ayarlanabilir.
    - Her düğüm noktası, üzerinde bulunan şerit sayısı kadar düğüm etmenine sahiptir. Bu yapı, paralel veri işleme kapasitesini artırır.
    - Düğümler, sağa ve sola bağlanma yolları için optimize edilmiş bir yapıya sahiptir ve şerit değişimlerinin yalnızca bu noktalarda yapılması, sistemin kontrolünü kolaylaştırır.
- Haberleşme ve Görevleri:
    - Düğüm etmenleri, araç etmenleri ile sistem etmeni arasında bir geçiş noktası işlevi görür. Araçlardan gelen veriler, düğüm etmenleri aracılığıyla sistem etmenine iletilir ve tersi bir akışla araçlara komutlar gönderilir.
    - Ayrıca, düğüm etmenleri sorumlu oldukları linkte meydana gelen olayları (örneğin, bir araç arızası veya beklenmedik bir hız değişikliği) hızlı bir şekilde tespit ederek sistem etmenine rapor eder.

Bu yapı sayesinde MCS, modüler bir şekilde çalışır ve sistemin her bir bileşeni arasında etkin bir koordinasyon sağlanır. Link ve düğüm kavramları, sistemin yüksek verimlilikle çalışmasını ve kontrol edilebilir olmasını garanti eder (Bozuyla 2019).

2.5.2 MCS'de Kullanılan Algoritmalar

MCS'de kullanılan bazı algoritmalar ve önceki bölümlerde de bahsi geçen düz yol takip ve birleşme algoritmaları bu kısımda anlatılacaktır.

2.5.2.1 Rulet Tekeri Seçim Algoritması

Bu algoritma, bir veri kümesindeki elemanları uygunluk derecelerine göre seçmek amacıyla geliştirilmiş bir yöntemdir. Genellikle genetik algoritmalarda birey seçiminde kullanılan bu teknik, çalışmada araçların çıkış yollarını belirlemek için uyarlanmıştır. Algoritma, yolların rastgele değil, belirli olasılık değerlerine dayalı

olarak seçilmesini sağlar ve böylece daha gerçekçi bir dağılım elde edilir (Bozuyla 2019).

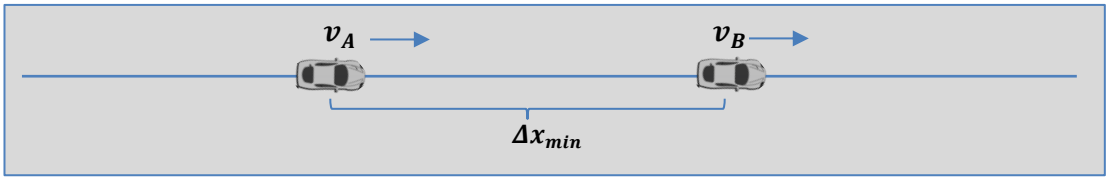
MCS'ye özel olarak düzenlenip uyarlanan rulet algoritmasının işleyişi, Şekil 2.15'te açıklanmaktadır.

```
1.Başla
2.Tüm dizi değerlerini topla ve dizi içerisindeki yüzdelik
  oranlarını belirle
3.Oranları küçükten büyüğe sırala
4.Rastgele bir sayı üret
  a.Üretilen sayı ≥ dizi[a] ve sayı < dizi[a+1] ise
      i.  Çıkış = a
     ii.  Bitir
  b.4.adıma git
```

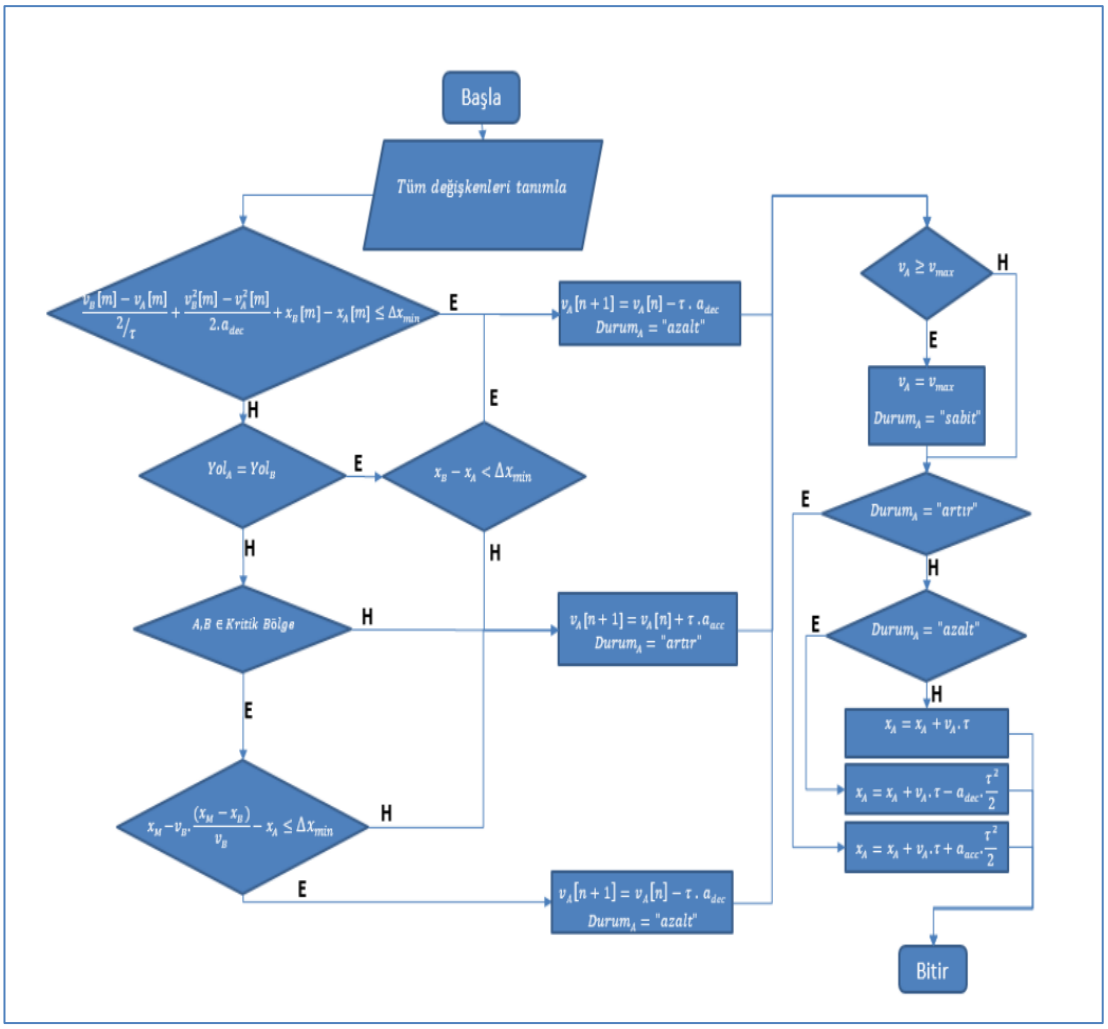
Şekil 2.15: Rulet algoritması (Bozuyla 2019).

2.5.2.2 Düz Yol Takip Algoritması

MCS'de araların düz güzergahta birbirlerini kaza yapmadan takip edebilmeleri gerekmektedir. Bunun için arkadaki aracın bir öndeki aracı Şekil 2.16'da Δx_{min} olarak ifade edilen güvenli bir mesafede takip etmesi beklenmektedir. Bunun için araçların "HIZLAN" veya "YAVAŞLA" şeklinde iki durumlu bir takip algoritması geliştirilmiştir (Bozuyla 2019). Bu geliştirilen algoritmanın şeması ise Şekil 2.17'de verilmiştir.



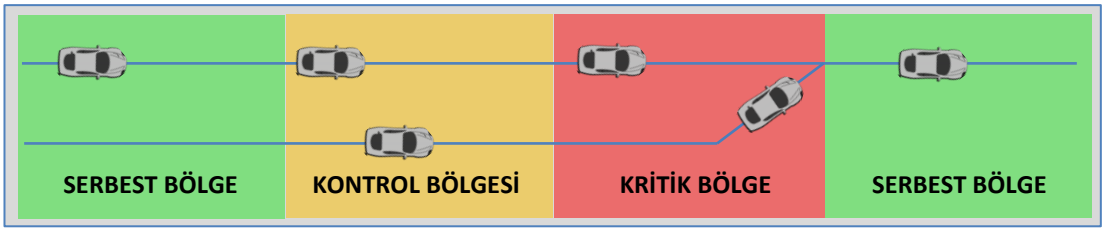
Şekil 2.16: Araç takibi.



Şekil 2.17: MCS takip algoritması (Bozuyla 2019).

2.5.2.3 Birleşme Algoritması

Bir link parçasını ifade eden Şekil 2.18'de görüldüğü gibi, iki yolun bir araya geldiği düğümlerde güvenli bir birleşme gerçekleşebilmesi için yollar üç farklı ayrı bölgeyle kategorize edilmiştir (Bozuyla 2019). Araçların bu kategorize edilmiş kısımlarda takip edeceği birleşme algoritmasının şeması Şekil 2.19'da verilmiştir.

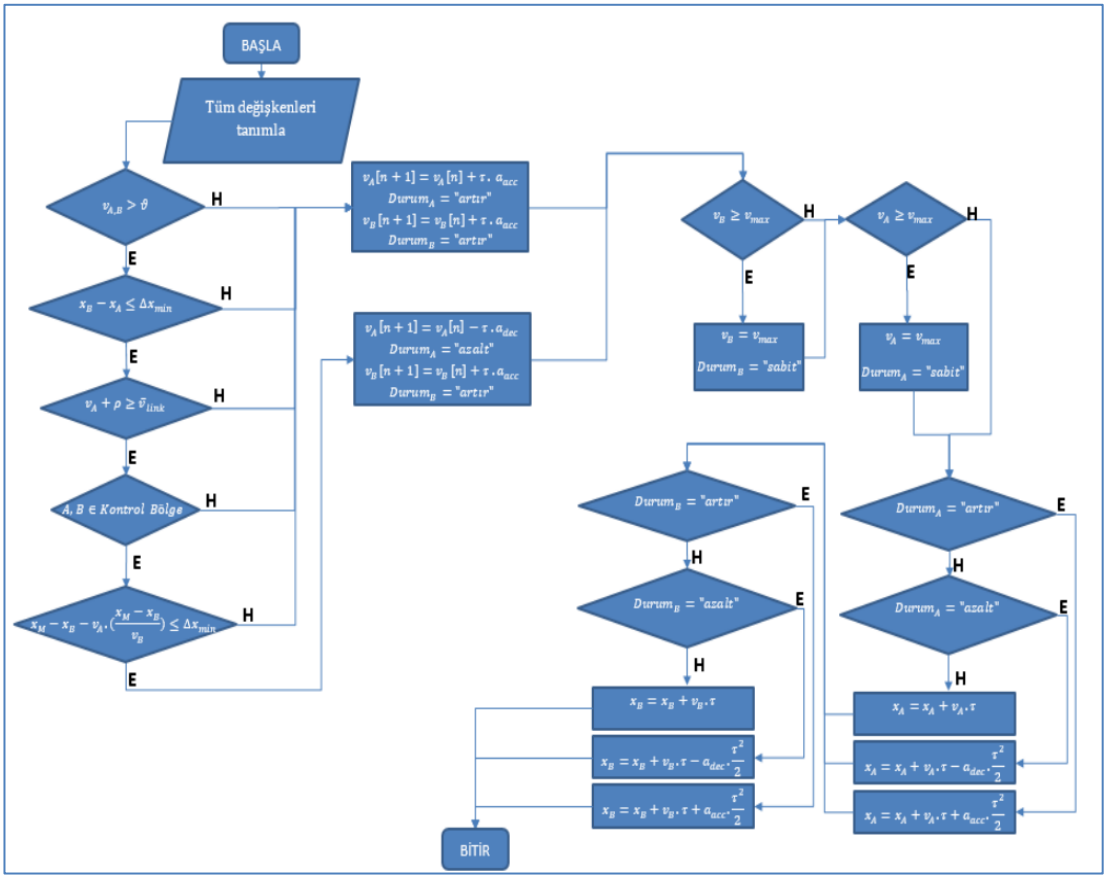


Şekil 2.18: MCS link bölgeleri (Bozuyla 2019).

Serbest bölgede araçlar düz yol takip algoritmasını aynen uygulayarak devam etmektedirler.

Kontrol bölgesindeki araçlar ise geçiş önceliği belirlenmesi amacıyla birbirleriyle haberleşmeye başlamaktadırlar. Bu bölgede giriş yolundaki araç aynı zamanda kontrolör vazifesini görmektedir (Bozuyla 2019).

Kritik bölgede ise araçlar tüm birleşme işlemlerini tamamlamış bir vaziyette olup aynı serbest bölgedeki gibi yeni sıralamaya göre takip algoritmasını uygulamaktadırlar (Bozuyla 2019).



Şekil 2.19: Birleşme algoritması (Bozuyla 2019).

3. YÖNTEM VE MATERYAL

3.1 Giriş

Ulaşım sistemlerindeki yenilikçi bir yaklaşım olan MCS incelendiğinde şehir içi ulaşıma verimli, sürdürülebilir ve hızlı bir yaklaşım olduğu görülmektedir. Ancak yeni bir çalışma alanı olmasının getirdiği eksiklikler de mevcuttur.

Bu tez çalışması kapsamında ise önceden *NetLogo* ortamında gerçeklenen MCS, bu çalışmada *Python* kullanılarak gerçeklenmiş ve aynı zamanda MCS'de mevcut olan takip algoritması iyileştirilmiş ayrıca yeni bir birleşme algoritması önerilmiştir. Bu algoritmalar test edilerek birbirleriyle kıyaslanmıştır.

Bu bölüm kapsamında ise öncelikle *Python* ortamındaki MCS simülasyonu ele alınarak tanıtımı yapılacaktır. Devamında ise sonraki algoritma geliştirmelerine ön ayak olması adına ayrık zaman formüllerinde bazı genelleştirmeler yapılacaktır. Bu genelleştirilmiş formüller kullanılarak önceki çalışmalarda hızlanan veya yavaşlayan şeklinde 2 durumlu olarak geliştirilmiş olan takip algoritması tekrardan ele alınacak, yeni bir durum olarak sabit gidebilme kabiliyeti kazandırılarak 3 durumlu yeni bir algoritma tanıtılacaktır. Ardından genelleştirilen ayrık zaman formüllerinin yardımıyla yeni bir birleşme algoritması önerilecektir. Bu geliştirilen algoritmalar *Python* simülasyon ortamında kontrollü bir şekilde test edilerek sonuçları incelenecektir.

3.2 Python MCS Simülasyon Ortamı

Python kolay ve anlaşılabilir bir dil olmasının getirdiği avantajlardan biri de karmaşık simülasyonların gerek duyduğu paket programların yapabileceği işleri daha verimli ve ücretsiz bir şekilde, en basit halde sunabilmesidir.

Bu bölüm kapsamında *Python* ortamında geliştirilen MCS simülasyonu geliştirilirken kullanılan kütüphaneler, nesne merkezli tasarımdaki *class* yapıları, arayüz ve veri görselleştirmeleri üzerinde durulacaktır.

3.2.1 Kütüphaneler

Simülasyonda kullanılacak amaçlar doğrultusunda doğru kütüphanelerin seçilmesi, geliştirme sürecini hızlandırmak ve karmaşık işlemleri basitleştirmek açısından büyük önem taşımaktadır. *Python*, geniş kütüphane desteğiyle bu konuda büyük avantaj sağlar. MCS simülasyonu kapsamında kullanılan kütüphaneler, matematiksel hesaplamalardan zaman takibine, veri görselleştirmeden rastgelelik içeren veri setlerinin oluşturulmasına kadar birçok ihtiyaçta kullanılmaktadır. MCS simülasyonunda başlıca kullanılan kütüphaneler şöyledir:

3.2.1.1 Math

Matematiksel işlemler, simülasyonun temel taşlarından biridir ve bu tür işlemleri hızlı ve doğru bir şekilde gerçekleştirmek, simülasyonun verimliliği açısından kritik bir öneme sahiptir. Math kütüphanesi, MCS simülasyonunda karekök alma, kuvvet hesaplama gibi temel matematiksel işlemler için kullanılmıştır. Özellikle araçların hız ve pozisyon hesaplamalarında büyük bir kolaylık sağlamıştır. Bu kütüphane, *Python*'ın yerleşik yapısı sayesinde ek bir yük getirmeden hesaplama gereksinimlerini karşılamış ve simülasyonun temel matematiksel altyapısını sağlamıştır.

3.2.1.2 Time

Python'da zaman takibi, simülasyon performansını değerlendirmek ve çıktıları optimize etmek açısından kritik bir rol oynar. *Time* kütüphanesi, MCS simülasyonunda ağırlıklı olarak simülasyonun gerçek dünyada ne kadar sürede tamamlandığını ölçmek ve saniyedeki kare sayısı takibi yapmak için kullanılmıştır. Bununla birlikte, simülasyonun iç zaman akışı Adım Sayısı adı verilen ayrı bir kavramla yönetilmekte olup, simülasyonun yavaş çalıştığı durumlarda bile geçen zaman sabit kalmaktadır. Bu ayrım sayesinde simülasyon mantığı ve performans analizi birbiriyle karışmadan ele alınmıştır. *time.time* fonksiyonu, gerçek zaman ölçümlerini sağlayarak simülasyonun optimizasyonu ve performans değerlendirmesi için gerekli olan temel verileri sunmaktadır.

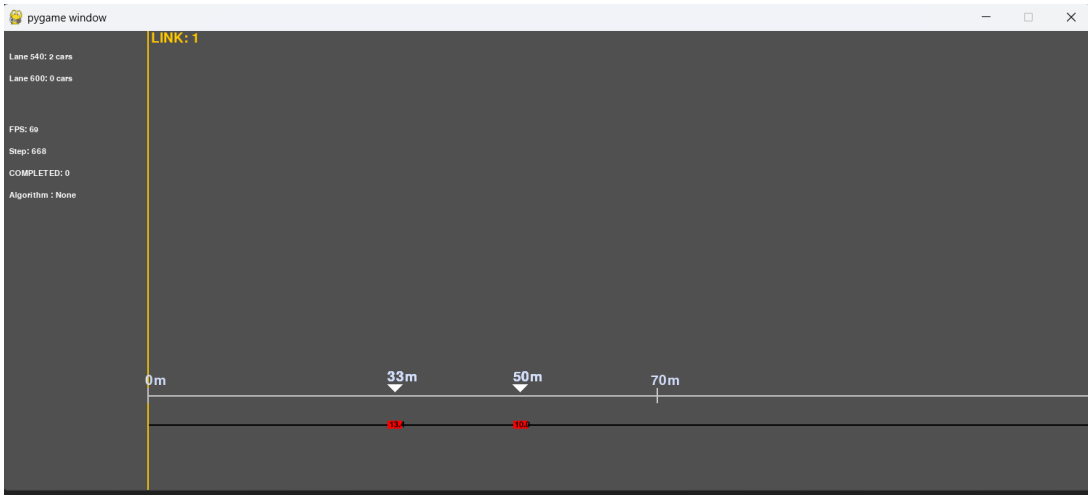
3.2.1.3 Numpy

Simülasyonda kullanılan veri setlerinin oluşturulmasında gerekli olan nümerik *Python* kütüphanesidir. MCS simülasyonunda araç giriş veri setlerini oluşturmak için kullanılan *NumPy* kütüphanesi, *NumPy*'ın *numpy.random.normal* fonksiyonu, araç giriş zamanlarını modellemek için kullanılan normal dağılımı kolayca oluşturmuş, bu sayede rastgelelik içeren gerçeğe yakın simülasyonların geliştirilmesini mümkün kılmıştır. Ayrıca araç giriş zamanlarının yeniden ölçeklendirilmesi, sıralanması ve minimum değerlerinin düzeltilmesi gibi işlemler, *NumPy* dizilerinin yapısı sayesinde kolay bir şekilde gerçekleştirilmiştir. İstatistiksel parametrelerin hesaplanması sırasında da *NumPy*'ın optimize edilmiş *mean* ve *std* fonksiyonlarından yararlanılarak elde edilmiştir. Bunun yanı sıra, Matplotlib ile veri görselleştirme işlemleri için gerekli olan bin sınırlarının hesaplanmasında *NumPy*'ın güçlü fonksiyonları kullanılmıştır. Tüm bu özellikleriyle *NumPy*, hem MCS simülasyonu sırasında araç giriş veri setlerinin oluşturulmasını kolaylaştırmış hem de bu süreçteki hesaplama yükünü hafifleterek kodun daha okunabilir ve verimli olmasını sağlamıştır.

3.2.1.4 Pygame

Pygame kütüphanesi, özellikle oyun ve görsel simülasyon geliştirme için kullanılan bir *Python* modülüdür. Bu simülasyonda, MCS'nin görselleştirilmesi ve kullanıcı arayüzü tasarımı için kullanılmıştır. Simülasyon sırasında araçların şerit değiştirme, takip etme ve birleşme gibi dinamik hareketlerini görselleştirmek gerektiğinde etkileşime girebilmek ve de fonksiyonelliğini daha anlaşılır kılmak amacıyla *Pygame* kullanılmıştır.

Bu özellikler sayesinde, simülasyonun sonuçlarının görsel olarak değerlendirilmesi ve test edilen algoritmaların etkinliğinin gözlemlenmesi kolay bir hale gelmektedir. Buna ek olarak *Pygame*'de bulunan bazı modüller sayesinde de farklı kütüphanenin açılır pencere operasyonlarını kendi penceresine entegre edebilme gibi kullanışlı özellikleri sayesinde farklı simülasyon parametreleri tek bir pencerede inceleme imkânı sunmaktadır. Şekil 3.1 ve Şekil 3.2'de *Python* ortamında gerçekleştirilen MCS benzetimi verilmiştir.



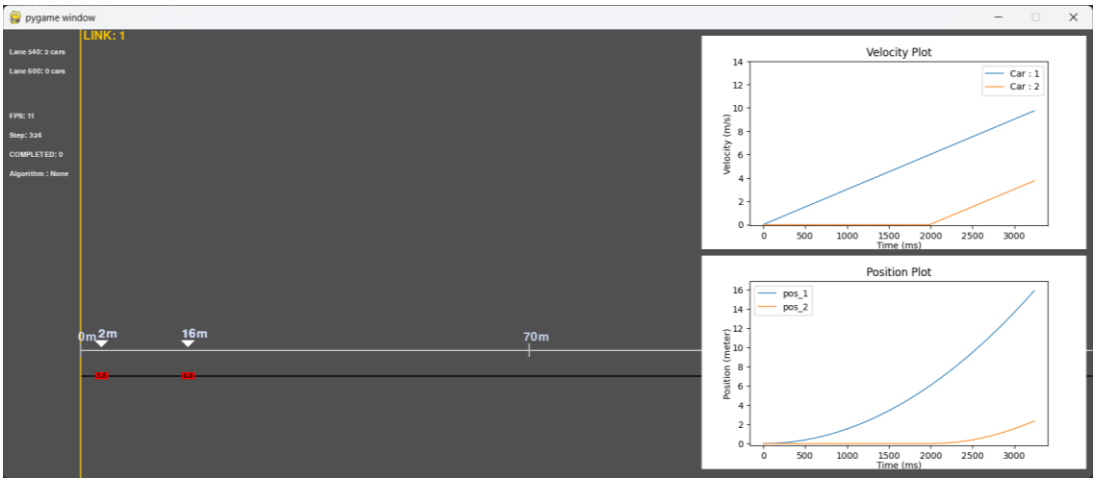
Şekil 3.1: Pygame MCS simülasyon ortamı.

Şekil 3.2: Pygame MCS simülasyon ortamının genişletilmiş görüntüsü.

3.2.1.5 Matplotlib

Matplotlib kütüphanesi, *Python*'da veri görselleştirme için günümüzde en çok tercih edilen araçlardan biridir. Bu kütüphane sayesinde simülasyondan elde edilen veriler, görsel olarak daha anlaşılır hale getirilebilmektedir. *Matplotlib*'in bir alt modülü olan *pyplot*, grafik çizimini kolaylaştıran bir arayüz sunar ve *Matplotlib* kütüphanesinin işlevselliğini daha sade bir şekilde kullanılmasına olanak tanır (Hunter 2007).

Bu projede hem *Matplotlib*'in genel özellikleri hem de *Matlab* benzeri bir yapıya sahip olan *Pyplot* modülü kullanılarak grafik oluşturma işlemleri Şekil 3.3'teki gibi gerçekleştirilmiştir.



Şekil 3.3: Pygame MCS simülasyon ortamında Matplotlib kullanımı.

3.2.2 Python MCS Simülasyon Class Yapıları

Simülasyon tabanlı sistemlerde, kodun düzenli ve modüler bir yapıda olması, sistemin sürdürülebilirliği ve genişletilebilirliği açısından kritik önem taşır. Bu bağlamda, *class* yapıları, simülasyonun temel bileşenlerini nesne yönelimli programlama paradigmasına uygun bir şekilde oluşturarak yönetmeyi sağlar. *Python* dilinde geliştirilen MCS simülasyonu, iki ana *class* üzerinden yapılandırılmıştır: *MetroCarSim* ve *CAR* class'ları. Bu *class*'lar simülasyonun ana fonksiyonlarını ve araç nesnelerinin davranışlarını tanımlayarak sistemi işler hale getirir.

3.2.2.1 MetroCarSim Class

MetroCarSim class'ı, simülasyonun ana yapısını ve iş akışını kontrol eden merkezi bir yapıdır. Sistemin başlangıç parametreleri bu *class* içerisinde tanımlanarak başlatılır (örneğin simülasyon modu, link sayısı, araç boyutları, hız-ivme sınırları ve görselleştirme araçlarının açıp kapanması gibi).

MetroCarSim, araç giriş-çıkış operasyonları, trafik yoğunluğu hesaplamaları, araçların konum güncellemeleri ve çizim işlemleri gibi temel simülasyon adımlarını içeren birçok fonksiyonu barındırır.

```
class MetroCarSim:
    """
    MAIN SIMULATION ENVIRONMENT
    """

    new *
    def __init__(self,
                 link_pair_number=1,
                 fps=100,
                 car_speed=20,
                 car_length=2,
                 car_width=1.1,
                 phi=2,
                 delta=4,
                 speed_display='mps',
                 full_screen=False,
                 matplot=True,
                 car_text=True,
                 lane_density=True,
                 lengths=True,
                 testing=False,
                 window_width=1920,
                 window_height=1080):
```

Şekil 3.4: MetroCarSim class yapısının giriş parametreleri.

MetroCarSim class yapısında simülasyonun parametrelerini rahatlıkla kontrol edebilmek adına giriş parametreleri tanımlanmıştır. Şekil 3.4'teki giriş parametreleri şu şekilde açıklanabilir:

link_pair_number: Bu parametre sistemde simüle edilmek istenen giriş ve çıkış link çifti sayısını belirmektedir. Bu sayı arttıkça otomatik olarak simülasyon ortamı büyümektedir.

fps: Açılımı saniye başına kare olan bu parametre sistemin çalışacağı maksimum kare sayısını belirtmektedir.

car_speed: Simülasyondaki araçların metre/saniye cinsinden varsayılan hızını belirtir. Bu parametre, araçların maksimum olarak ne kadar hızlı hareket edeceğini kontrol eder.

car_length: Araçların uzunluğunu belirler.

car_width: Araçların genişliğini tanımlar.

phi: Araçların takip algoritmasında kullanılan bir parametre olup, yeni geliştirilen takip algoritmasındaki sönümleyici alan (φ) değerini metre cinsinden temsil eder.

delta: Araçların takip algoritmasında kullandığı güvenli takip mesafesi (δ) değerini metre cinsinden ifade etmektedir.

speed_display: Hız birimini belirtir. 'mps' değeri metre/saniye birimini kullanarak hızın gösterimini sağlar.

full_screen: Simülasyonun tam ekran modunda çalışıp çalışmayacağını kontrol eder. *False* değeri, pencere modunda çalışmayı ifade eder.

matplot: Görsel analiz ve grafik gösterimi için kullanılır. *True* değeri, simülasyon sırasında *matplotlib* gibi araçlarla grafiklerin oluşturulmasını sağlar.

car_text: Araçlar üzerinde hız bilgi metni gösterilip gösterilmeyeceğini belirler.

lane_density: Şerit yoğunluğunu görsel olarak gösterilmesini sağlayan değerdir.

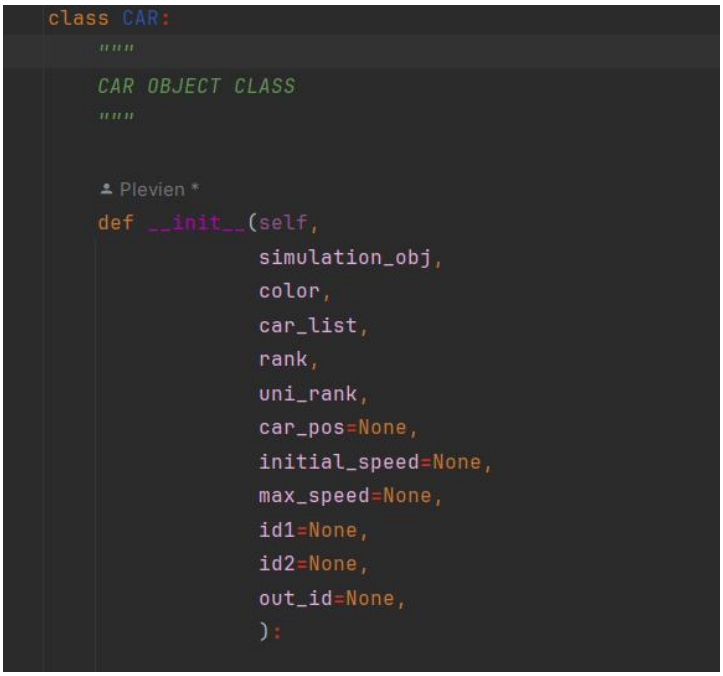
lengths: Şerit uzunluklarını görsel olarak etkinleştirir.

testing: Simülasyonun test modunda çalışıp çalışmayacağını kontrol eder. *True* değeri, sistemin test modunda çalışmasını sağlar.

window_width ve *window_height*: Simülasyon penceresinin genişlik ve yüksekliğini piksel bazında ayarlar. Bu parametreler, görsel arayüzün boyutlarını tanımlar.

3.2.2.2 CAR Class

Araçların sistemdeki bireysel nesnelerini oluşturan *CAR class*'ı, her bir aracın kimliğini ve özelliklerini tanımlar. Araç nesneleri, hız, konum, şerit bilgisi gibi parametreler üzerinden kontrol edilir. Bu yapı, araçların düğüm noktalarında birleşme, hız güncelleme veya şerit değiştirme gibi hareketlerini simüle etmek için gereklidir. *CAR* nesneleri, *MetroCarSim* nesnesi üzerinden oluşturulup sisteme dahil edilir ve belirlenen algoritmalara göre hareket eder. Araçların bireysel davranışları; *MetroCarSim class*'ı tarafından çağrılan *MetroCarSim.update()* gibi fonksiyonlar ile sürekli güncellenir.



Şekil 3.5: CAR class giriş parametreleri.

Simülasyon araçlarını oluşturan bu *class*'daki giriş parametreleri Şekil 3.5'te görüldüğü gibidir. Bu parametrelerin kullanım amaçları şöyle açıklanabilir:

Simulation_obj: Oluşturulacak olan araç nesnesinin simülasyonda bir birey haline gelebilmesi için ana simülasyon nesnesine bağlanması gerekmektedir. Her bir araç oluşturulurken hangi simülasyon nesnesine bağlandığı bu parametreyle belirlenmektedir.

color: Aracın simülasyondaki rengini belirler.

car_list: Aracın bulunduğu linkteki araçların listesi araçla paylaşılır.

rank: Rütbe manasına gelen bu terim simülasyonda sisteme dahil olduğu girişten kaçıncı sırada girdiğini belirten bir kimlik değeridir. Bu verileri analiz edebilmek için gereklidir.

uni_rank: Evrensel rütbe manasına gelen bu terim simülasyonda bütün veri seti içinde kaçıncı olarak sisteme dahil olduğunu ifade etmektedir. Veri setindeki her bir araç eşsiz bir evrensel rütbe değerine sahiptir.

car_pos: Aracın sisteme dahil olacağı konum bilgisini ifade eder.

max_speed: Aracın maksimum çıkabileceği hız sınırını belirler.

id1 ve *id2*: Simülasyonda araçları incelenebilirliğini arttırmak adına *rank* ve *uni_rank* değerlerinin *string* ifadesidir.

out_id: Aracın sistemden çıkış yapacağı düğümün bilgisini içermektedir. Bu bilgi veri setinden alınmaktadır.

3.2.3 Python MCS Simülasyon Akışı

Python ortamındaki simülasyonun akış diyagramı Şekil 3.6'da verilmiştir. Bu akış diyagramı, MCS simülasyonunun çalışma aşamalarını yüzeysel biçimde temsil etmektedir.

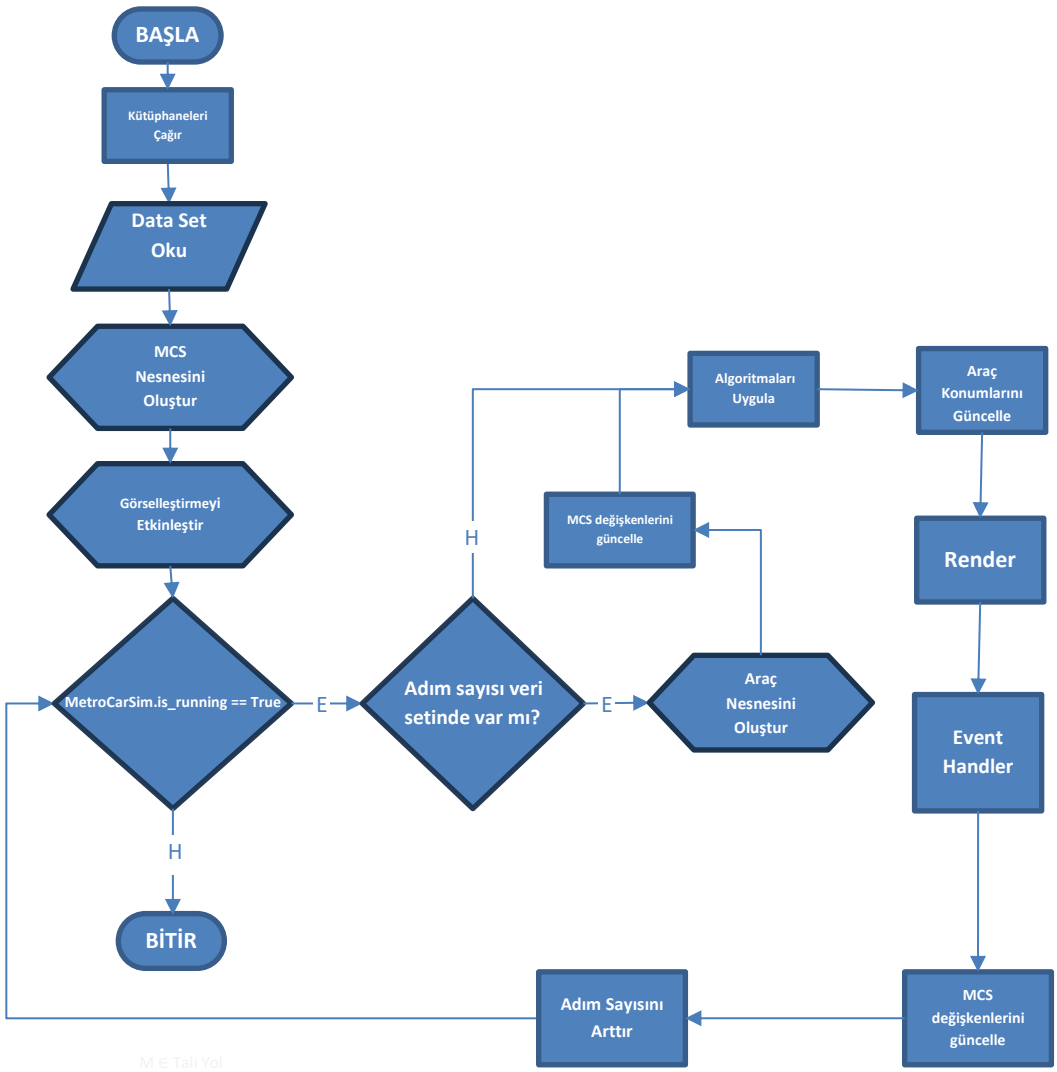
Simülasyon, "BAŞLA" noktasıyla başlar ve ilk olarak veri setini okur. Bu veri seti, simülasyonda kullanılacak araç bilgilerini ve başlangıç durumlarını içerir. Ardından, *MetroCarSim* nesnesi oluşturulur, bu da sistemin ana yapısını tanımlar.

Simülasyonun görsel bir şekilde izlenebilmesi için görselleştirme özelliği etkinleştirilir. Bu aşamadan sonra, simülasyonun çalışıp çalışmayacağını kontrol etmek amacıyla "*MetroCarSim.is_running==True"* durumu doğrulanır.

Eğer bu durum geçerliyse, bir sonraki adıma geçilir. Bu adımda, mevcut adım sayısının veri setinde var olup olmadığı kontrol edilir. Eğer veri setindeki adım sayısı eşleşme durumu varsa sisteme veri setine uygun şekilde araç girişi yapılır ve yeni araç verileriyle birlikte *MetroCarSim* nesnesindeki değişkenler güncellenir. Sistemdeki araçlar mevcut verilere göre algoritma işlemlerine tabi tutulur. Algoritma işlemlerinden sonra araçlarda meydana gelen değişikliklerle birlikte araçların yeni konumu güncellenir.

MetroCarSim değişkenlerinin güncellenmesi ise sistemdeki bütün araç verileri ve bağlı olduğu link bilgilerinin tamamı *MetroCarSim* değişkenlerinde saklanmaktadır. Sistemdeki araçlarda bir değişiklik olması durumunda *MetroCarSim* değişkenlerinin de ona göre tekrar güncellenmesi gerekmektedir.

Bu işlemlerin ardından, simülasyonun görselleştirilmesi için çizim (*Rander*) işlemi gerçekleştirilir. Ayrıca, kullanıcıdan gelen herhangi bir giriş olup olmadığı olay yöneticileri (*Event Handlers*) ile kontrol edilir. Son olarak *MetroCarSim* nesnesinin değişkenleri araçlarda meydana gelen değişimlerden ötürü tekrar güncellenir. Bu süreç tamamlandıktan sonra, adım sayısı artırılır ve "*MetroCarSim.is_running == True*" ifadesi doğruysa döngü yeniden başlar.



Şekil 3.6: Python MCS simülasyon akış şeması.

3.3 Ayrık Zaman Formüllerinin Genelleştirilmesi

Bu kısımda Bölüm 2'deki ayrık zaman formüllerinin geliştirilecek olan algoritma yapısına uygunluk kazanması adına bazı genelleştirmeler yapılacaktır

Varsayım 1: MCS araçlarının herhangi bir başlangıç adımı m olarak kabul edilmiştir.

Varsayım 2: MCS'de her bir araç sabit ivme ile yavaşlayıp hızlanmaktadır. Bu nedenle ivmeler $a[m]$ yerine a olarak tanımlanmakla birlikte hesaplanacak olan bu formüller sadece sabit ivmeli durumlar için geçerli olacaktır.

3.3.1 Hız

Ayrık zaman sistemleri başlığında incelediğimiz Denklem (2.10)'un başlangıç adımı m ile temsil edilecek şekilde düzenlenir ve denklem kullanılarak herhangi bir k anındaki hızını ifade eden Denklem (3.16.c) bulunmaktadır. Burada $v[m]$ başlangıç anındaki hızı temsil etmektedir.

$$v[m+1] = v[m] + a \cdot \tau \tag{3.16}$$

$$v[m+2] = v[m+1] + a \cdot \tau = v[m] + 2 \cdot a \cdot \tau \tag{3.16.a}$$

$$v[m+3] = v[m+2] + a \cdot \tau = v[m] + 3 \cdot a \cdot \tau \tag{3.16.b}$$

$$\vdots$$

$$v[m+k] = v[m] + k \cdot a \cdot \tau \tag{3.16.c}$$

Denklem (3.16.c)'de $k = n + 1$ yazıldığında Denklem (3.17) elde edilmektedir.

$$v[m+n+1] = v[m] + (n+1) \cdot a \cdot \tau \tag{3.17}$$

3.3.2 Konum

Konum formüllerinden olan Denklem (2.14), başlangıç durumu ile ifade edildikten sonra herhangi bir *k* anındaki gösterimi hesaplandığında Denklem (3.18.c) elde edilmektedir.

$$x[m+1] = x[m] + v[m] \cdot \tau \tag{3.18}$$

$$x[m+2] = x[m] + v[m] \cdot \tau + v[m+1] \cdot \tau \tag{3.18.a}$$

$$x[m+3] = x[m] + v[m] \cdot \tau + v[m+1] \cdot \tau + v[m+2] \cdot \tau \tag{3.18.b}$$

$$\vdots$$

$$x[m+k] = x[m] + \tau \cdot \sum_{i=0}^{k-1} v[m+i] \tag{3.18.c}$$

Denklem (3.18.c)'de $k = n+1$ yerine yazılmış ve iki kere toplandığında ve birtakım sadeleştirmeler yapıldığında ise Denklem (3.19.d) elde edilmiştir.

$$x[m+n+1] = x[m] + \tau \cdot \sum_{i=0}^{n} v[m+i] \tag{3.19}$$

$$x[m+n+1] = x[m] + \tau \cdot \begin{pmatrix} v[m] + 0 \cdot a \cdot \tau + v[m] + 1 \cdot a \cdot \tau + \cdots + \\ v[m] + (n-1) \cdot a \cdot \tau + v[m] + n \cdot a \cdot \tau \end{pmatrix} \tag{3.19.a}$$

$$x[m+n+1] = x[m] + \tau \cdot \begin{pmatrix} v[m] + n \cdot a \cdot \tau + v[m] + (n-1) \cdot a \cdot \tau + \cdots + \\ v[m] + 1 \cdot a \cdot \tau + v[m] + 0 \cdot a \cdot \tau \end{pmatrix} \tag{3.19.a}$$

$$2 \cdot x[m+n+1] = 2 \cdot x[m] + \tau \cdot \begin{pmatrix} 2 \cdot v[m] + n \cdot a \cdot \tau + 2 \cdot v[m] + n \cdot a \cdot \tau + \cdots + \\ 2 \cdot v[m] + n \cdot a \cdot \tau + 2 \cdot v[m] + n \cdot a \cdot \tau \end{pmatrix} \tag{3.19.b}$$

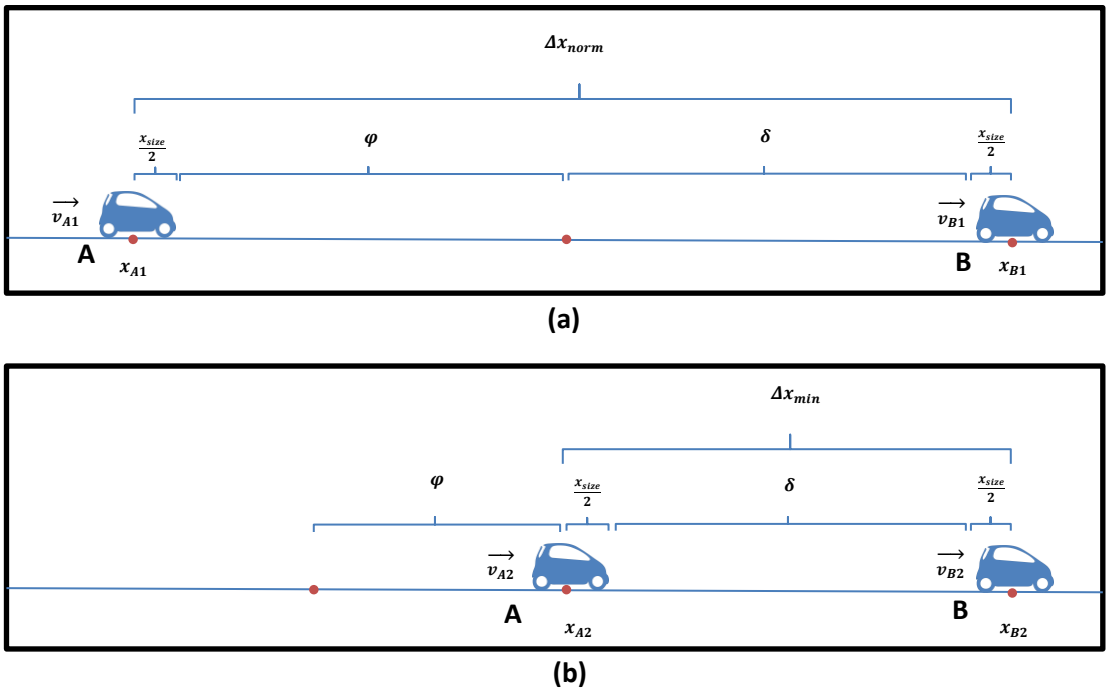
$$2 \cdot x[m+n+1] = 2 \cdot x[m] + \tau \cdot (2 \cdot (n+1) \cdot v[m] + n \cdot (n+1) \cdot a \cdot \tau) \tag{3.19.c}$$

$$x[m+n+1] = x[m] + \tau \cdot (n+1) \cdot \left(v[m] + \frac{n}{2} \cdot a \cdot \tau\right) \tag{3.19.d}$$

3.4 Takip Algoritması

Bir aracın öndeki aracı kaza yapmadan güvenli bir mesafede takip edebilmesi güvenlik açısından bir gerekliliktir. Bu işlemi gerçekleştirmek için ise öndeki aracın belli bir mesafeyi belirli koşulları da sağlayarak muhafaza etmesi gerekmektedir. Aksi takdirde aradaki mesafe ölçümlerinden yararlanarak aracın güvenli bir takip yapması mümkün değildir. MCS'de araçların V2V ve V2I haberleşebilme kabiliyetleri sayesinde gerekli olan parametreleri aralarında haberleşerek geliştirilen algoritmalar mümkün hale gelmektedir.

Önceki çalışmalarda önerilen güvenli bir takip algoritmasında ise aracın yavaşlama veya hızlanma mantığı ele alınarak geliştirilmiş bir metot olduğu için sonucunda ardışık hızlanma ve yavaşlama olması kaçınılmaz olup bunun için ek olarak filtreleme çözümleri sunulmuştur. Bu çalışma kapsamında mevcut hızlanma ve yavaşlama eylemlerine ek olarak aracın sabit hızla gideceği koşul da eklenerek geliştirilecek olan algoritma ile birlikte bu istenmeyen durumun çözüldüğü gözlemlenmiştir. Geliştirilen algoritma ise adım adım aşağıdaki gibi bulunmuştur:



Şekil 3.7: Araç takibi a) önceki durum b) sonraki durum.

Şekil 3.7'de olduğu gibi öndeki araç B takip eden araç ise A olarak belirlenmiş ve takip algoritması buna göre oluşturulmuştur.

Genelleştirdiğimiz ayrık zaman formülünden Denklem (3.19.d)'de $n+1$ yerine, aracın durmuş olacağı n_{stop} adımı yazılırsa aracın duracağı konum olan Denklem (3.20) bulunabilir.

$$x[m+n_{stop}] = x[m] + \tau \cdot (n_{stop}) \cdot \left(v[m] - \frac{(n_{stop}-1)}{2} \cdot a_{dec} \cdot \tau \right) \qquad (3.20)$$

Konumu bulmadan önce bulunması gereken bilinmeyen ise aracın duracağı adım sayısı n_{stop} belirlenmelidir. Aracın konumu bulmamızı sağlayacak olan n_{stop} adımını bulmak için ise Denklem (3.17)'de $n+1$ yerine n_{stop} yazılarak sıfıra eşitlenmesi yeterli olacaktır. Böylelikle gerekli işlemler yapıldığında aracın duracağı adım sayısı Denklem (3.21b) olarak bulunmaktadır.

$$v[m+n_{stop}] = v[m] - (n_{stop}) \cdot a_{dec} \cdot \tau \qquad (3.21)$$

$$v[m+n_{stop}] = 0 \qquad (3.21.\text{a})$$

$$n_{stop} = \frac{v[m]}{a_{dec} \cdot \tau} \qquad (3.21.\text{b})$$

Artık araçların hangi adımda ve hangi konumda duracağı bulunabilmektedir. Ayrıca araçlar güvenli şekilde takip işlemini gerçekleştirebilmesi için birbirine güvenli bir mesafeye kadar (δ) yaklaşabilmeli ve bu mesafeden sonra aracın artık yavaşlaması gerekmektedir. Bu mesafeye ek olarak bir sönümleme alanı (φ) oluşturulmalı, araç bu alanda ne hızlanmalı ne de yavaşlamalıdır. Bu işlem yapılmadığı takdirde araç sürekli ve ardışık olarak artan ve azalan hareket gösterecektir. Bu ise istenmeyen bir durum olduğu için bunu tolere edecek sönümleme alanı φ bu vazifeyi görmektedir.

$$\Delta x_{min} = \delta + x_{size} \qquad (3.22)$$

$$\Delta x_{norm} = \delta + \varphi + x_{size} \qquad (3.23)$$

Her iki aracın durdukları andaki adımları n_{stop-A} ve n_{stop-B} olarak tanımlanırsa araçlar $x_A[m+n_{stop-A}]$ ve $x_B[m+n_{stop-B}]$ konumlarında durmuş olacaktır.

Araçların duracakları konumların farkları alındığı taktirde her iki aracın durur vaziyetteki mesafe farkı Δx_{stop} bulunacaktır.

$$\Delta x_{stop} = x_B[m + n_{stop-B}] - x_A[m + n_{stop-A}] \tag{3.24}$$

Bulunan Denklem (3.20), Denklem (3.24)'te açılımlarıyla birlikte yazarak sadeleştirilirse Denklem (3.25.b) elde edilmektedir.

$$\Delta x_{stop} = \begin{pmatrix} \left(x_B[m] + \tau \cdot (n_{stop-B}) \cdot \left(v_B[m] - \frac{(n_{stop-B} - 1)}{2} \cdot a_{dec} \cdot \tau\right)\right) - \\ \left(x_A[m] + \tau \cdot (n_{stop-A}) \cdot \left(v_A[m] - \frac{(n_{stop-A} - 1)}{2} \cdot a_{dec} \cdot \tau\right)\right) \end{pmatrix} \tag{3.25}$$

$$\Delta x_{stop} = x_B[m] - x_A[m] + \tau \cdot \begin{pmatrix} \left((n_{stop-B}) \cdot \left(v_B[m] - \frac{(n_{stop-B} - 1)}{2} \cdot a_{dec} \cdot \tau\right)\right) - \\ \left((n_{stop-A}) \cdot \left(v_A[m] - \frac{(n_{stop-A} - 1)}{2} \cdot a_{dec} \cdot \tau\right)\right) \end{pmatrix} \tag{3.25.a}$$

$$\Delta x_{stop} = x_B[m] - x_A[m] + \frac{1}{2} \cdot \left(\left(\frac{v_B[m]^2}{a_{dec}} + v_B[m] \cdot \tau\right) - \left(\frac{v_A[m]^2}{a_{dec}} + v_A[m] \cdot \tau\right)\right) \tag{3.25.b}$$

Sonuç itibariyle elde ettiğimiz Denklem (3.25.b), her iki aracın durduğu andaki konumlarının farkını vermektedir.

Birbirini takip edecek olan araçlar için araçlar henüz durmadan en kötü senaryo olarak her iki aracın durma anındaki konumları değerlendirilip birbirlerine çarpmayacak şekilde yönlendirme yapılması mümkündür.

Bu fark aşağıdaki gibi koşula bağlandığı takdirde arkadaki aracın bir öndeki aracı 3 şekilde takip edebilmektedir:

1.Durum: $[\Delta x_{stop} > \Delta x_{norm}]$ ise araç HIZLANIR

2.Durum: $[\Delta x_{norm} \geq \Delta x_{stop} > \Delta x_{min}]$ ise araç SABİT HIZ ile ilerlerler

3.Durum: $[\Delta x_{stop} \leq \Delta x_{min}]$ ise araç YAVAŞLAR

Algoritma:

$if\,[\Delta x_{stop} > \Delta x_{norm}]$

$\quad v_A[n+1] = v_A[n] + a_{acc} \cdot \tau$

$else\ if\,[\Delta x_{norm} \geq \Delta x_{stop} > \Delta x_{min}]$

$\quad v_A[n+1] = v_A[n]$

$else$

$\quad v_A[n+1] = v_A[n] - a_{dec} \cdot \tau$

Şekil 3.8: Takip Algoritması.

Şekil 3.8'deki algoritmanın geliştirilmesiyle birlikte, sönümleme alanının büyüklüğünün doğru şekilde tespit edilmesi büyük önem taşımaktadır. İleride gerçekleştirilecek sistem algoritma testlerinde de görüleceği üzere, bu alanın gereğinden fazla büyümesi sistemde aksamalara yol açabilmektedir. Bu nedenle, sönümleme alanının alabileceği minimum değer titizlikle belirlenmeli ve uygulanmalıdır. Minimum değer, doğrudan aracın hızıyla ilişkilidir. Araçların ardışık olarak hızlanma ve yavaşlama hareketleri yaptığı durumlar dikkate alındığında, sönümleme alanı öyle bir seviyede olmalıdır ki araç, bir adım önce yavaşladıktan sonra bir sonraki adımda hızlanmak yerine sönümleme alanına girerek hızını sabit tutabilsin.

Seçilebilecek minimum φ değerinin tespit edilebilmesi için Denklem $(3.25b)$'de üç farklı senaryo incelenecektir. Aşağıdaki araçlar aynı hizada ancak farklı şeritlerdedir.

1. Senaryo: İki Aracın da maksimum hız 20 m/s ile başlangıç noktasında olduğunu varsaydığımız bu durumdaki aracın durma mesafesi $\Delta x_{stop} = 0$ olacaktır. Yani araçlar aynı hizada duracaklardır.
2. Senaryo: Birinci araç başlangıç konumunda 20 m/s, ikinci araç yine başlangıç konumunda 19,97 m/s hız olarak belirlediğinde $\Delta x_{stop} = 0{,}2\ m$ olacaktır.
3. Senaryo: Birinci Araç başlangıç konumunda 20 m/s, ikinci araç yine başlangıç konumunda 19,94 m/s hız olarak belirlendiğinde $\Delta x_{stop} = 0{,}3997\ m$ olarak hesaplanmaktadır.

Araçların durma anındaki konum farkı olan Δx_{stop}; iki aracın hız farkı, yavaşlama süreci ve başlangıç konumları hesaba katılarak ifade edilmesidir. Ancak sönümleme alanı φ, yalnızca hız farkı ve yavaşlama süreciyle ilişkilidir ve takip algoritmasındaki stabil hız alanını sağlaması için gereken minimum mesafeyi temsil etmektedir; dolayısıyla, başlangıç konumları φ'nin hesaplanmasında bir etkiye sahip değildir.

Senaryolardaki φ'nin Anlamı:

- 2. senaryoda $\Delta x_{stop} = 0{,}2$ değeri aynı zamanda φ 'nin hız farkı nedeniyle sabit hızda kalınması gereken minimum mesafe olduğunu gösterir. Araçların hız farkı (20 m/s ve 19,97 m/s) bu değerin bir periyotta dengelenmesini sağlar.
- 3. senaryoda Δx_{stop}=0,3997 m ise hız farkının (20 m/s ve 19,94 m/s) iki periyotta dengelenmesi gerektiğini ifade eder. Bu durumda φ değeri, Δx_{stop}'un birden fazla periyotluk bir sönümleme alanına işaret ettiğini belirtir.

İlişkilendirme: φ, sistemdeki hız farkının bir veya birden fazla periyotta sabit hız koşulunu sağlayabilecek minimum değer olarak kabul edilir. Bu durumda:

- 2. Senaryoda $\varphi = 0{,}2\ m$ alınabilir, çünkü tek bir periyot yeterlidir.
- 3. Senaryoda $\varphi = 0{,}2\ m$ kalabilir çünkü sistemin hız farkını dengelemesi için bu mesafenin birden fazla periyot boyunca etkili olması gerektiği görülmektedir. Dolayısıyla, burada Δx_{stop}' un büyüklüğü, φ'nin iki periyot boyunca etkili olacağını ifade eder.

Bu sonuç, φ'nin değerinin araçların hız farkına ve bu farkın kaç periyot süreceğine göre ölçeklenebileceğini gösterir. 3. senaryoda, φ, $\Delta x_{stop} = 0{,}3997\ m$ değeri üzerinden, birden fazla periyottaki hız dinamiklerinin toplam etkisini hesaba katması gerektiğini ortaya koymaktadır. Bu nedenle φ, Denklem (3.26) olarak bulunur.

$$\varphi = \frac{\Delta x_{stop}}{n_{periyot}} \tag{3.26}$$

Burada $n_{periyot}$, hız farkının kaç periyot sürede dengelendiğini ifade eder. 3. senaryoda $n_{periyot} = 2$ olarak kabul edilir ve $\varphi \cong 0{,}2\ m$ değeri bulunur. Bu ifadeyi daha genel bir formülle ifade edilecek olursa araçların duracağı adım sayılarının farklarının $(n_{stop-B} - n_{stop-A})$ alınması yeterli olacaktır.

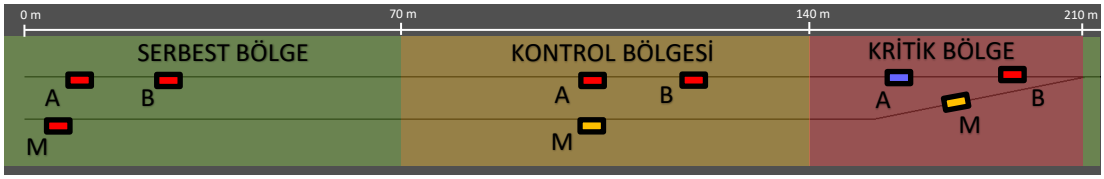
Sonuç olarak genel bir formülle φ değerini ifade edecek olursak, Denklem (3.25.b)'deki başlangıç koşullarının ihmal edilmesiyle birlikte alınabilecek minimum φ değeri bulunmuş olur.

$$\varphi_{min} = \frac{\frac{1}{2} \cdot \left(\left(\frac{v_B[m]^2}{a_{dec}} + v_B[m] \cdot \tau \right) - \left(\frac{v_A[m]^2}{a_{dec}} + v_A[m] \cdot \tau \right) \right)}{\frac{v_B[m] - v_A[m]}{a_{dec} \cdot \tau}} \tag{3.27}$$

3.5 Birleşme Algoritmaları

Sisteme dahil olacak olan araçların problemsiz bir şekilde tali yoldan anayola entegre olması gerekmektedir. Tali yoldan gelen araçların anayol üzerindeki mevcut araçlarla çarpışma riski olmadan, trafik akışını kesintisiz bir şekilde sürdürebilmesi kritik önem taşır. Bu doğrultuda geliştirilen birleşme algoritmaları, araçların düğüm noktalarına ulaşmadan önce hız, konum ve şerit bilgilerini diğer araçlarla paylaşmasına ve buna göre davranmalarına dayanır. Algoritma, bu bilgiler doğrultusunda her bir aracın hızını ve pozisyonunu optimize ederek, giriş yapan araçların mevcut trafik akışına uyum sağlamasını hedefler. Böylece, trafik yoğunluğunun yüksek olduğu senaryolarda dahi güvenli ve verimli bir birleşme süreci sağlanır. Bu algoritmalar iki şeridin birleşmesi esnasında izlenmesi gereken adımlar silsilesinden oluşmaktadır. Bu metot sayesinde araçların çarpışmadan en uygun hızda ilerlemesi amaçlanmıştır.

Birleşme algoritmaları, araçların geçiş yaptığı alanları belirli bölgelere ayırarak güvenli ve verimli bir trafik akışı sağlamayı hedefler. Bu bölgeler, araçların hız ve mesafe optimizasyonu açısından kritik öneme sahiptir. Her bir bölge, farklı görev ve sorumluluklara sahip olup algoritmanın uygulanabilirliğini kolaylaştırmaktadır.



Şekil 3.9: Yol bölümleri.

Birleşme algoritmasının temel prensibini oluşturan bölge sistemi birleşme düğümlerinden tali araç girişlerine kadar olan kısımda kritik bir rol üstlenmektedir. Bahsedilen birleşme alanını kapsayan alan Şekil 3.9'da görüldüğü gibi Serbest Bölge, Kontrol Bölgesi ve Kritik Bölge olmak üzere 3 bölgeden oluşmaktadır.

Birleşme algoritmalarında, hız ve mesafe optimizasyonu açısından bu bölgelerin uzunlukları, sistemin temel fiziksel parametrelerine dayalı olarak belirlenmiştir. Birleşmenin geleceği alanda bu bölgelerin 70 metre uzunluğunda seçilmesi, 0 m/s hızdan başlayıp sistemin maksimum hızı olan 20 m/s hıza ulaşmaları ya da maksimum hızdan itibaren yavaşlayarak durabileceği minimum mesafe, adım aralığı $\tau = 0.01$ sn ve hızlanma ve yavaşlama ivmesi 3 m/s² olarak belirlenen bir sistemde yaklaşık 67 metre olmasıdır.

Serbest Bölge: Bu bölgede hem anayol hem de tali yoldaki araç için hiçbir kısıtlama olmaksızın kendi şeridinde takip algoritmasına göre yollarında seyir etmektedir. Ayrıca Kontrol Bölgesi ve Kritik bölge dışındaki bütün alanlar da Serbest Bölge olarak kabul edilmektedir.

Bu bölge yolun girişinden itibaren ilk 70 metrelik alanı ihtiva etmektedir.

Kontrol Bölgesi: Bu kısımda ise tali yoldaki araç için arkasına dahil olacağı ana yoldaki aracı seçme ve takibe alma faslıdır birleşme algoritması bu alanda devreye girer ve bu bölge sonunda tamamlanmış olur. Bu alan ise serbest bölgenin bitiminden itibaren 70 metrelik alanı kapsamaktadır.

Kritik Bölge: Birleşme düğümünden önceki son 70 metredeki bölge olan bu kısımda sıralama işlemi tamamlanmış olan iki farklı şeritteki araçlar sanki tek şeritte gibi davranarak takip algoritmasını uygularlar.

3.5.1 Orijinal Birleşme Algoritmasının İyileştirilmesi

Önceki çalışmada önerilmiş olan eşitsizlik Denklem (3.28) olarak verilmiştir.

$$x_M - x_B - \frac{v_A(x_M - x_B)}{v_B} \leq \Delta x_{min} \tag{3.28}$$

Bu eşitsizlik sayesinde, önündeki araç düğüm noktasına ulaştığında, arkadaki aracın konumu hesaplanır ve elde edilen sonuç, Δx_{min} ile karşılaştırılarak hızın artırılması veya azaltılması kararı verilir (Bozuyla 2019).

Algoritma:

$if \left[x_M - x_B - \frac{v_A(x_M - x_B)}{v_B} \leq \Delta x_{min} \right]$

$v_A[n+1] = v_A[n] - a_{dec} \cdot \tau$

$else$

$v_A[n+1] = v_A[n] + a_{dec} \cdot \tau$

Şekil 3.10: Orijinal birleşme algoritması.

Şekil 3.10'dan da anlaşılabileceği üzere bu algoritma sadece hızlanma ve yavaşlama kabiliyetine sahip olduğundan takip algoritmasında olduğu gibi istenmeyen ardışık durum burada da meydana gelmektedir. Bu tez kapsamında geliştirilen takip algoritmasında önerilen φ değeri kullanılarak iyileştirilmesi ve bu istenmeyen durumun giderilmesi mümkündür.

Denklem (3.28) eşitsizliğinden yararlanılarak iki durumlu olan birleşme algoritmasını alternatif bir çözüm olarak üç durumda çalışması istendiğinde algoritma Şekil 3.11'deki gibi olmaktadır.

Algoritma:

$$if \left[x_M - x_B - \frac{v_A(x_M - x_B)}{v_B} \leq \Delta x_{min} \right]$$

$$v_A[n+1] = v_A[n] - a_{dec} \cdot \tau$$

$$else\ if \left[\Delta x_{norm} \geq x_M - x_B - \frac{v_A(x_M - x_B)}{v_B} > \Delta x_{min} \right]$$

$$v_A[n+1] = v_A[n]$$

$$else$$

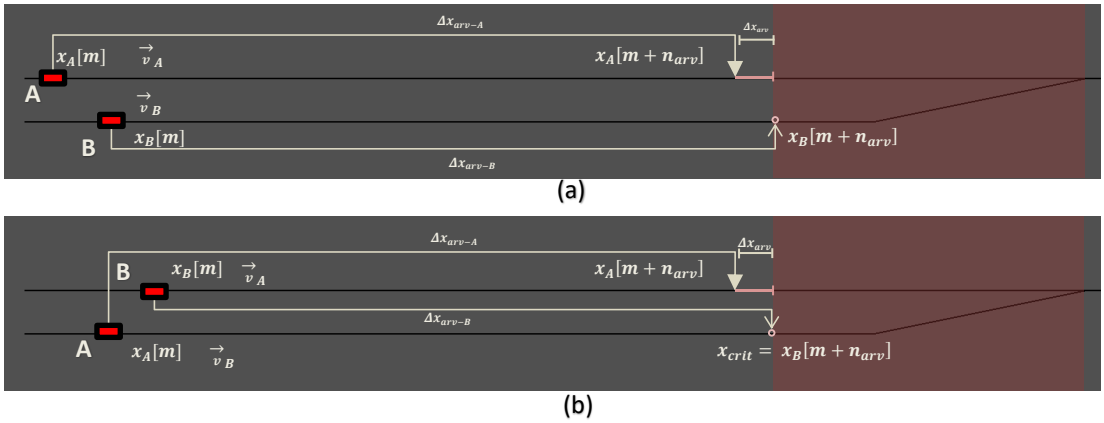
$$v_A[n+1] = v_A[n] + a_{dec} \cdot \tau$$

Şekil 3.11: İyileştirilmiş birleşme algoritması.

3.5.2 Varış Tahmini Birleşme Algoritması

Varış tahmini birleşme algoritması, tali yoldan anayola giriş yapan araçların güvenli ve düzenli bir şekilde trafik akışına entegre olmasını sağlamak için tasarlanmıştır. Algoritma, araçların konum ve hız bilgilerinden faydalanarak, kritik bölgeye varış sürelerini (n_{arv}) tahmin eder ve buna göre hız ayarlamaları yapar. Araçlar, serbest bölgede hareket ederken, öncelikle linkin ortalama hızına ulaşmayı hedefler. Kontrol bölgesinde ise algoritma, araçların sıralama işlemlerini gerçekleştirerek birleşme sürecini planlar. Kritik bölgeye ulaşıldığında araçlar, sıralanmış bir şekilde birleşme işlemini bekler ve bu bölgeyi tek bir şerit gibi kullanarak birleşme noktalarına yönelir. Bu yapı, araçların birleşme sırasında çarpışma riskini önlemek ve minimum takip mesafesini korumak amacıyla hızlanma veya yavaşlama kararlarını kontrol eder. Böylece, trafik akışı kesintisiz ve güvenli bir şekilde gerçekleştirilmiş olur.

Şekil 3.12'de A ve B araçlarının her iki durumlarına göre varış tahmini birleşme algoritmasını temsil etmektedir.



Şekil 3.12: Varış tahmini a) tali yoldaki araç önde b) anayoldaki araç önde.

B aracın kritik bölgeye varış adım sayısı olan n_{arv} kolay bir şekilde bulunabilmektedir.

$$x_B[m+n_{arv}] = x[m] + \tau \cdot (n_{arv}+1) \cdot v_B \tag{3.29}$$

$$\Delta x_{arv-B} = x_B[m+n_{arv}] - x_B[m] \tag{3.29.a}$$

$$\Delta x_{arv-B} = \tau \cdot (n_{arv}+1) \cdot v_B \tag{3.29.b}$$

$$n_{arv} = \frac{\Delta x_{arv-B}}{\tau \cdot v_B} - 1 \tag{3.29.c}$$

$$x_A[m+n_{arv}] = x[m] + \tau \cdot (n_{arv}+1) \cdot v_A \tag{3.29.d}$$

A aracının n_{arv} adım sonraki konumu Denklem (3.29.d) hesaplanır.

$$\Delta x_{arv} = x_B[m+n_{arv}] - x_A[m+n_{arv}] \tag{3.30}$$

Denklem (3.30)'da ifade edilen iki aracın n_{arv} adım sonraki konumlarının farkı Δx_{arv} bulunarak Δx_{min} ile karşılaştırılır. Eğer Δx_{arv} büyük olması durumunda hızlanır diğer durumlarda ise araç yavaşlar. Bu şekilde araçların kritik noktaya varma anındaki mesafeleri en az Δx_{min} olacak şekilde belirlenmiş olur. Önerilen bu varış tahmini birleşme algoritması Şekil 3.13'te ve bu algoritmanın akış şeması Şekil 3.14'te verilmiştir.

Algoritma:

$if\,[\Delta x_{min} \geq \Delta x_{arv}]$

$$v_A[n+1] = v_A[n] - a_{dec} \cdot \tau$$

$else\ if\,[\Delta x_{norm} \geq \Delta x_{arv} > \Delta x_{min}]$

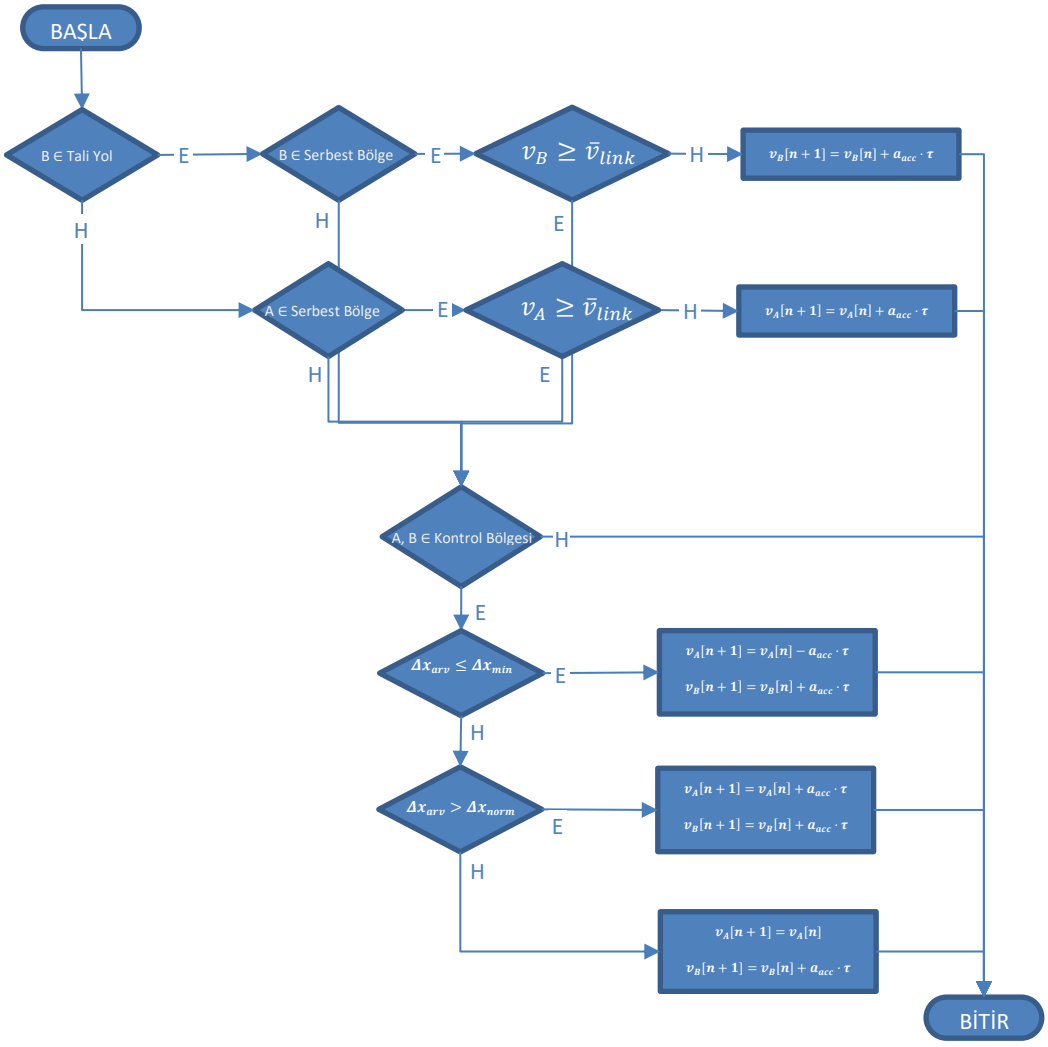
$$v_A[n+1] = v_A[n]$$

$else$

$$v_A[n+1] = v_A[n] + a_{dec} \cdot \tau$$

Şekil 3.13: Varış tahmini birleşme algoritması.

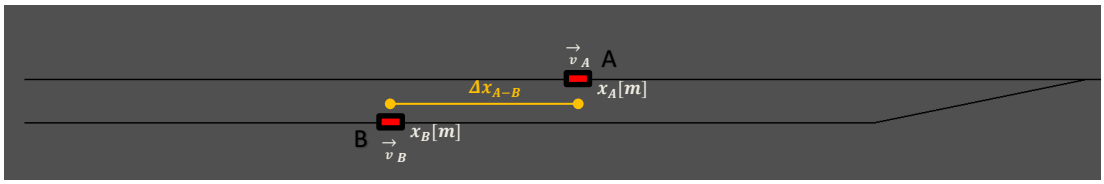
Bu adımlarda maksimum hızı geçen araçların hızları tekrar maksimum hıza eşitlenir.



Şekil 3.14: Varış tahmini birleşme algoritması akış şeması.

3.5.3 Test Birleşme Algoritması

Test birleşme algoritması, daha önce tanıtılan varış tahmini birleşme algoritmasına kıyasla daha basit bir yapı sunarak, araçların birleşme davranışlarını temel mesafe ve hız ilişkilerine dayalı olarak modellemek amacıyla geliştirilmiştir. Bu algoritmanın temel amacı, karmaşık tahmin modellerinin olmadığı senaryolarda araçların insan davranışlarına benzer bir şekilde karar vererek hareket etmelerini sağlamaktır. Ayrıca, sistemin daha önceki algoritmayla karşılaştırmalı olarak analiz edilmesine ve performans farklarının değerlendirilmesine imkân tanır. Test algoritması, araçların arasındaki mesafeye bağlı olarak hızlanma, yavaşlama veya sabit hızla hareket etme gibi temel karar mekanizmalarını içerir.



Şekil 3.15: Test birleşme algoritması.

Bu algoritma, iki araç arasındaki mesafeye bağlı olarak arkadaki aracın tepki verdiği bir yaklaşıma dayanır. Şekil 3.15'te gösterildiği gibi, hesaplanmış tahminler olmaksızın insan davranışına benzer bir şekilde çalışmaktadır.

İlk adım olarak araçlar arasındaki mesafe Denklem (3.31) bulunur.

$$\Delta x_{A-B} = x_B[m] - x_B[m] \tag{3.31}$$

Bu mesafenin minimum takip mesafesinden az olduğu durumlarda araç yavaşlar, fazla olduğunda hızlanır ve eşit olduğunda sabit hızda ilerler. Algoritmanın işleyişi Şekil 3.16'da verilmiş, akış şeması ise Şekil 3.17 ile detaylandırılmıştır.

Algoritma:

$if\,[\Delta x_{min} > \Delta x_{A-B}]$

$$v_A[n+1] = v_A[n] - a_{dec} \cdot \tau$$

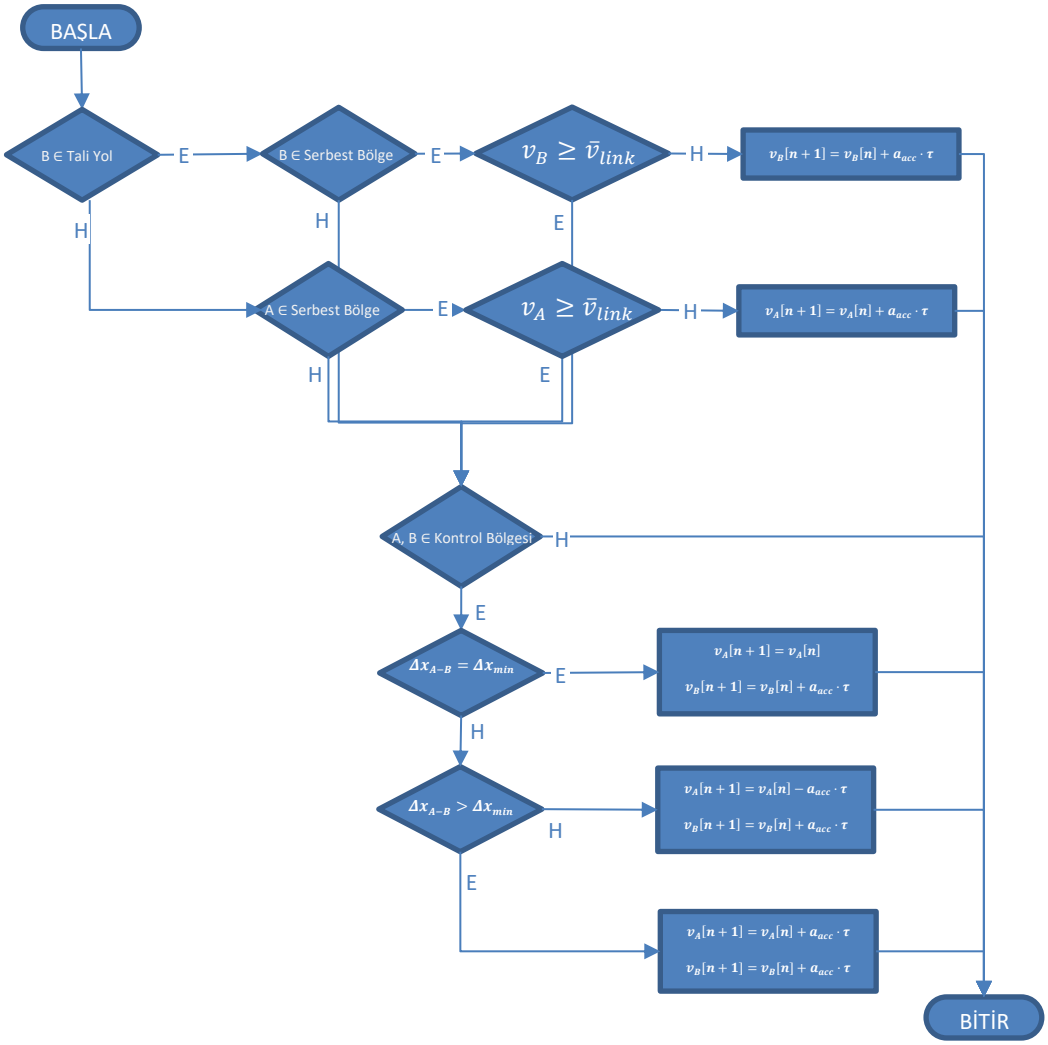
$else\ if\,[\Delta x_{min} < \Delta x_{A-B}]$

$$v_A[n+1] = v_A[n] + a_{acc} \cdot \tau$$

$else$

$$v_A[n+1] = v_A[n]$$

Şekil 3.16: Test birleşme algoritması.



Şekil 3.17: Test birleşme algoritması akış diyagramı.

3.6 Sistem Varsayılanları

Sistemin etkin ve güvenli bir şekilde çalışabilmesi için temel sistem varsayılanlarının dikkatlice belirlenmesi gerekmektedir. Bu sabitler, elektrikli araçların fiziksel ve performans özellikleri ile sistemin hızlanma, yavaşlama ve takip mesafesi, simülasyon ortamının özellikleri gibi parametreleri kapsar. Elektrikli araçlara dayalı bir çözüm olduğundan, sistemin temel varsayımları ve değişkenlerinin, piyasa koşulları ve sınırlamaları göz önünde bulundurularak ön araştırma ile belirlenmesi isabetli olacaktır.

3.6.1 Maksimum Hız ve İvmeler

Bu kapsamda, piyasada bulunan kompakt elektrikli araçların özellikleri değerlendirilmektedir.

Tablo 3.1: Bazı elektrikli araç özellikleri

Model	Uzunluk	Genişlik	0-100 km/h	Yolcu
e.Go e.wave X	3360 mm	1814 mm	12.0 sn	4
Renault Twingo Electric	3615 mm	1646 mm	12.6 sn	4
Fiat 500e Hatchback	3631 mm	1900 mm	9.0 sn	4
Mini Cooper SE	3850 mm	1727 mm	7.3 sn	4
Honda e	3894 mm	1752 mm	9.0 sn	4
Renault Zoe ZE50 R110	4087 mm	1787 mm	11.4 sn	5

Tablo 3.1'de verilen bu araçlar hem kompakt yapıları hem de hızlanma ve yavaşlama performansları açısından MCS için referans niteliğindedir. Özellikle hızlanma ivmesi hesaplamalarında, araçların 0-100 km/s hızlanma süreleri dikkate alınmıştır. Buna göre, en düşük hızlanma ivmesi 2,3 m/s^2, en yüksek hızlanma ivmesi ise 3,8 m/s^2 olarak tespit edilmiştir. MCS için önerilecek hızlanma ivmesi ise bu değerlerin ortalamasına yakın olan 3 m/s^2 olarak belirlenmiştir.

Yavaşlama ivmesi değerleri konusunda yapılan araştırmalar da bu seçimi desteklemektedir. Ulaştırma Mühendisleri Enstitüsü (*Institution of Transportation Engineers*) ve Amerikan Eyalet Karayolu ve Ulaştırma Yetkilileri

Birliği (*American Association of State Highway and Transportation Officials*, AASHTO) gibi otoritelerin önerdiği yavaşlama ivme değerleri dikkate alınmıştır:

- *Institution of Transportation Engineers* (1999): Önerilen yavaşlama ivmesi 3,0 m/s² (Zhu ve diğ. 2018).
- AASHTO (2018): Önerilen yavaşlama ivmesi 3,4 m/s² (AASHTO 2018).

Bu bağlamda, MCS için önerilen yavaşlama ivmesi de 3 m/s² olarak seçilmiştir. Bu değer, sistemin hem güvenliği hem de konforu açısından ideal bir denge sunmaktadır. Sistemdeki araçların maksimum çıkabileceği hız ise önceki çalışmalardaki gibi 20 m/s alınmıştır.

3.6.2 Güvenli Takip Mesafesi ve Sönümleme Alanı

Sistemdeki araçların güvenli takip mesafesi δ, önceki çalışmalardaki gibi 2m alınmıştır.

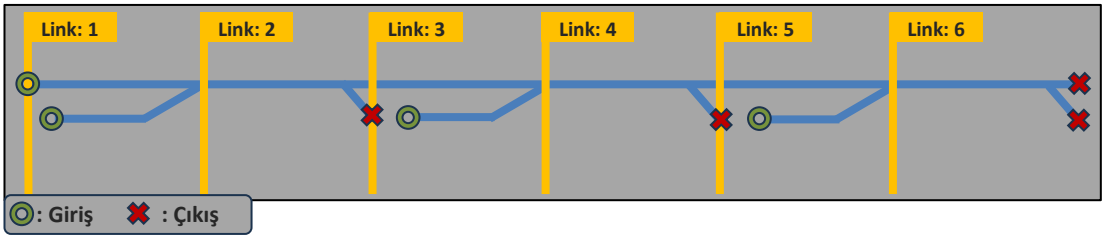
Her bir araç için özel ve değişken bir φ seçmek yerine maksimum hızı 20 m/s olarak belirlenen bir sistemde Denklem (3.27) incelendiğinde herhangi bir aracın alabileceği φ_{min} değerinin azami değeri $\varphi_{min} = 0{,}2$ olacaktır. Bu sebepten dolayı sistemde sabit bir φ değeri seçilmesi durumunda $\varphi = 0{,}2$ en uygun değerdir. Ancak bazı sistem testleri φ değerinin farklı seçilmesi durumunda nasıl tepki verdiğini görmek adına çeşitli testler yapılacaktır.

3.6.3 Simülasyon Genel Yapısı

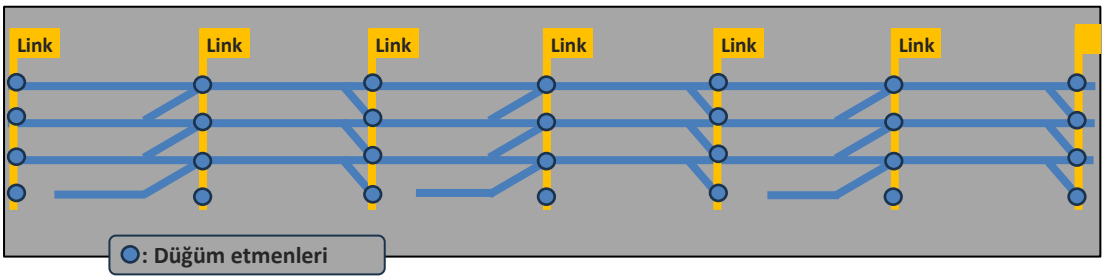
Yapılacak testlerde, MCS'nin ortam yapısı, toplamda dört giriş ve dört çıkıştan oluşan tek şeritli bir yol ağına dayanmaktadır. Bu yapı, üç tali yol girişi, üç tali yol çıkışı, bir ana yol girişi ve bir ana yol çıkışını içerdiği Şekil 3.18'de görülmektedir.

Ancak, önerilen MCS'de yollar yalnızca tek şeritten ibaret değildir. Şekil 3.19'da gösterildiği gibi, sistem üç şeritli bir yapıya sahip olmasına rağmen, bu tezin

kapsamı yalnızca takip ve birleşme algoritmalarını ele aldığından, çoklu şerit benzetimine ihtiyaç duyulmamıştır.



Şekil 3.18: Python simülasyon benzetim ortamı.



Şekil 3.19: Üç şeritli MCS ortamı.

Simülasyonun bu şekilde tasarlanmasının temel nedeni, deney süresinin aşırı uzamasını önlemektir. Bununla birlikte, tercih edilen giriş ve çıkış sayısı, simülasyonun gerçekçiliğini artıran önemli bir faktördür. Daha fazla giriş ve çıkış noktası, MCS'nin gerçek trafik koşullarını daha doğru şekilde yansıtmasına olanak sağlar. Ancak, Şekil 3.19'da gösterilen üç şeritli yapı gibi daha karmaşık bir model kullanılması, simülasyonun hesaplama süresini önemli ölçüde artıracağından, bu çalışma kapsamında daha sade bir yaklaşım benimsenmiştir.

3.6.4 Tüm Sistem Sabitleri

Son olarak, test sürecinde kullanılan ve sistemin temel çalışma prensiplerini belirleyen tüm sabitler Tablo 3.2'de verilmiştir. Bu sabitler, algoritmaların doğruluğunu ve tutarlılığını sağlamak adına dikkatlice seçilmiş ve simülasyon boyunca sabit tutulmuştur.

Tablo 3.2: Sistem sabitleri

τ	= 0,01 sn	Giriş Yolu	= 210 m
δ	= 2 m	Çıkış Yolu	= 70 m
φ	= 0,2 m	Aracın Maksimum Hızı	= 20 m/s
a_{acc}	= 3 m/s^2	Serbest Bölge Uzunluğu	= 70 m
a_{dec}	= 3 m/s^2	Kontrol Bölge Uzunluğu	= 70 m
x_{size}	= 2 m	Kritik Bölge Uzunluğu	= 70 m
σ	= 5000 τ		
μ	= 15000 τ		

3.7 Sistem Testleri

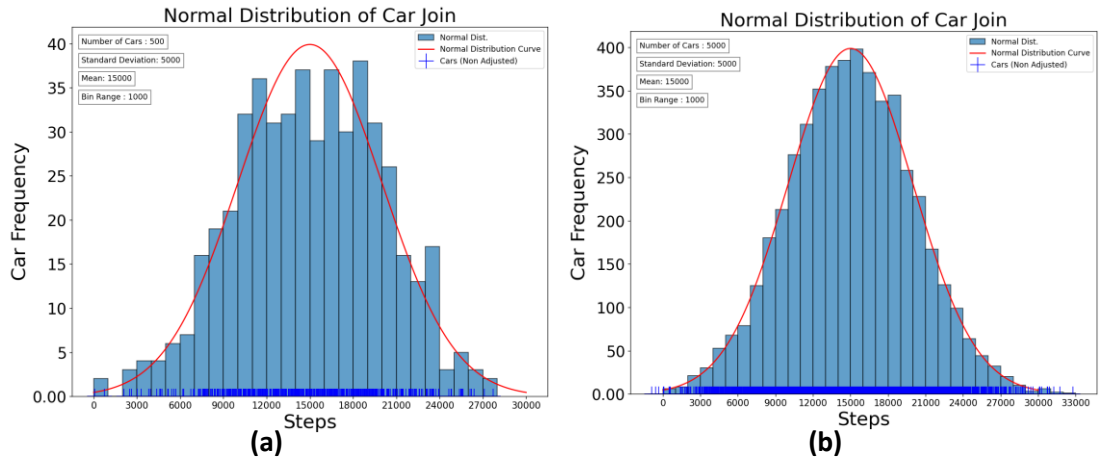
Bu tez kapsamında geliştirilmiş olan algoritmaların *Python* ile yapılan testleri ve bu testler için geliştirilen veri setlerinin işlem süreci şöyle gerçekleştirilmiştir:

3.7.1 Veri Setleri

Bir simülasyonda kullanılan veri setleri, sistemin doğruluğunu ve güvenilirliğini belirleyen temel unsurlardan biridir. Doğru ve iyi yapılandırılmış veri setleri, sistemin performansını değerlendirmek, farklı senaryoları test etmek ve algoritmaların etkinliğini ölçmek için gereklidir. Veri setleri genellikle geçmiş verilerden, gerçek zamanlı ölçümlerden veya simülasyonun ihtiyaçlarına uygun olarak tasarlanan sentetik verilere dayanır. Bu bağlamda, MCS simülasyonu için kullanılan veri setleri, araçların girişten ne zaman girecekleri bilgisini içermektedir.

MCS simülasyonunda kullanılan veri setleri, araçların sisteme giriş zamanlarını ve bu girişlerin trafik akışına olan etkilerini modellemek amacıyla oluşturulmuştur. Bu veri setleri, sistemin dinamik yapısını gerçekçi bir şekilde yansıtabilmek için normal dağılım metodu kullanılarak üretilmiştir. Normal dağılım, araçların giriş sıklıklarının belirli bir ortalamaya yakın şekilde dağıtılmasını sağlarken, sapma miktarı ile sistemin yoğunluk koşullarını kontrol etme olanağı tanımaktadır. Böylece, farklı trafik senaryolarını simüle etmek ve algoritmaların bu senaryolara nasıl tepki verdiğini analiz etmek mümkün olmaktadır. Bu yaklaşım hem sistemin

kararlılığını değerlendirme hem de olası darboğazların saptanmasında etkili bir rol oynamaktadır.



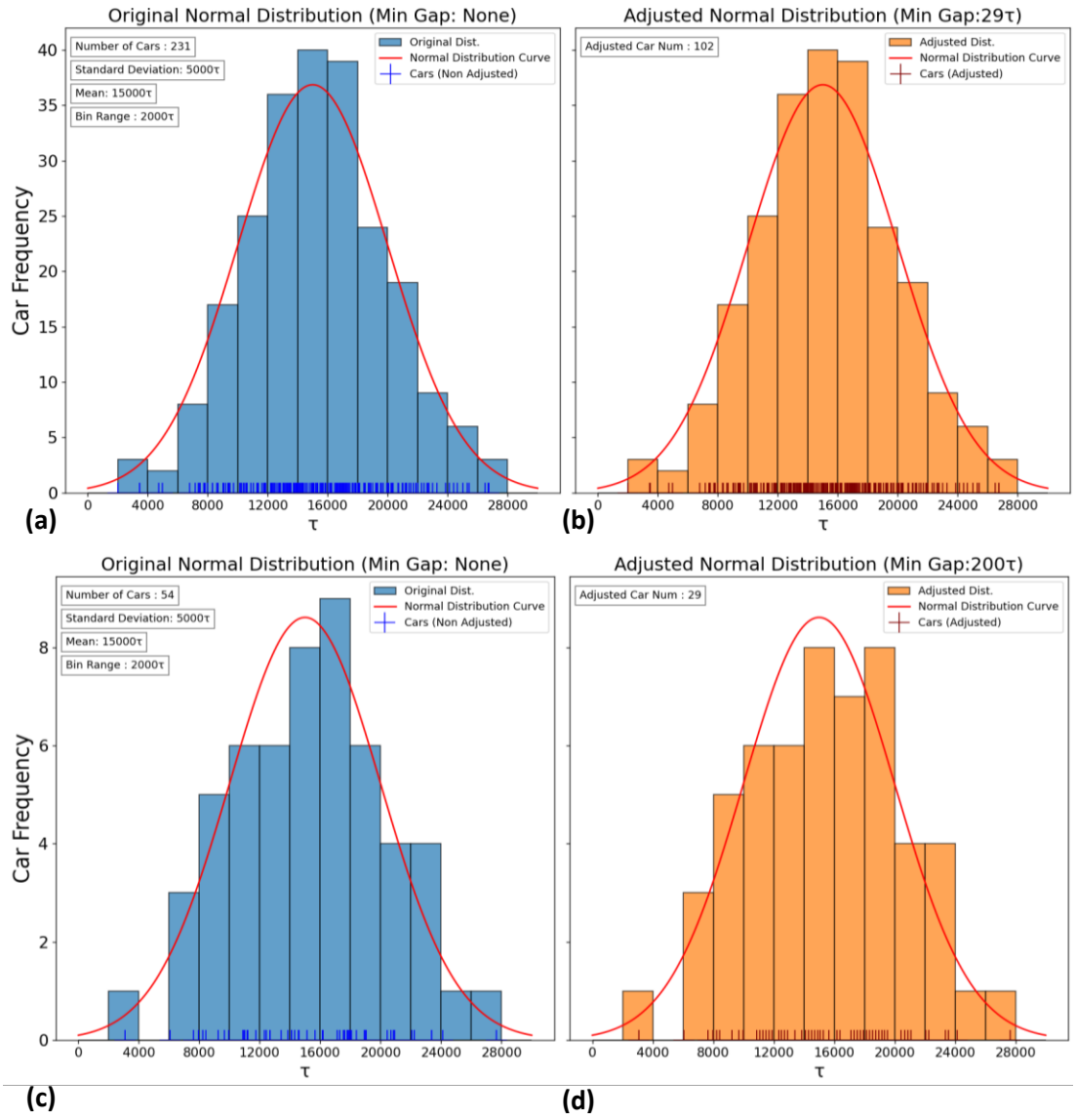
Şekil 3.20: Normal dağılım histogram grafikleri a) araç sayısı=500 b) araç sayısı=5000.

Bu analizde, gözlemlenen verilerin daha net bir şekilde yorumlanmasını sağlamak için y-ekseni araç frekansı olarak temsil edilmiştir. Histogram, belirli zaman aralıklarında sisteme dahil olan araçların sayısını göstermekte ve veri setinin bir görselleştirmesini sunmaktadır. Üzerine eklenen kırmızı eğri, teorik normal dağılımı temsil eder. Bu, gözlemlenen araç giriş dağılımını beklenen normal dağılım modeli ile karşılaştırmamızı daha basit bir hale getirmektedir. Ek olarak asıl araç girişlerinin hangi adımlarda olduğunu gösteren mavi işaretçiler de eklenmiştir.

Şekil 3.20'de oluşturulan veriler aslında bir kıyaslama amacıyla oluşturulmuştur. İdeal koşullarda araç sayısının arttırılmasıyla veri setinin normal dağılımla teorik değere yaklaşabileceği Şekil 3.20'de açıkça görülmektedir. Ancak gerek MCS simülasyon ortamının sınırlamaları gerek gerçek koşulların getirdiği sınırlamalar nedeniyle belirli zaman aralığında sisteme girebilecek araç sayısını kısıtlamaktadır.

Ayrıca normal dağılım metodu MCS simülasyonunda kullanımı bazı önlemler gerektirmektedir. Simülasyona giriş yapacak olan araçların sisteme dahil olma hızları bu veri setlerinde kritik bir rol oynamaktadır. Araç dahil olma hızlarının ardışık araç girişlerinde çarpışma olmadan ve güvenli mesafede takip edebilecekleri şekilde oluşturulabilmesi için normal dağılımdan sonra bir filtreden geçirilmesi gerekmektedir. Bu şekilde normal dağılım uygulanan sistemde yoğunluk koşullarında

araçların olabileceği en yoğun ve güvenli konumda olabilmeleri için hıza bağlı olarak minimum eklenmesi gereken gecikmeler belirlenmelidir.



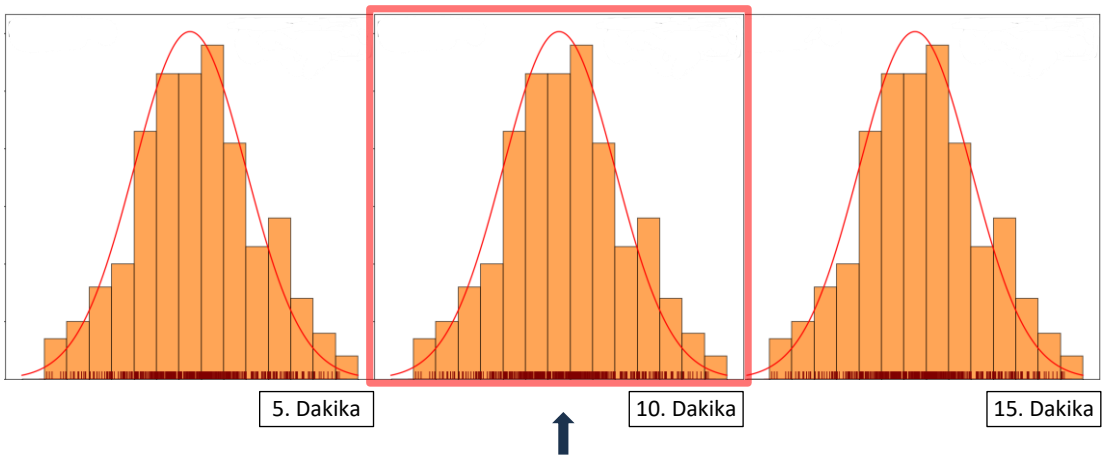
Şekil 3.21: Normal dağılım veri üretim grafiği a) ilk oluşturulan birinci veri seti b) düzenlenmiş birinci veri seti c) ilk oluşturulan ikinci veri seti d) düzenlenmiş ikinci veri seti.

Bunun için yapılan hesaplamalarda araç girişlerinde sisteme giriş hızları 0 m/s olan bir veri setinde uygulanması gereken gecikme 200 τ, eğer araç sisteme maksimum hız olan 20 m/s hızda dahil olması söz konusuysa uygulanması gereken gecikme 29 τ olarak belirlenmiştir.

Şekil 3.21'te ana yol girişlerini (Şekil 3.21.a) ve (Şekil 3.21.b) ve tali yol girişlerini (Şekil 3.21.c) ve (Şekil 3.21.d) temsil eden bir veri setinin ilk hali ile test için uygun hale getirilmiş hallerinin histogram grafikleri verilmiştir.

Oluşturulan her bir veri seti gerçek zamanda 5 dakikalık dilimi ifade eden 30000 τ'luk süreci alacak şekilde oluşturulmuş, bu veri setleri üç kez tekrarlanarak 15 dakikalık zaman dilimi test edilmiştir.

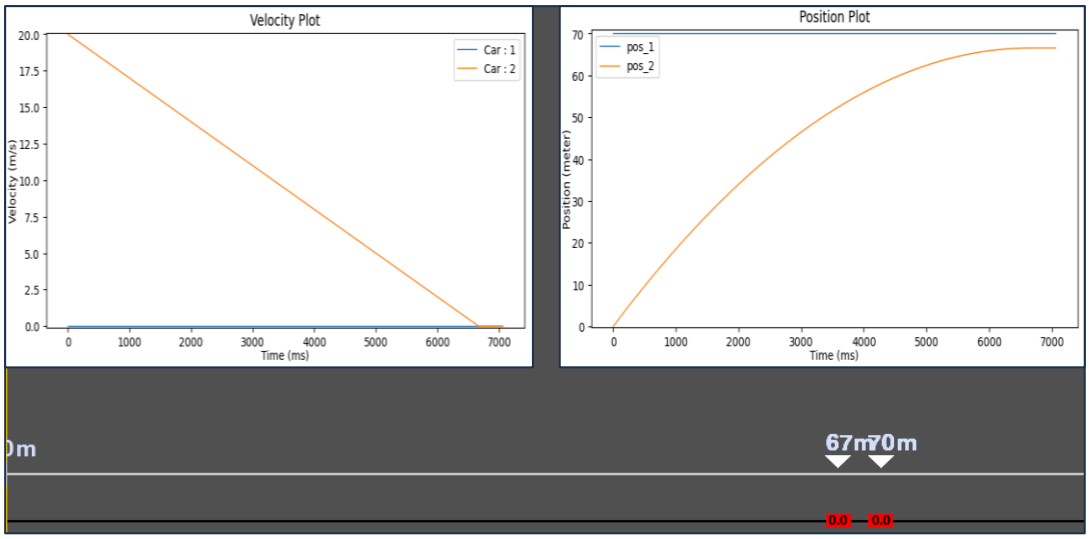
Bu çalışmada, veri setlerinin üç kez tekrarlanmasından sonra sadece ikinci tekrarın, yani ikinci 5 dakikalık kısmın analiz edilmesi, bir dizi önemli gerekçeye dayanmaktadır. Öncelikle ilk tekrarlarda sistemin başlangıç davranışları ve adaptasyon süreci etkili olabileceği gibi, son tekrarlarda sistemin yorgunluk veya doygunluk gibi dinamiklerle karşılaşması muhtemeldir. Dolayısıyla, Şekil 3.22'de gösterildiği gibi her bir veri setinin üç kez tekrar edilmesiyle oluşturulan süreçte, sadece ikinci tekrar analiz için seçilmiştir. Bu şekilde genel verimliliğin ve işleyişin daha net bir biçimde yansıtması amaçlanırken, ilk veya son bölümlerde karşılaşılabilecek olası sapmalardan kaynaklanan yanlışlıkların en aza indirgenmesi hedeflenmiştir.



Şekil 3.22: Simülasyonda 15 dakikalık zaman diliminde giren araçlar.

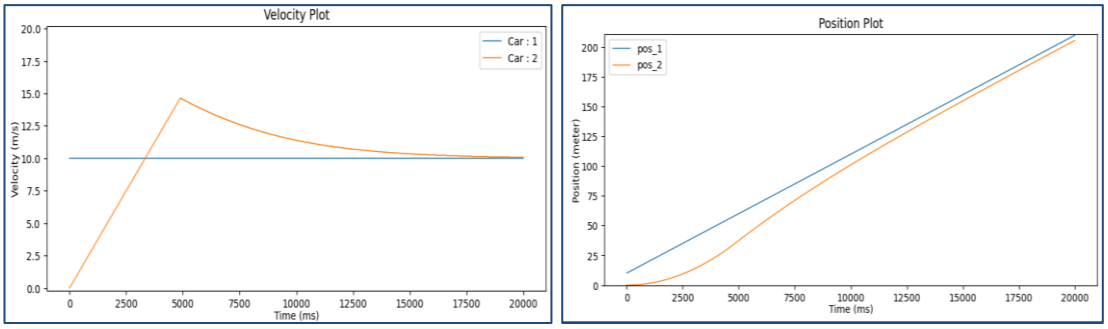
3.7.2 Takip Algoritması

Tezin bu bölümünde, MCS için geliştirilen takip algoritmasının performansı değerlendirilmiştir. Grafikler, sistemdeki araçların belirli koşullardaki davranışlarını temsil etmektedir. Bu analiz, önceki çalışmalara kıyasla algoritmanın iyileştirilmiş yönlerini sergilemekte ve φ parametresinin farklı değerlerdeki etkilerini incelemektedir. $\varphi = 0$ durumu aynı zamanda algoritmanın önceki sürümüne karşılık gelirken, yeni eklenen φ parametresiyle takip algoritmasındaki iyileştirmelere değinilecektir.



Şekil 3.23: Takip algoritması: duran araç.

Şekil 3.23'te sistemin durağan bir aracı takip etme performansı incelenmiştir. Araç, sabit bir başlangıç noktasında durmakta ve diğer araç bu aracı güvenli bir mesafeden takip etmektedir.



(a) **(b)**

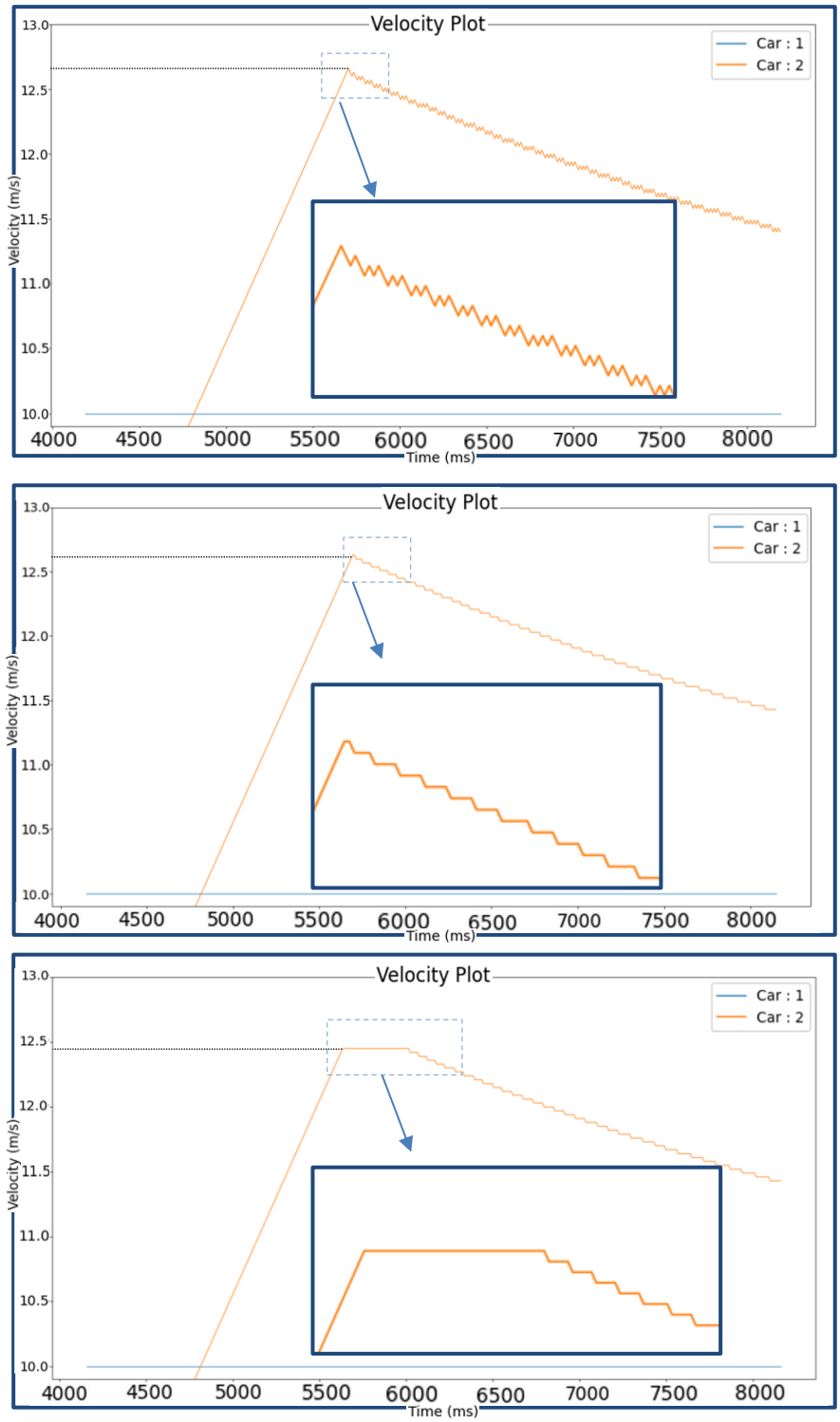
Şekil 3.24: Takip algoritması: sabit hızlı araç a) hız grafiği b) konum grafiği.

Şekil 3.24'teki grafiklerde sabit bir hızda ilerleyen bir aracı takip eden aracın hız profili verilmiştir. Bu grafikte araç, öndeki araca olan mesafesini koruyarak hızını ayarlamaktadır.

(a)

(b)

(c)



Şekil 3.25: Takip algoritması hız grafikleri a) $\varphi = 0$ b) $\varphi = 0.2$ c) $\varphi = 1$.

Araçlar arasındaki hız farkını dengelemek ve takip algoritmasındaki kararlılığı sağlamak için φ kritik bir parametre olduğundan bahsedilmişti. $\varphi = 0.2$, hız farkının (örneğin 20 m/s ve 19,97 m/s gibi) tek bir döngüde dengelenmesine olanak tanır. Bu, araçların hızlanma ve yavaşlama döngülerine girmeden stabil bir şekilde ilerlemelerini sağlar. Ancak bu alanın gereğinden fazla düşük olması ardışık hızlanma ve yavaşlama

döngüsüne sebep olduğu gibi, bu alanın fazla seçilmesi de hantallaşmaya sebebiyet vermektedir. Çünkü, optimum φ değeri ile takip ederken çıkabileceği maksimum hızda Şekil 3.25.c'deki gibi düşüş meydana gelecektir. Ayrıca φ değerinin artması sıkışık trafiklerde araç takiplerinde tepki gecikmesi gibi yan etkiler meydana getirmektedir. Bu gecikmenin başlıca sebebi ise φ değeri büyüdükçe, hız farkının dengelenmesi için gereken mesafe artmasıdır. Bu durumda aracın, sabit hızda kalması gereken bölge büyümüş olur. Geçiş adımının 1 olması arzu edilirken, φ_{min} değerinin katı kadar geçiş adımına maruz kalır. Bu ise takip eden bir araç için gecikme demektir.

Şekil 3.25.a'da $\varphi = 0$ grafiği aynı zamanda önceki çalışmadaki iki durumlu (hızlanma ve yavaşlama) algoritmayı da temsil etmektedir.

3.7.3 Birleşme Algoritmaları

Birleşme algoritmaları öncelikle temel karakteristiklerinin daha iyi anlaşılabilmeleri adına basit ve kontrollü bir ortamda incelenecek, ardından bu algoritmalar kalabalık veri setlerine tabi tutulacaklardır.

Şimdiye kadar değinilen birleşme algoritmalarının anlaşılabilir bir şekilde test edebilmek için Şekil 3.26'daki formasyonda üç farklı algoritma ile test edilmiştir. Bu formasyonda her bir araç 20 m/s hız ve kontrol bölgesi içinde olacak şekilde başlatılmıştır. Her bir araç sırasıyla birer metre fark ile konumlandırılmıştır. Bu şekilde üst şeritteki araçlar tampon tampona olacak şekilde denk gelmektedir.

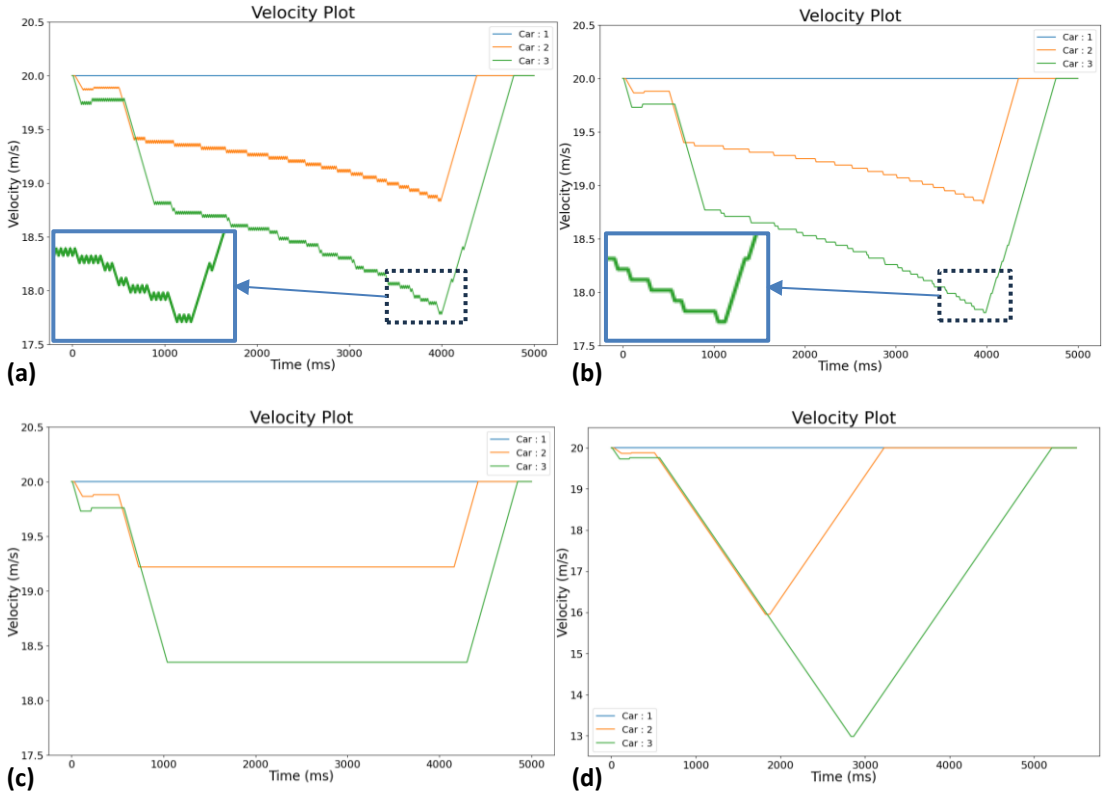
Şekil 3.26: Temel test formasyonu.

İlk test orijinal birleşme algoritması ile, ikinci test sabit hızla gidebilme kabiliyeti kazandırılarak iyileştirilmiş orijinal birleşme algoritması bu tez kapsamında geliştirilen varış tahminli birleşme algoritması ve son olarak test amacı ile oluşturulan test birleşme algoritması test edilmiştir.

Tablo 3.3: Algoritmalara göre ortalama hız

	Araç 1	Araç 2	Araç 3	Genel Ortalama
Algoritma 1: Orijinal Birleşme Algoritması	20.000	19.396	18.760	19.385
Algoritma 2: İyileştirilmiş Birleşme Algoritması	20.000	19.392	18.749	19.380
Algoritma 3: Varış Tahmini Birleşme Algoritması	20.000	19.414	18.717	19.377
Algoritma 4: Test Birleşme Algoritması	20.000	18.958	16.942	18.633

Her bir aracın ayrıca ve hepsinin genel olarak 5000 ms sonraki ortalama hızları Tablo 3.3'te ve hız grafikleri Şekil 3.27'de verilmiştir. Bu bilgilerden, orijinal algoritmanın iyileştirilmesi sonucu genel ortalama hızda sadece %0,026 oranında bir düşüş olduğu anlaşılmaktadır.



Şekil 3.27: Hız-zaman grafiği a) Orijinal birleşme algoritması b) İyileştirilmiş birleşme algoritması c) Varış tahmini birleşme algoritması d) Test birleşme algoritması

Yapılan deney sonuçlarında genel ortalamaya göre en iyi sonucun orijinal birleştirme algoritmasının verdiği ve ardından onun iyileştirilmiş algoritmanın takip ettiği görülmektedir. Ancak araçların bireysel hız ortalamalarını da göz önüne aldığımız zaman varış tahminli birleşme algoritmasında orta konumda bulunan araç 2'de en iyi sonucu verdiği dikkat çekmektedir.

Bütün test sonuçlarında araçların hepsi Şekil 3.28'deki gibi sorunsuz bir şekilde birleşme işlemini gerçekleştirebildikleri görülmüştür.

Şekil 3.28: Simülasyonda araçların birleşme anı.

Şekil 3.27 incelendiğinde iyileştirilmiş ve varış tahminli birleşme algoritmaları, insan davranışını temsilen oluşturulmuş test algoritmasından çok daha iyi bir performans ortaya koyduğu görülmektedir. Ancak bu tür merkezi kontrollü otonom sistemlerde araçlarda tepki gecikmesinin olmaması buradaki test algoritmasının gerçek hayat koşullarında bile insan davranışından daha iyi bir sonuç vermesi kaçınılmazdır. Bu sonuçlara dayanarak bu tür bir otonom sistemin hayalet trafik sıkışıklığına kesin bir çözüm olacağı öngörülmektedir.

3.7.4 Genel Test

Simülasyonun genel yapısı başlığında önceden bahsedilen sistem test formasyonunda orijinal birleşme algoritması, varış tahminli birleşme algoritması ve test algoritması, altı farklı veri setinde test edilmiştir. Simülasyonun her aşamasında takip algoritması aktif olarak çalışmakta olup, birleşme işlemleri sırasında birleşme algoritmaları da devreye girmektedir. Birleşme algoritmalarının olmadığı durumlarda ise yalnızca takip algoritmasının etkisi değerlendirilmektedir.

Veri setleri belirlenirken tali yoldan giren aracın en az 30 olmasına dikkat edilmiştir. Belirlenen tali yol araç sayısı, yaklaşık olarak 7 olan ana yol ve tali yol gecikme oranı ile çarpılmış ve ana yoldan girecek olan araç sayısı olarak belirlenmiştir. Bu sayılar sırasıyla %10, %20 %40, %80 ve %160 oranlarında arttırılarak yeni veri setleri oluşturulmuş ve bu veri setleri ile test yapılmıştır.

Tablo 3.4: Teste göre 5 dakika sürecinde giren araç sayısı

Test No	Oran	Ana Yol Araç Sayısı	Tali Yol Başına Araç Sayısı	Giren Toplam Araç Sayısı
1	%100	210	30	300
2	%110	252	33	330
3	%120	294	36	360
4	%140	269	42	420
5	%180	378	54	540
6	%260	546	78	780

Oluşturulan bu 5 dakikalık veri setleri kullanılarak iyileştirilmiş birleşme algoritması, varış tahminli birleşme algoritması ve test birleşme algoritması 15 dakikalık süreç boyunca test edilmiştir. Testlerden elde edilen sonuçlar ise Tablo 3.5, Tablo 3.6, Tablo 3.7, Tablo 3.8, Tablo 3.9 ve Tablo 3.10'da verilmiştir.

Tablo 3.5: Birinci testin sonuçları

İyileştirilmiş Birleşme Algoritması				
60.0 Araç/dk				
	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	20.00	20.00	20.00	20.00
GİRİŞ 2	17.96	19.08	19.41	19.40
GİRİŞ 3	-	17.97	19.08	19.09
GİRİŞ 4	-	-	17.97	18.03
Ortalama hız	19.572			
	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	61	43	55	51
GİRİŞ 2	12	11	3	4
GİRİŞ 3	0	5	12	13
GİRİŞ 4	0	0	19	11

Varış Tahminli Birleşme Algoritması				
60.0 Araç/dk				
	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	20.00	20.00	20.00	20.00
GİRİŞ 2	17.96	19.08	19.41	19.40
GİRİŞ 3	-	17.97	19.08	19.09
GİRİŞ 4	-	-	17.97	18.03
Ortalama hız	19.571			
	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	61	43	55	51
GİRİŞ 2	12	11	3	4
GİRİŞ 3	0	5	12	13
GİRİŞ 4	0	0	19	11

Test Birleşme Algoritması				
60.0 Araç/dk				
	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	19.99	19.99	19.98	19.99
GİRİŞ 2	17.95	19.07	19.41	19.38
GİRİŞ 3	-	17.97	19.07	19.09
GİRİŞ 4	-	-	17.97	18.03
Ortalama hız	19.550			
	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	61	43	55	51
GİRİŞ 2	12	11	3	4
GİRİŞ 3	0	5	12	13
GİRİŞ 4	0	0	19	11

Tablo 3.6: İkinci testin sonuçları

İyileştirilmiş Birleşme Algoritması	Varış Tahminli Birleşme Algoritması	Test Birleşme Algoritması
66.0 Araç/dk	66.0 Araç/dk	66.0 Araç/dk

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	19.99	19.99	19.99	20.00
GİRİŞ 2	17.96	19.07	19.41	19.41
GİRİŞ 3	-	17.95	19.08	19.09
GİRİŞ 4	-	-	17.97	18.03

Ortalama hız	19.566

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	59	58	67	47
GİRİŞ 2	6	7	9	11
GİRİŞ 3	0	13	13	7
GİRİŞ 4	0	0	16	17

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	19.99	19.99	19.99	20.00
GİRİŞ 2	17.96	19.07	19.41	19.41
GİRİŞ 3	-	17.95	19.08	19.09
GİRİŞ 4	-	-	17.97	18.03

Ortalama hız	19.566

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	59	58	67	47
GİRİŞ 2	6	7	9	11
GİRİŞ 3	0	13	13	7
GİRİŞ 4	0	0	16	17

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	19.96	19.97	19.98	19.99
GİRİŞ 2	17.95	19.01	19.40	19.39
GİRİŞ 3	-	17.94	19.08	19.08
GİRİŞ 4	-	-	17.96	18.03

Ortalama hız	19.549

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	59	58	67	47
GİRİŞ 2	6	7	9	11
GİRİŞ 3	0	13	13	7
GİRİŞ 4	0	0	16	17

Tablo 3.7: Üçüncü testin sonuçları

İyileştirilmiş Birleşme Algoritması	Varış Tahminli Birleşme Algoritması	Test Birleşme Algoritması
72.0 Araç/dk	72.0 Araç/dk	72.0 Araç/dk

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	19.99	19.99	19.99	19.99
GİRİŞ 2	17.94	19.09	19.40	19.41
GİRİŞ 3	-	17.96	19.08	19.10
GİRİŞ 4	-	-	17.96	18.02

Ortalama hız	19.561

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	66	68	53	65
GİRİŞ 2	8	8	11	9
GİRİŞ 3	0	14	12	10
GİRİŞ 4	0	0	14	22

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	19.99	19.99	19.99	19.99
GİRİŞ 2	17.93	19.09	19.40	19.41
GİRİŞ 3	-	17.96	19.08	19.10
GİRİŞ 4	-	-	17.96	18.02

Ortalama hız	19.560

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	66	68	53	65
GİRİŞ 2	8	8	11	9
GİRİŞ 3	0	14	12	10
GİRİŞ 4	0	0	14	22

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	19.97	19.97	19.98	19.98
GİRİŞ 2	17.89	19.09	19.40	19.40
GİRİŞ 3	-	17.96	19.07	19.10
GİRİŞ 4	-	-	17.96	18.01

Ortalama hız	19.546

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	66	68	53	65
GİRİŞ 2	8	8	11	9
GİRİŞ 3	0	14	12	10
GİRİŞ 4	0	0	14	22

Tablo 3.8: Dördüncü testin sonuçları

İyileştirilmiş Birleşme Algoritması
84.0 Araç/dk

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	19.99	19.99	19.99	19.99
GİRİŞ 2	17.96	19.08	19.40	19.41
GİRİŞ 3	-	17.96	19.08	19.09
GİRİŞ 4	-	-	17.96	18.03

Ortalama hız	19.559

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	70	79	63	82
GİRİŞ 2	13	11	12	6
GİRİŞ 3	0	11	21	10
GİRİŞ 4	0	0	22	20

Varış Tahminli Birleşme Algoritması
84.0 Araç/dk

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	19.98	19.99	19.99	19.99
GİRİŞ 2	17.95	19.08	19.40	19.41
GİRİŞ 3	-	17.96	19.08	19.09
GİRİŞ 4	-	-	17.96	18.02

Ortalama hız	19.558

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	70	79	63	82
GİRİŞ 2	13	11	12	6
GİRİŞ 3	0	11	21	10
GİRİŞ 4	0	0	22	20

Test Birleşme Algoritması
84.0 Araç/dk

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	19.95	19.98	19.98	19.98
GİRİŞ 2	17.92	19.06	19.39	19.41
GİRİŞ 3	-	17.94	19.07	19.09
GİRİŞ 4	-	-	17.95	18.02

Ortalama hız	19.550

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	70	79	63	82
GİRİŞ 2	13	11	12	6
GİRİŞ 3	0	11	21	10
GİRİŞ 4	0	0	22	20

Tablo 3.9: Beşinci testin sonuçları

İyileştirilmiş Birleşme Algoritması
108.0 Araç/dk

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	19.97	19.98	19.98	19.98
GİRİŞ 2	17.92	19.05	19.40	19.40
GİRİŞ 3	-	17.95	19.07	19.08
GİRİŞ 4	-	-	17.97	18.02

Ortalama hız	19.555

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	82	109	91	96
GİRİŞ 2	14	11	12	17
GİRİŞ 3	0	14	15	25
GİRİŞ 4	0	0	29	25

Varış Tahminli Birleşme Algoritması
108.0 Araç/dk

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	19.96	19.98	19.97	19.98
GİRİŞ 2	17.91	19.05	19.40	19.39
GİRİŞ 3	-	17.95	19.07	19.08
GİRİŞ 4	-	-	17.97	18.02

Ortalama hız	19.553

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	82	109	91	96
GİRİŞ 2	14	11	12	17
GİRİŞ 3	0	14	15	25
GİRİŞ 4	0	0	29	25

Test Birleşme Algoritması
0.0 Araç/dk

BAŞARISIZ

Tablo 3.10: Altıncı testin sonuçları

İyileştirilmiş Birleşme Algoritması	Varış Tahminli Birleşme Algoritması	Test Birleşme Algoritması
156.0 Araç/dk	156.0 Araç/dk	0.0 Araç/dk

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	19.72	19.86	19.90	19.89
GİRİŞ 2	17.86	19.00	19.38	19.38
GİRİŞ 3	-	17.95	19.07	19.08
GİRİŞ 4	-	-	17.97	18.02

Ortalama hız	19.460

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	125	154	119	148
GİRİŞ 2	18	21	17	22
GİRİŞ 3	0	21	33	24
GİRİŞ 4	0	0	33	45

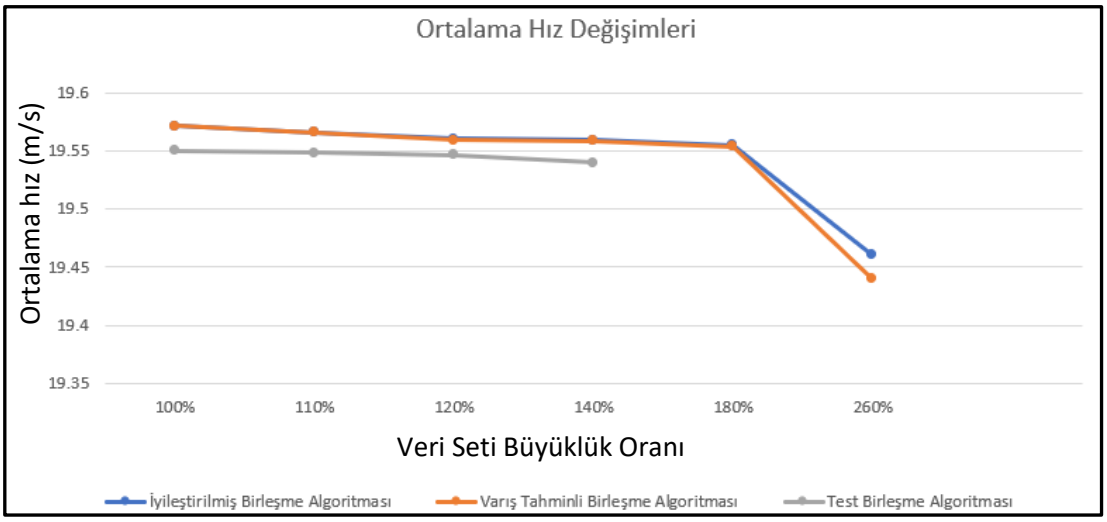
	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	19.66	19.83	19.88	19.88
GİRİŞ 2	17.88	19.00	19.36	19.38
GİRİŞ 3	-	17.95	19.07	19.08
GİRİŞ 4	-	-	17.96	18.02

Ortalama hız	19.440

	ÇIKIŞ 1	ÇIKIŞ 2	ÇIKIŞ 3	ÇIKIŞ 4
GİRİŞ 1	125	154	119	148
GİRİŞ 2	18	21	17	22
GİRİŞ 3	0	21	33	24
GİRİŞ 4	0	0	33	45

BAŞARISIZ

Tablo 3.5, Tablo 3.6, Tablo 3.7, Tablo 3.8, Tablo 3.9 ve Tablo 3.10'da elde edilen sonuçların ortalama hızlarındaki farklılıkların daha kolay bir biçimde incelenebilmesi adına Şekil 3.29'da her bir veri setine göre algoritmaların ortalama hız grafiği çizdirilmiştir.



Şekil 3.29: Veri setine göre ortalama hız grafiği.

Yapılan bu testler sonucunda en iyi ortalama hız değerlerini sağlayan algoritma iyileştirilmiş birleşme algoritması olmuştur. Bu algoritmayı, özellikle yoğun trafik senaryolarında etkinlik gösteren varış tahminli birleşme algoritması takip etmiştir. Ancak, test algoritmasının başarısız olduğu senaryolar, simülasyon sistemin

kapasitesinin artırılması ile giderilebilse bile diğer algoritmaların başarılı olabildiği beşinci ve altıncı testlerde başarısız olması bu tür bir trafik ortamında alternatif çözümlerin daha tercih edilebilir sonuçlar verdiğini ortaya koymaktadır.

Bu tür sınırlı giriş ve çıkış olan sistemlerde ana yoldan dahil olan araçlar 20 m/s hızla girdikleri için birleşme algoritmalarının sonucu olarak yavaşlayan araçlar ana yoldaki araçları da etkileyeceklerdir. Eğer sistemde araç girişi bu yavaşlamayı tolere edemeyecek duruma gelirse yavaşlayan araçlar ana yoldan giren araçlarla çarpışma meydana getirecektir. Bu çarpışma, algoritmaların başarısız olmasından değil simülasyonun doyum noktasını aştığı ve artık isabetli sonucun alınamayacağı anlamına gelmektedir. Beşinci ve altıncı testlerde gözlemlenen başarısızlık durumu simülasyonun doyum noktasını aştığından dolayı meydana gelmektedir. Yapılan temel test sonucunu gösteren Şekil 3.27'de diğer bütün algoritmaların yavaşladığı minimum hızın yaklaşık olarak 18 m/s olduğu bir ortamda test algoritmasının 13 m/s ye kadar yavaşlaması böyle bir sonucun gerçekleşebileceğini destekler niteliktedir. Varış tahminli birleşme algoritması ve iyileştirilmiş birleşme algoritmasının doyuma ulaşması veri set arttırma oranı %270-280 aralığında gerçekleşmektedir.

4. SONUÇ VE ÖNERİLER

Bu tez çalışması, şehir içi ulaşımda trafik yoğunluğunun ve sıkışıklıklarının azaltılması için yenilikçi bir model olarak MCS'nin etkili bir şekilde uygulanabilirliğini ortaya koymuştur. Çalışma kapsamında, MCS'nin verimlilik ve güvenlik açısından daha iyi performans göstermesi için takip ve birleşme algoritmaları önemli ölçülerde iyileştirilmiş, bu algoritmaların etkileri Python tabanlı simülasyon ortamında detaylı olarak incelenmiştir. Çalışmanın temel bulguları ve bunlara dayalı öneriler aşağıda sunulmuştur.

Bulgular:

- Python tabanlı simülasyon ortamı, geliştirilen algoritmaların performansını gözlemlemek için etkili bir altyapı sunmuş ve yeni algoritmaların mevcut yaklaşımlara göre performans kıyaslamasına kolaylık sağlamıştır.
- Yeni geliştirilen takip algoritması, MCS'deki aracın mevcut hızlanma ve yavaşlama kabiliyetlerine ek olarak sabit hızda gidebilme yetisi kazandırılarak öndeki aracı takip ederken daha stabil bir hız profili sergilemesi sağlanmış ve istenmeyen ardışık hızlanma ve yavaşlama durumu ortadan kaldırılmıştır.
- Mevcut birleşme algoritmasında da meydana gelen bu ardışık hareket takip algoritmasındaki yöntemden yararlanılarak iyileştirme yapılmış, stabil bir hareket profiline ulaşması sağlanmıştır.
- Mevcut ve iyileştirilmiş algoritmalara ek olarak alternatif ve yeni bir birleşme algoritması önerilmiş, elde edilen sonuçlar yeni önerilen algoritmanın bazı durumlarda performans üstünlüğü sağladığını göstermiştir.
- Bu tür merkezi kontrollü otonom sistemlerde, araçların tepki gecikmesinin olmaması, sistemin gerçek hayat koşullarında bile insan davranışına kıyasla üstün sonuçlar vermesini sağlamaktadır. Bu bağlamda, MCS'nin hayalet trafik sıkışıklıkları gibi karmaşık bir problemi ortadan kaldırabilecek etkili bir çözüm sunduğu sonucuna ulaşılmıştır.

Karşılaşılan Sorunlar ve Sınırlamalar:

- Sistem testlerinde, hız ve konum bilgilerinin anlık olarak tespit edilebildiği kabul edilmiş, haberleşme ve veri işleme kaynaklı gecikmeler dikkate alınmamıştır.
- Geniş çaplı ve yüksek araç sayılı simülasyonların getirdiği hesap yüklerinden dolayı sınırlı koşullarda test yapılabilmiştir. MCS'de her bir link için hesap merkezi ayrı olması gerekirken bu tür deneysel çalışmalarda yapılan denemeler tek merkezli olması hesap yükünü ve deney sürelerini arttırmaktadır.

Öneriler:

- Bu tezde elde edilen bulgular, MCS'nin potansiyelini ortaya koysa da algoritmaların gerçek dünya koşullarında da test edilmesi önemlidir. Bu nedenle, MCS sisteminin pilot projelerle desteklenerek gerçek trafik senaryolarında denenmesi önerilmektedir.
- Sistem performansını daha gerçekçi bir şekilde değerlendirebilmek için, ileride yapılacak testlerde haberleşme ve veri işleme gecikmelerinin de dikkate alınması önerilmektedir.
- Bu çalışmada tek şeritli olarak ele alınan MCS, geliştirilen ve iyileştirilen algoritmaların çok şeritli ortamlarda da test edilmesi ve şerit değiştirme algoritmalarının da bu kapsamda incelenmesi ve geliştirilmesi önerilmektedir.

Sonuç olarak, bu tez çalışması, şehir içi ulaşımın önemli problemlerinden biri olan trafik yoğunluğu ve hayalet trafik sıkışıklığı gibi olgulara karşı yenilikçi bir çözüm önerisi sunan MCS modelini detaylı bir şekilde ele almış ve bu sistemin şehir içi ulaşımda daha güvenli, verimli ve sürdürülebilir bir alternatif olarak uygulanabilirliğini ortaya koymuştur. Çalışma, mevcut sorunlara etkili çözümler üretmekle kalmayıp, aynı zamanda gelecekte yapılacak çalışmalara da güçlü bir temel sağlamaktadır.

5. KAYNAKLAR

AASHTO, "AASHTO Releases 7th Edition of its Highway & Street Design 'Green Book' [online]", *AASHTO Journal*, (18 December 2024), (https://aashtojournal.transportation.org/aashto-releases-7th-edition-of-its-highway-street-design-green-book/), (2018).

Alabyad, N., Hany, Z., Mostafa, A., Eldaby, R., Tagen, I. A., ve Mehanna, A., *"*From Vision to Precision: The Dynamic Transformation of Object Detection in Autonomous Systems*"*, *6th International Conference on Computing and Informatics, ICCI 2024*, 332-344, doi: 10.1109/ICCI61671.2024.10485026. (2024).

Armstrong, J., "Advanced Statistics with Acusera 24.7 - Randox Laboratories [online]", (7 November 2024), (https://www.randox.com/advanced-statistics-with-acusera-24-7/), (2023).

Ban, X. (Jeff), Yang, D., Wang, J., ve Hamdar, S., "Editorial: Connected and automated vehicles (CAV) based traffic-vehicle control", *Transportation Research Part C: Emerging Technologies*, 112, 116-119, doi: 10.1016/J.TRC.2020.01.011, (2020).

Barney, M., "Six Sigma.", *The Industrial-Organizational Psychologist*, 39(4), 104-107, (2002).

Bimbraw, K., "Autonomous Cars: Past, Present and Future - A Review of the Developments in the Last Century, the Present Scenario and the Expected Future of Autonomous Vehicle Technology", *ICINCO 2015 - 12th International Conference on Informatics in Control, Automation and Robotics*, 191-198, doi: 10.5220/0005540501910198, (2015).

Bozuyla, M. "Çift modlu ulaşım sistemleri için yeni bir yaklaşım olarak metrocar", Doktora Tezi, *Pamukkale Üniversitesi Fen Bilimleri Enstitüsü*, Elektrik Elektronik Mühendisliği Anabilim Dalı, Denizli, (2019).

Bozuyla, M. ve Tola, A. T., "MetroCar—a novel transportation system for cities.", *ELECTRONICS WORLD*, 124, 32-33, (2018).

Bozuyla, M., Tola, A. T., ve Murat, Y. Ş., "A Novel Safe Merging Algorithm for Connected Vehicles Using NetLogo*"*, *Elektronika ir Elektrotechnika*, 24(3), doi: 10.5755/j01.eie.24.3.20956, (2018).

Bozuyla, M., ve A. T. Tola., "Designing a Novel Transportation System Using Microscopic Models and Multi-Agent Approach", *Automatic Control and Computer Sciences*, 55(2), 125-136, doi: 10.3103/S0146411621020036, (2021).

Bozuyla, M. ve Tola, A. T., *"*Comparison of Metrocar System with Dual-Mode Transit Systems*"*, *Automatic Control and Computer Sciences*, 56(6), 546-552, doi: 10.3103/S0146411622060037, (2022).

Burgan, M., *The Pontiac Firebird. Capstone*, United States, *Capstone Press*, (1999).

Chen, T., Chen, F., ve Chen, C., "Review on Driverless Traffic from Management Perspective", *MATEC Web of Conferences*, 124, doi: 10.1051/MATECCONF/201712403002, (2017).

Cranswick, M., *Pontiac Firebird: The Auto-Biography*, United Kingdom, Veloce Publishing Ltd., (2013).

Cruise, "Uber and Cruise to deploy autonomous vehicles on the Uber platform | Cruise [online]", (2 September 2024), (https://www.getcruise.com/news/blog/2024/uber-and-cruise-to-deploy-autonomous-vehicles-on-the-uber-platform/), (2024).

Di Febbraro, A., Gallo, F., Giglio, D. ve Sacco, N., *"*Traffic management system for smart road networks reserved for self-driving cars*"*, *IET Intelligent Transport Systems*, 14(9), 1013-1024, doi: 10.1049/IET-ITS.2019.0675, (2020).

El Hamdani, S., ve Benamar, N., *"*Autonomous traffic management: Open issues and new directions*"*, *2018 International Conference on Selected Topics in Mobile and Wireless Networking, MoWNeT*, 1-5, doi: 10.1109/MOWNET.2018.8428937, (2018).

Elsagheer Mohamed, S. A., ve Alshalfan, K. A., "Intelligent Traffic Management System Based on the Internet of Vehicles (IoV)", *Journal of Advanced Transportation*, 2021(4), 1-23, doi: 10.1155/2021/4037533, (2021).

Flyte, M. G., "Safe design of in-vehicle information and support systems: the human factors issues", *International Journal of Vehicle Design*, 16, 158-169, (1995).

Ganti, A., "Central Limit Theorem (CLT): Definition and Key Characteristics [online]", (27 December 2024), (https://www.investopedia.com/terms/c/central_limit_theorem.asp), (2024).

Gora, P., "Simulation-Based Traffic Management System for Connected and Autonomous Vehicles", *Road Vehicle Automation 4. Lecture Notes in Mobility*, *Springer*, 257-266, doi: 10.1007/978-3-319-60934-8_21, (2018).

Gökaşar, I., Timuroğullari, A., Deveci, M., ve Garg, H., "SWSCAV: Real-time traffic management using connected autonomous vehicles", *ISA Transactions*, 132, 24-38, doi: 10.1016/J.ISATRA.2022.06.025, (2023).

Grembek, O., Kurzhanskiy, A., Medury, A., Varaiya, P., ve Yu, M., "Making intersections safer with I2V communication", *Transportation Research Part C: Emerging Technologies*, 102, 396-410, doi: 10.1016/J.TRC.2019.02.017, (2019).

Guanetti, J., Kim, Y., ve Borrelli, F., *"*Control of connected and automated vehicles: State of the art and future challenges*"*, *Annual Reviews in Control*, 45, 18-40, doi: 10.1016/J.ARCONTROL.2018.04.011, (2018).

Hunter, J. D., "Matplotlib: A 2D graphics environment", *Computing in Science & Engineering* 9(3), 90-95, doi: 10.1109/MCSE.2007.55, (2007).

Ishikawa, S., ve Arai, S., *"*Evaluating advantage of sharing information among vehicles toward avoiding phantom traffic jam*"*, *Proceedings - Winter Simulation Conference 2016-February*, 300-311, doi: 10.1109/WSC.2015.7408173, (2016).

Jiang, P., Dong, Z., Wang, Y., Su, Y. and Li, Z., *"*Empirical Study of Phantom Traffic Jams on Urban Expressway*"*, *Ninth International Conference on Intelligent Control and Information Processing, ICICIP*, Wanzhou, China, 2018, 99-102, doi: 10.1109/ICICIP.2018.8606713, (2018).

Krithikadatta, J., *"*Normal Distribution*"*. *Journal of Conservative Dentistry*, 17(1), 96-97, doi: 10.4103/0972-0707.124171, (2014).

Molnár, T. ve Orosz, G., "Destroying Phantom Jams with Connectivity and Automation: Nonlinear Dynamics and Control of Mixed Traffic.*"*, *Transportation Science*, 58(6), doi: 10.1287/TRSC.2023.0498, (2024).

Pressnell, J., *Citroen DS: The complete story*, United Kingdom, Crowood Press (1999).

Rodrigues, E., "The Scikit-HEP Project.", *EPJ Web of Conferences*, 214, doi: 10.1051/epjconf/201921406005, (2019).

Pestorius, M. S., *"*Apply six sigma to sales and marketing.*"*, *Quality Engineering*, 52, 629-630, (2007).

PSU, "4.1.3 - Impact of Sample Size | STAT 200 [online]", (27 December 2024), (https://online.stat.psu.edu/stat200/lesson/4/4.1/4.1.2), (2024).

SAE, "Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems", doi: 10.4271/J3016_201401, (2014).

SAE, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles", doi: 10.4271/J3016_202104, (2021[a]).

SAE, "SAE Levels of Driving Automation™ Refined for Clarity and International Audience [online]", (10 November 2024), (https://www.sae.org/blog/sae-j3016-update), (2021[b]).

Temple, D. W., ve Adler D., *GM's Motorama : the glamorous show cars of a cultural phenomenon*,United States, (2006).

Toma, S. G., "What Is Six Sigma?" *Manager Journal*, 8, (2008).

Xie, M., L., Trassoudaine, J. Alizon, M. Thonnat, ve J. Gallice., "Active and intelligent sensing of road obstacles: Application to the European Eureka-PROMETHEUS project", *1993 IEEE 4th International Conference on Computer Vision*, 616-623, doi: 10.1109/ICCV.1993.378154, (1993).

Zhu, S., Inhi, K., Qi, Z., Guohua, S., Lei, Y., ve Zhigui, C., "Analysis of Vehicle Stop-and-Go Driving Behaviors at Signalized Intersections", *CICTP 2017: Transportation Reform and Change—Equity, Inclusiveness, Sharing, and Innovation*, 3150-3160, doi:10.1061/9780784480915.330, (2018).