T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM 498 BİTİRME ÇALIŞMASI

MAKİNE ÖĞRENMESİ KULLANARAK YAPAY ZEKAYA OTONOM ARAÇ PARK ETTİRME

Omar GARİBOV

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ
Tez Danışmanı : Dr. Sümeyye KAYNAK

2022-2023 Yaz Dönemi

T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

MAKİNE ÖĞRENMESİ KULLANARAK YAPAY ZEKAYA OTONOM ARAÇ PARK ETTİRME

BSM 498 - BİTİRME ÇALIŞMASI

Omar GARİBOV

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Bu tez .. / .. / … tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

…….	…….	…….
Jüri Başkanı	**Üye**	**Üye**

ÖNSÖZ

Yapay sinir ağlarını kullanarak simülasyon ortamında park yeri aramak ve park etmek karmaşık senaryolarda mümkün müdür? Hala gelişmeye devam eden ve yeni nesil araçlarda sık sık gördüğümüz otonom sürüş bir standart haline gelmeye başlıyor. İnsanlar bu özelliğin birçok avantajı olduğunu düşünüyor, ancak genellikle karmaşık durumlarda park etme işlemi zorluklar çıkarabiliyor. Benim gibi pek çok insan trafikte araba kullanmada iyidir ama söz konusu aracı park etmek olduğunda gerektiğinden fazla efor sarf ederiz. Günümüzde hala insanların çoğunluğu araçlarını park etmekte zorlanır. Peki ya optimizasyon algoritmaları ve makine öğrenmesi kullanılarak bu karmaşık işlem otomatikleştirilebilirse?

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

PPO	: Proksimal Tedbir Optimizasyonu (Proximal Policy Optimization)
RL	: Takviyeli Öğrenme (Reinforcement Learning)
IRL	: Ters Takviyeli Öğrenme (Inverse Reinforcement Learning)
ML	: Makine Öğrenmesi (Machine Learning)
AI	: Yapay Zeka (Artificial Intelligence)
IL	: Taklit Öğrenme (Imitation Learning)
GAIL	: Generative Adversarial Imitation Learning

ŞEKİLLER LİSTESİ

ÖZET

Anahtar kelimeler: Otonom Park, Derin Takviyeli Öğrenme, Gerçekliğe Yakın Simülasyon Ortamı, Unity, MLagents

Unity ile fizik kurallarına uygun olarak oluşturulan, gerçekliğe yakın simülasyon ortamında makine öğrenmesi ve optimizasyon algoritmaları kullanılarak otonom park işlemini gerçekleştirecek yapay zeka modeli eğitilmiştir. Araç modeli, üzerinde bulunan uzaklık sensörleri ve kameralar sayesinde araç park yeri bulma, park manevrası gerçekleştirme ve karşıdan gelen araç denetimini sağlamaktadır. Unity ML-Agents eklentisi model eğitimi için kullanılır. En optimal ödüllendirme sistemi deneme yanılma yöntemiyle oluşturulmuştur. Bu sayede yapay zekanın her turda sadece ödülü almak için istenmeyen kısayollara başvurması engellenmiştir. Bu projede kullanılan model eğitim mantığı aynı konfigürasyonlar ve sensörler kullanılarak gerçek araçlar için de uygulanabilir.

Bu çalışmada derin takviyeli öğrenme (Deep Reinforcement Learning), PPO (Proximal Policy Optimization) ve taklit öğrenim (Imitation Learning) yöntemleri kullanılmış olup oluşturulan model deploy(kullanıma hazırdır) edilmiştir.

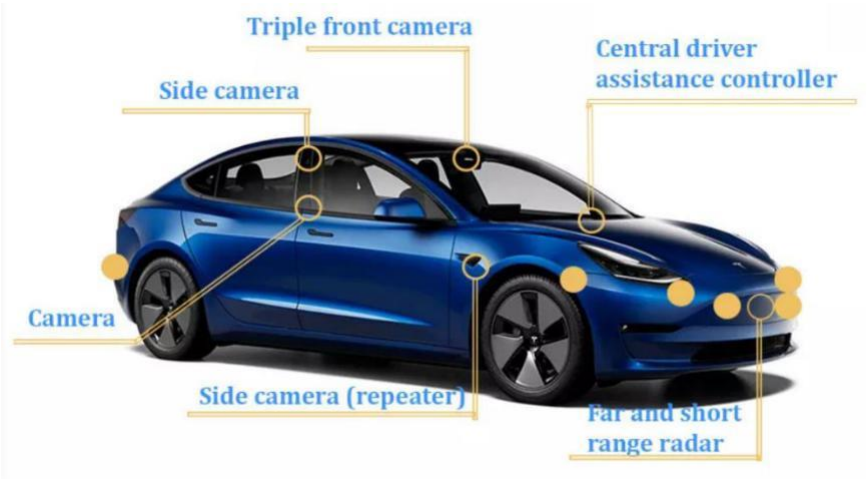
BÖLÜM 1. GİRİŞ

Bu proje kapsamında unity ile oluşturulan simülasyon ortamında otomatik park işlemini gerçekleştirecek yapay zeka modelini eğitilmiştir. Eğitilen model gerçekçi temeller üzerinden eğitildiği için (örneğin park sensörleri ve kameralar) gerçek hayatta da uygulanabilir.

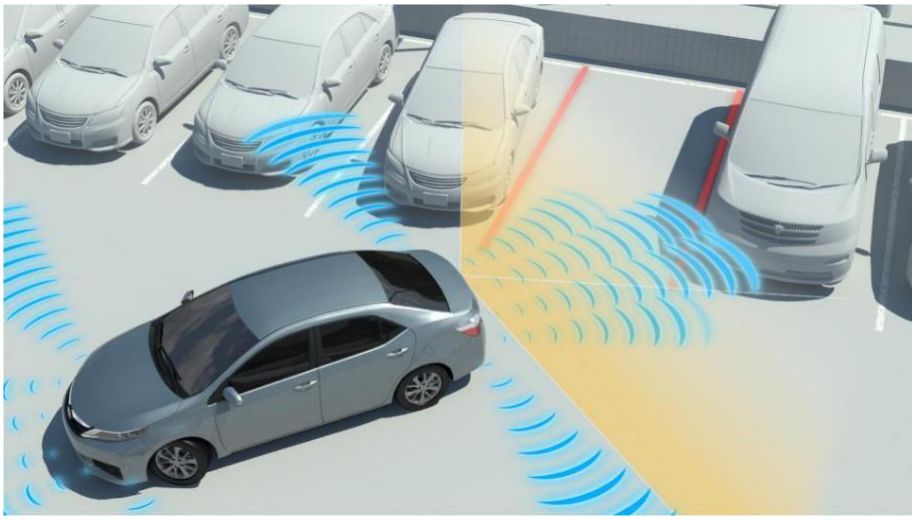
1.1. Geliştirilen Yazılımın Amacı Hedeflenen Sonuç

Geliştirilen yazılımın ana amacı dışarıdan müdahale olmadan park yeri bulma, park etme ve dış etken kontrollerinin sağlanarak güvenli şekilde bu işlemleri yerine getirecek olan yapay zeka modelinin makine öğrenmesi yöntemleri kullanılarak eğitilmesi ve modelin sonradan kullanılabilecek şekilde kaydedilmesidir. Eğitim için oluşturulan simülasyon ortamı gerçek dünyaya benzetilmelidir.

Gerçek dünyada arabalarda bulunan uzaklık sensörü ve kamera mantığı simülasyon ortamına uyarlanmalıdır. Otonom sürüş ve park etme işlemleri bu kamera ve sensörler yardımıyla yaptırılmalıdır.



Şekil 1.1. Modern Arabalarda Kamera Kullanımı



Şekil 1.2. Modern Arabalarda Sensör Yardımıyla Park Etme

Bu tarz yaklaşımın amacı simülasyon ortamında geliştireceğimiz çözümün gerçek hayatta da uygulanabilir olmasını garantilemektir.

1.2. Gerçek Dünyada Potansiyel Kullanım Alanları

Yeni nesil akıllı araba pazarı dünya ve Türkiye ekonomisinde büyümeye devam ediyor. Gerekli iyileştirme ve optimizasyon süreçlerinden sonra araçlar için otonom park yazılımı olarak uygulanabilir.

Bunun yanı sıra sanal gerçeklik yardımıyla veya üç boyutlu öğretici olarak tasarlanan araba park etme oyunlarında kullanıcıya çözüm yolunda yardımcı olacak şekilde oyun yazılımına entegre edilebilir.

1.3. Projede Kullanılan Teknolojiler

Conda – Açık kaynaklı, platformlar arası, dilden bağımsız bir paket yöneticisi ve çevre yönetim sistemidir. Başlangıçta Python veri bilimcilerinin karşılaştığı zorlu paket yönetimi zorluklarını çözmek için geliştirildi ve bugün Python ve R için popüler bir paket yöneticisidir [1]. Projede ML-Agents eklentisinin çalıştırılabilmesi için sanal ortam oluşturma ve kullanılacak Python kurulumlarının yapılmasında gereklidir.

Pythorch – Torch kütüphanesine dayanan açık kaynaklı bir makine öğrenme kütüphanesidir, bilgisayarla görme ve doğal dil işleme gibi uygulamalar için kullanılır [2]. Yapay zeka modelinin makine öğrenmesi kullanılarak eğitilme işlemi için gereklidir.

Python 3.8 – Web uygulamaları, yazılım geliştirme, veri bilimi ve makine öğreniminde (ML) yaygın olarak kullanılan bir programlama dilidir [3].

Unity – Öncelikli olarak bilgisayarlar, konsollar ve mobil cihazlar için video oyunları ve simülasyonları geliştirmek için kullanılan ve Unity Technologies tarafından geliştirilen çapraz platform bir oyun motorudur [4]. Eğitimin gerçekleştirileceği simülasyon ortamının oluşturulması için kullanılır.

Unity Hub – Editör sürümleri arası geçiş yapabilmeye, kolayca projelerin özelliklerini kontrol etmeye ve eklentiler eklemeye olanak sağlar.

ML-Agents – Unity Machine Learning Agents Toolkit (ML-Agents), oyunların ve simülasyonların derin pekiştirmeli öğrenme ve taklit öğrenme kullanarak akıllı modelleri eğitmek için ortamlar olarak hizmet etmesini sağlayan açık kaynaklı bir projedir [5].

Microsoft Visual Studio Code – Microsoft tarafından Windows, Linux ve MacOS için geliştirilen bir kaynak kodu düzenleyicisidir [6]. Proje dahilinde kullanılan kontrol ve konfigürasyon dosyaları bu editör kullanılarak düzenlenmiştir.

BÖLÜM 2. MAKİNE ÖĞRENMESİ VE ALGORİTMALAR

Makine öğrenmesi temelde veri ile beslenen ve bu verileri işleyerek öğrenen, öğrenirken performansını iyileştirmeyi hedefleyen yazılımlardır. Sıradan bilgisayar yazılımlarından onu ayıran en önemli özellik iki cisim arasındaki farkın yazılımcı tarafından elle tanımlanması yerine yapay zeka modelini bu cisimlerle alakalı büyük boyutta veriler kullanarak eğitip aradaki ayrımı güvenli şekilde yapabilmesini sağlamaktır. Eğitim işlemi sırasında farklı algoritmalar kullanılabilmektedir.

Makine öğrenmesi yapay zekanın bir koludur ve kendi içinde öğrenme algoritmaları açısından türlere ayrılır. Bu proje dahilinde takviyeli öğrenme algoritması kullanılmıştır.

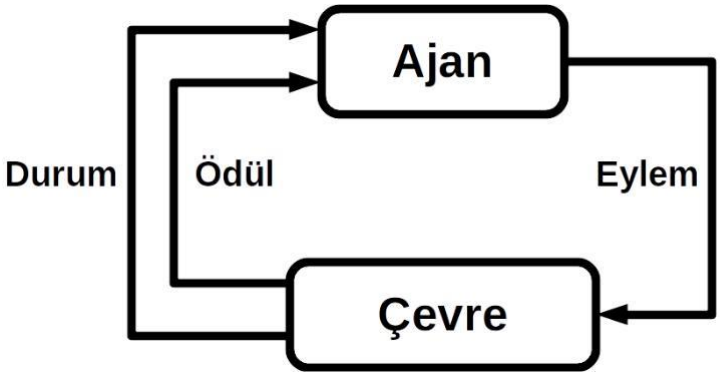
2.1. Takviyeli Öğrenme (Reinforcement Learning)

Bu makine öğrenmesi yaklaşımı temelde en yüksek ödül miktarına ulaşmayı hedefleyen öznelerin hangi eylemi yapmasıyla ilgilenir. Takviyeli öğrenmede yazılımcı, istenen davranışları ödüllendirmek ve olumsuz davranışları cezalandırmak için bir yöntem geliştirir. Bu yöntem, ajanı teşvik etmek için istenen eylemlere pozitif değerler ve istenmeyen davranışlara negatif değerler atar [6]. Optimum bir çözüme ulaşmak için uzun vadeli ve maksimum ödül aramaya programlanır. Bu uzun vadeli hedefler, temsilcinin daha küçük hedefler üzerinde oyalanmasını önlemeye yardımcı olur. Temsilci zamanla olumsuzdan kaçınmayı ve olumluyu aramayı öğrenir. Bu öğrenme yöntemi, denetimsiz makine öğrenimini ödüller ve cezalar yoluyla yönlendirmenin bir yolu olarak yapay zekada (AI) benimsenmiştir.

2.1.1. Ödül

Pekiştrimeli öğrenmeyi diğer makina ve derin öğrenme algoritmalarından ayıran en önemli özelliği ise verilerin etiket barındırmamasıdır. Ajan (agent) ve ortam

(environment) olacak şekilde iki ana unsur bulunmaktadır. İlk aşamada ajan ortami inceler ve işlem yapar. Yapılan işleme cevap olarak ortam ajanı ödüllendirir. Ajanın hedefi süreç boyunca alabileceği ödül miktarını maksimize etmektir.



Şekil 2.1. Takviyeli Öğrenme

2.1.2. Takviyeli Öğrenmenin Zorlukları

En iyi yaklaşıma ulaşmak için, alınan ödülleri temel alan birden fazla RL algoritması ve tekniği mevcuttur. Genel olarak bu teknikler yüksek performanslı olsa da özellikle ödüllerin sınırlı olduğu durumlarda (örneğin yalnızca oyun kazanıldığında veya kaybedildiğinde ödül elde edildiği durum) verimliliği büyük ölçüde azalır. Bu sorunu çözmek için, ödül sayısını artıracak ve ajana daha fazla ödül kazanma imkanı tanıyacak fonksiyonları el ile girmek gereklidir. Özellikle bu projedeki otonom araç park etme işlemi için hiçbir doğrudan ödül işlevi bulunmadığından manuel yaklaşım zorunludur. Bu yöntemi uygulamanın getirdiği zorluklardan biri de yapay zeka modelinin ödül sisteminde açık bulup kısayoldan hızlıca puan kazanmaya yatkın olmasıdır. Taklit Öğrenme tam da bu noktada işimize yarıyor.

2.1.3. Tersine Takviyeli Öğrenme (Inverse Reinforcement Learning)

Normal bir RL ortamında amaç, önceden tanımlanmış olan ödül metotları kullanılarak en verimli yaklaşımı üreten karar sürecini öğrenmektir. 2000'de Andrew Ng ve Stuart Russel tarafından geliştirilen "Tersine Takviyeli Öğrenme" işlemi tersine çevirerek ödül fonksiyonunu gözlemlenen davranıştan çıkarmaya çalışır [9].

Örneğin otonom sürüş işlemini ele alalım. RL ile öğrenim yaptırılırsa tüm olayları (Kırmızı ışıkta dur, Yayalardan kaçın, Kaldırımdan uzak dur vb.) kapsayacak bir ödül fonksiyonu yazmamız gerekecektir, bunun yanı sıra her davranışın önem düzeyini belirten bir ağırlık listesi (iki kısıtlandırılmış durum aynı anda oluştuğunda hangisine öncelik verileceğini belirten liste) de oluşturmamız gerekli olacaktır.

IRL yöntemi ise eğitim için bir dizi sürüş verisi alır ve insan davranışlarından yola çıkarak bu işlemleri gerçekleştirmek için gereken ödül fonksiyonunu tahmin etmeye çalışır**.** Ng ve Russel'in da belirttiği gibi, "Ödül fonksiyonu bir problemin en özlü sağlam ve aktarılabilir çözümüdür". Doğru ödül fonksiyonunu elde ettikten sonra sorunun çözümüne ulaşmak için RL yöntemi kullanılabilir.

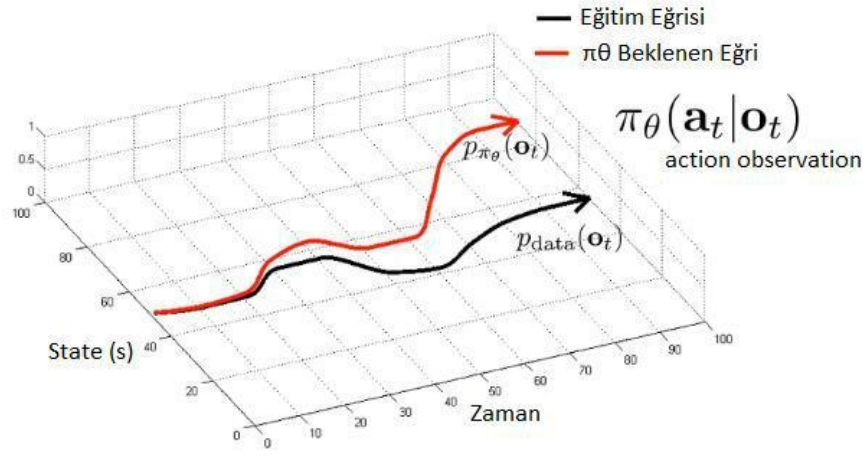
2.2. GAIL (Generative Adversarial Imitation Learning)

Taklit yoluyla öğrenim hiç de yeni bir terim değildir. Bu fikrin günümüzde tekrardan popülerlik kazanmasının ardındaki en büyük sebep bilgisayar teknolojilerindeki gelişmeler ve artan yapay zeka temelli yazılımlardır. IL'de ana mantık yaptırılacak işin bir uzman tarafından yaptırılıp örnek olarak sunulmasıdır. Devamında ajan ona sunulan örnekteki karar yapılarını inceleyip taklit ederek en optimum çözüme ulaşmaya çalışır. Genel olarak taklit öğrenme, bir uzmanın aynı davranışı üretecek bir ödül işlevi belirlemek veya doğrudan politikayı öğrenmek yerine istenen davranışı göstermesinin daha kolay olduğu durumlarda yararlıdır [7]. GAIL bir IRL algoritmasıdır. Modelden bağımsız bir algoritma yapısına sahiptir. Büyük, yüksek boyutlu ortamlarda karmaşık işlemleri taklit etmek için tercih edilir. Performans açısından diğer modelden bağımsız algoritmalara nazaran olukça etkileyicidir.

GAIL, IRL ve GAN konseptlerinin bir birleşimi gibidir. Bu da ona küçük miktarda örnek veriden öğrenebilme imkanı sağlar. Algoritmanın ana amacı verilen örnek verileri kullanarak benzer davranışları barındıran üreticiler yetiştirmektir.

2.2.1. Davranışsal Klonlama (Behavioral Cloning)

"Observation" ve "action" çiftinden oluşacak şekilde istenilen davranışı gösteren örnek niteliğinde bir veri kümesi oluşturulur. Devaminda RL kullanılarak model eğitiliyor. Elde edilen yaklaşım πθ ile gösterilir. Örneğin yapacağımız projede araç hangi durumdayken (state, observation) sürücü ne yapıyor (action) verilerine göre bu işlem uyarlanabilir.



Şekil 2.2. Davranışsal Klonlama

Asıl amaç, yukarıda gördüğünüz bu kırmızı ve siyah okların ya da matematiksel deyişle Pπ(obs) ile Pdata(obs)'nin yakınsamasıdır.

- **Avantajlar:** basitlik ve verimlilik. Uygun uygulamalar, uzun vadeli planlamaya ihtiyaç duymadığımız, uzmanın durum uzayını kapsadığı ve hata yapmanın ölümcül sonuçlara yol açmadığı uygulamalar olabilir [8].
- **Dezavantajlar:** Uzun vadeli planlamaya uygun değil, eğitim ve test aşamaları arasında durum dağılımında uyumsuzluk meydana gelebilir.

2.3. Proksimal Politika Optimizasyonu (Proximal Policy Optimization)

2017 yılında OpenAI tarafından tanıtılan algoritma performans ve kavrama arasındaki dengeyi koruması açısından Takviyeli öğrenmede son teknoloji olarak kabul edilmektedir. Aynı zamanda geniş bir kullanıcı kitlesi tarafından "pratikliği açısından benimsenmesi kolay olarak" nitelendiriliyor. Eğitim sürecinde büyük dalgalanmalar sorunuyla karşılaşıldığında, PPO bununla kolayca başa çıkabilir. PPO yakın uç strateji optimizasyonu fikri, eğitimin her adımının strateji güncellemesinin boyutunu sınırlandırarak eğitim aracısı davranışının istikrarını iyileştirmektir. Yukarıdaki fikirleri gerçekleştirmek için, PPO, politika güncellemesini küçük bir aralıkta sınırlamak için kırpma yoluyla yeni bir amaç işlevi olan "Kırpılmış vekil amaç işlevi" tanıttı.

2.3.1. Kırpılmış Vekil Amaç İşlevi

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, \text{ so } r(\theta_{\text{old}}) = 1.$$

Şekil 2.3. Önem Oranı

Yukarıda denklemde, rt() eski ve yeni stratejiler arasındaki olasılık oranını temsil eder:

Rt() ise > 1. Şu anki strateji kapsamındaki eylemin gerçekleşme olasılığı, orijinal stratejiden daha yüksektir.

Rt() (0,1) ise, şu anki stratejiye göre eylemin gerçekleşme olasılığı orijinal stratejiden daha düşüktür.

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$$

Şekil 2.4. Yeni Amaç İşlevi

Şu anki stratejinin eylem olasılığı önceki stratejiden çok daha yüksekse ve hedeflenen işlev bu adımda kısıtlandırılmamışsa, rt (value) değeri yüksek olacaktır. Bunun

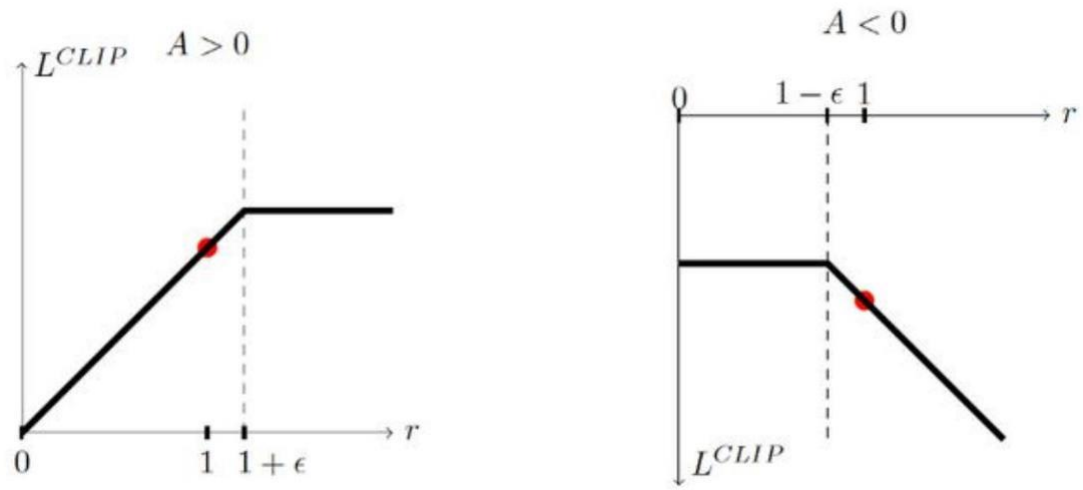
karşısını almak için, nesnel işlevi sınırlandırmak ve rt () değerinin 1'den uzak olmasına neden olabilecek değişiklikleri cezalandırmak ve bu sayede önemli yaklaşım değişikliklerinin gerçekleşmemesini sağlamak gerekir.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(\underbrace{r_t(\theta)\hat{A}_t}_{\text{L CPI}}, \underline{\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t})\right]$$

Modifies the surrogate objective by clipping the prob ratio.
--> Which removes the incentive for moving rt outside of the interval [1 - e, 1 + e]

Şekil 2.5. Özelleştirilmiş Amaç İşlevi

Bu fonksiyon bize iki farklı olasılık oranı sonucunu verir. Sonraki adımda minimum kırpma ve kırpmama değerini gireriz. Bunu yaparken iki durumu dikkate almak gereklidir.

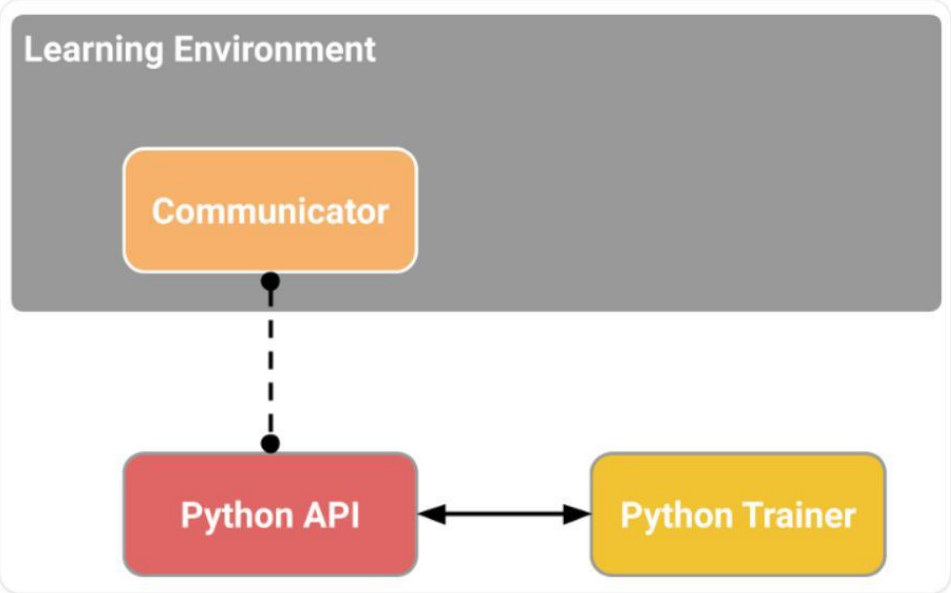


Şekil 2.6. Kırpma işlemi

PPO'nun bu varyantında, vekil avantajı kırpılır. Güncelleştirilen ilke özgün ilkeden fazla saparsa, örnek 0 gradyanını verir. Bu mekanizma, politikanın aşırı büyük güncellemelerini önler ve güvenilir bir bölgede tutar.

2.4. Unity ML-Agents Eklentisi Çalışma Mantığı

Dört temel komponent bulunmaktadır:



Şekil 2.7. MlAgents

1. Unity sahnesini ve çevre unsurlarını içeren Öğrenme Ortamıdır.
2. Ortamla etkileşim kurmak ve değişiklikler yapmak için Python arabirimini içeren API. Aynı zamanda bu API'ı eğitimi başlatmak için kullanırız.
3. Öğrenme Ortamını Python API'sine bağlayan Harici İletişimci.
4. Python trainer'leri: PyTorch ile yapılan RL algoritmaları (PPO, SAC…).

BÖLÜM 3. PROBLEMİN SİMÜLE EDİLMESİ

Araç fizikleri ve gerçekleştirilecek olan işlemler gerçek hayata uyarlanabilecek şekilde oluşturulmuştur. Buna arabanın hızlanması, üzerindeki sensörler, tork değeri, maksimum hız değeri ve diğer özellikler de dahildir.

3.1. Simülasyon Ortamının Avantaj ve Dezavantajları

Simülasyon bize zamanı hızlandırma, işleri çok sayıda arabayla paralel olarak yapma, pahalı kazalardan kaçınma ve gerçek hayatta tehlikeli durumlar yaratmak zorunda kalmama avantajı sağlıyor. Ayrıca gerçek bir araba satın alıp modifiye etmekten çok daha ucuz. Dezavantaj olarak ise gerçek hayatta karşılaşılabilecek durumların birebir oluşturulmasının asla mümkün olmamasını söyleyebiliriz. Örneğin yağmurlu havada sensörlerin ne kadar verimli çalışacağını simülasyon ortamında tam anlamıyla test etmemiz mümkün olmayacaktır.

3.1.1. Unity

Unity eğitim ortamını oluşturmamıza, bu ortamın fizik kurallarına müdahale etmemize, modeller ve haritalar oluşturarak detaylı bir dünya yaratmamıza olanak sağlayan ücretsiz bir oyun motorudur. Farklı oyun motorları kullanmak yerine Unity kullanmamın arkasındaki en önemli sebep yapay zeka modeli eğitimini unity'nin eklentisi olan ML-Agents kullanarak gerçekleştirmiş olmamdır.

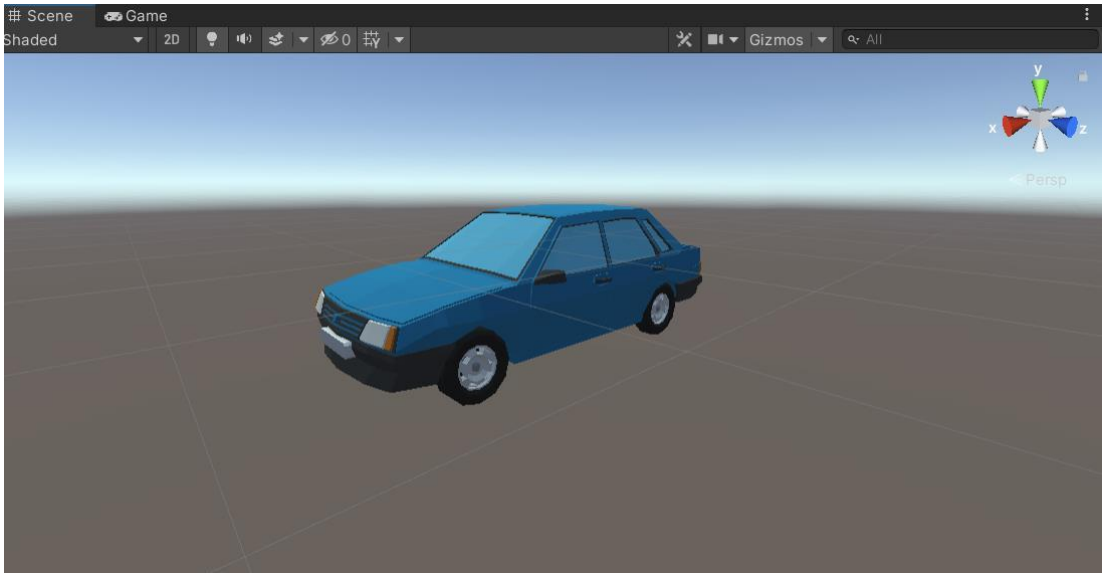
3.2. Ortamın Oluşturulması

Hedeflediğimiz işlemler otonom park yeri bulma ve dış etkenlere dikkat ederek park etme işleminin yaptırılması olduğundan oluşturmamız gereken ortamda bir adet park yeri ve park etme işlemi sırasında dikkat edilmesi gereken bağımsız bir olay

bulunmalıdır. Ben bağımsız olay olarak belirli aralıklarla ortaya çıkan ve park alanı dışına ulaştığında yok edilecek olan ikinci bir araç kullandım. Bu araç yapay zeka kontrolünde olan aracımıza park işlemi sırasında engel oluşturacak şekilde yolun karşı tarafından gelmektedir. Yapay zeka park etme işlemi zamanı karşıdan gelen aracı uzaklık sensörleri yardımıyla algılamalı ve geçmesine izin vermelidir.

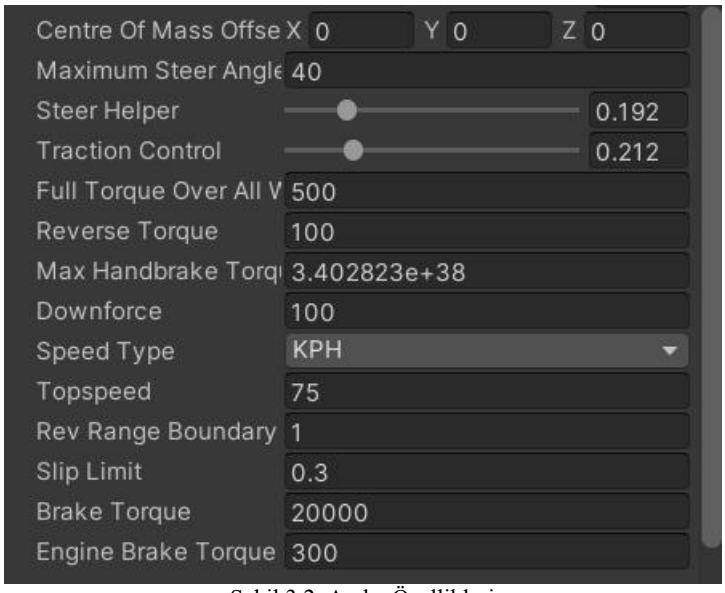
3.2.1. Araç Modelleri

Eğitim sırasında performanstan ödün vermemek adına araba modelleri basit tutulmuştur. Aynı anda 30 bir birinin benzeri park ortamında öğretme işlemi gerçekleştirilecektir.



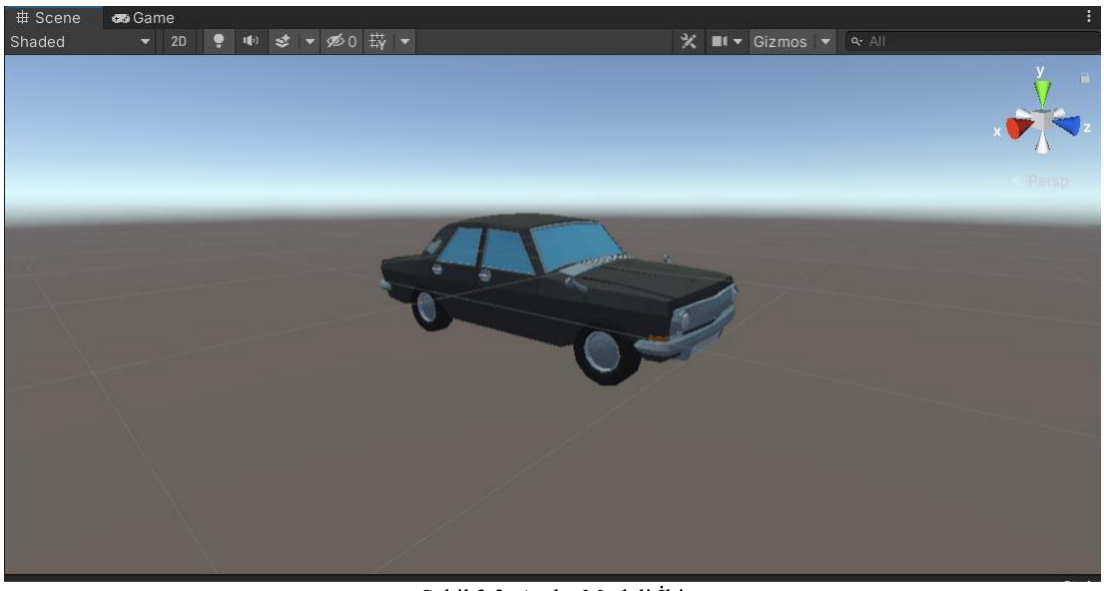
Şekil 3.1. Yapay Zeka Kontrolündeki Araba Modeli

Aracın fiziksel özelliklerini ve motor gücünü gerçek hayattaki ortalama bir arabayı baz alarak oluşturmaya çalıştım. Buna tekerleklerin maksimum dönme aralığı, aracın kendi ağırlığı, sürtünme katsayısı, maksimum hızı, tork miktarı gibi bir sıra özellikleri örnek gösterebilirim. Değerler ise aşağıdaki gibidir.



Şekil 3.2. Araba Özellikleri

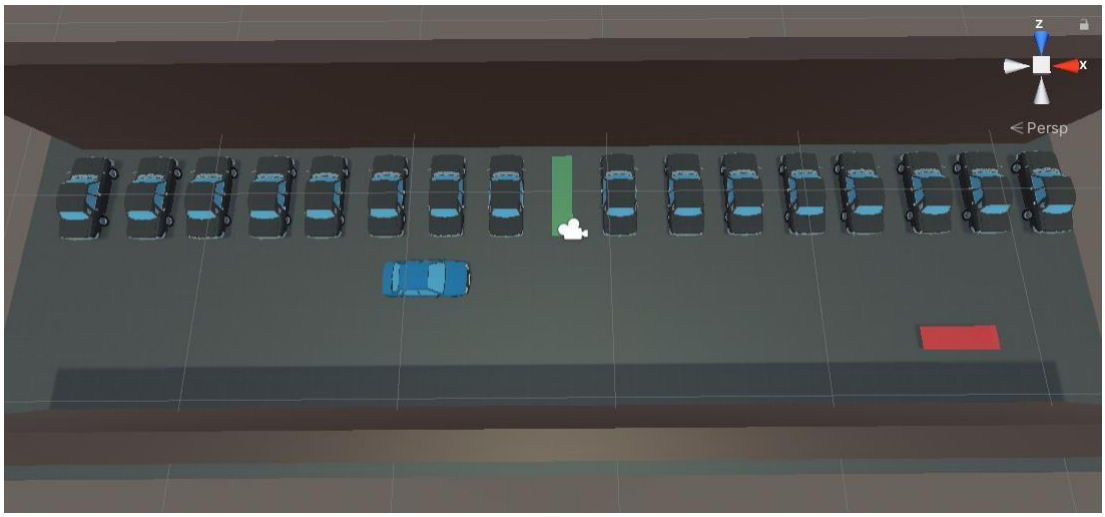
Karşıdan gelecek ve diğer park halindeki arabaları temsil edecek araç modeli de ayni şekilde düşük detaylı olacak şekilde oluşturulmuştur. Burada da aynı şekilde amaç gereksiz detaylar ekleyerek eğitim zamanı performanstan ödün verilmesinin önüne geçmektir.



Şekil 3.3. Araba Modeli İki

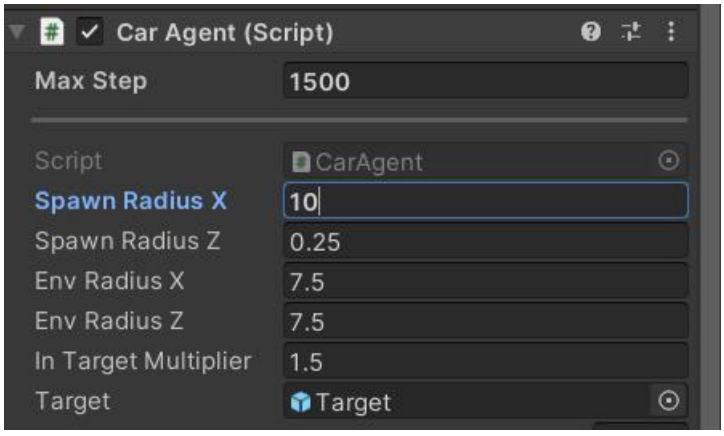
3.2.2. Harita Tasarımı

Harita tasarımı sadece işlemleri gerçekleştirecek şekilde oluşturulmuştur. Hedefimiz her turda yapay zeka modelini en kısa sürede başa çıkması gereken durumlara maruz kalmasını sağlamak olduğundan tasarımı da bu durumu göz önünde bulundurarak gerçekleştirmemiz gereklidir.



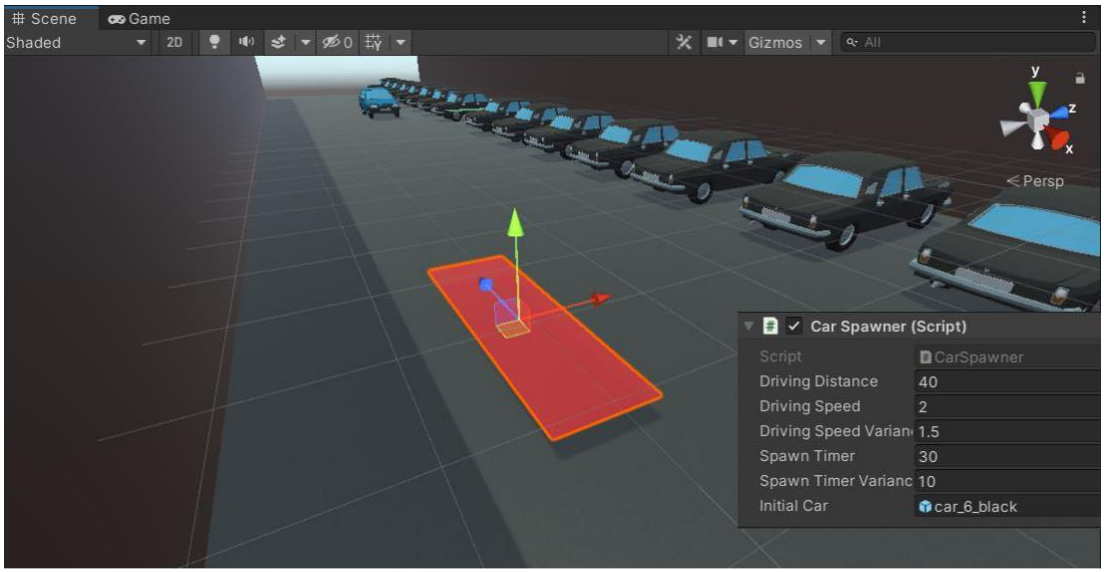
Şekil 3.4. Harita Tasarımı

Kırımız ile işaretli alan karşıdan gelecek olan aracın oluşturulma yerini temsil ediyor. Yeşil ile işaretli alan ise aracın park edilmesini istediğimiz yeri temsil ediyor.



Şekil 3.5. Oluşturulma Konumu

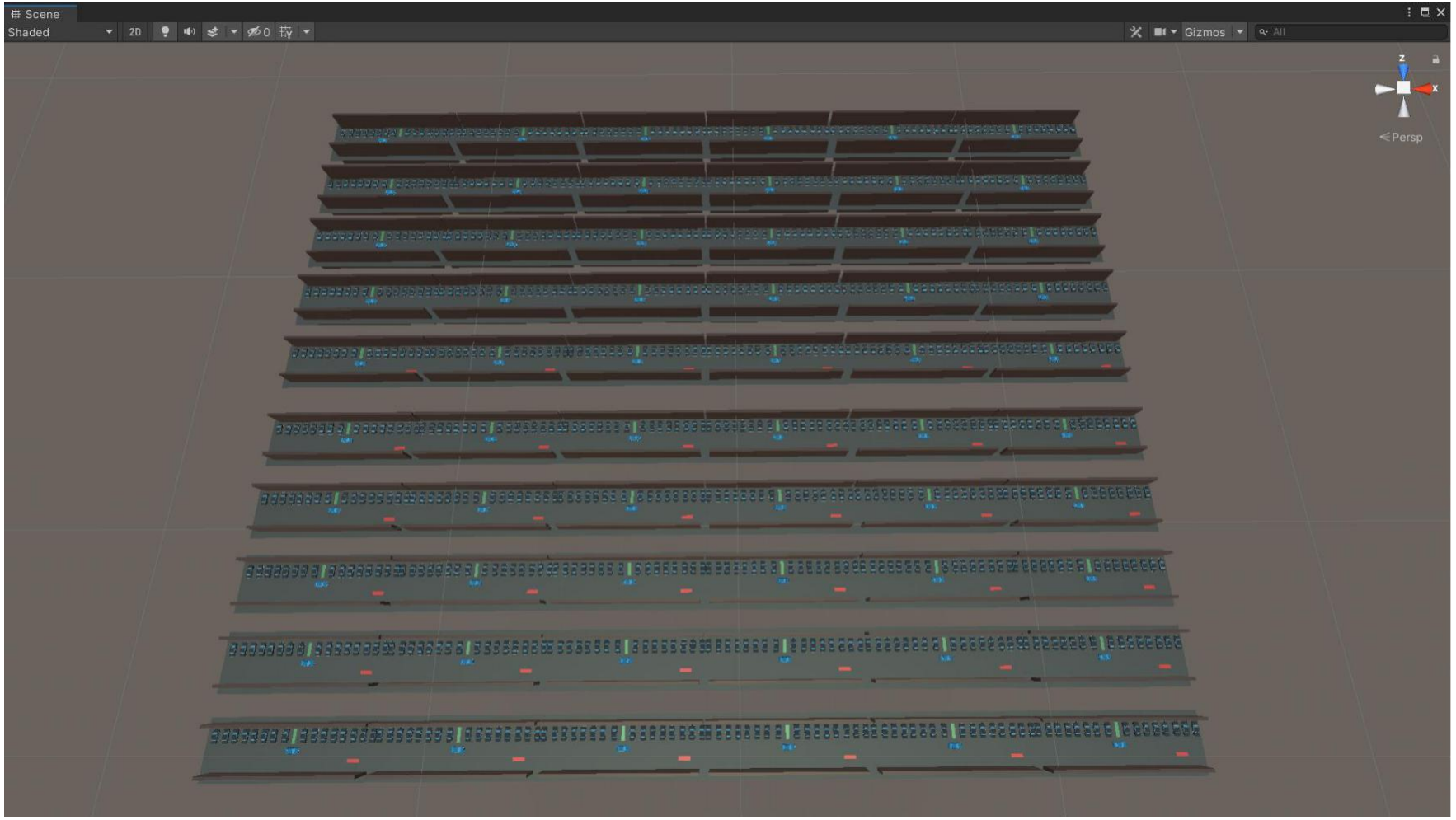
Yapay zeka kontrolündeki aracın başlangıç konumu sabit değildir ve x eksenine göre belirli aralık arasında rastgele oluşturulur.



Şekil 3.6. Araç Olışturucu

Karşıdan gelecek araca program başladığında oluşacak ve yok edildikten 10 saniye sonra kırmızı noktada tekrardan geri getirilecektir. Araç programın her evresinde düzenli olarak oluşturulmaya devam edilecektir. Oluşturulan araba modeli park halindeki araçlarla aynı model olacaktır.

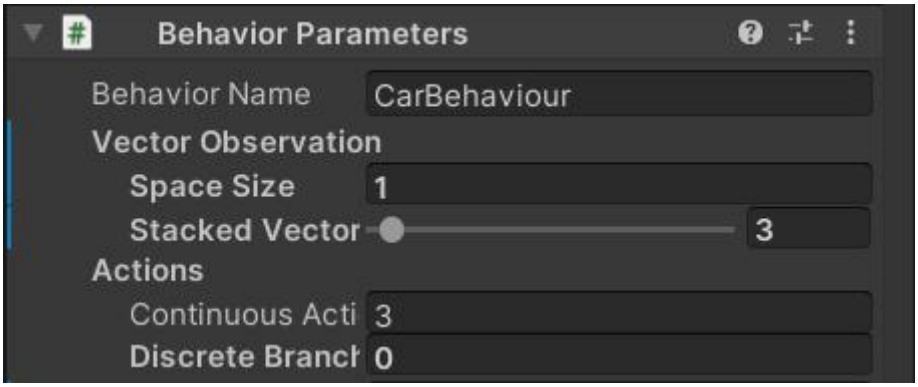
Öğrenme sürecini hızlandırmak ve yapay zeka modelini fazlasıyla farklı durumları içeren veriyle temin edebilmek için aynı anda birebir aynı özelliklerde 60 ortamda eğitme işlemi gerçekleştiriliyor.



Şekil 3.7. Haritanın Son Hali

BÖLÜM 4. ARAÇ PARK ETME İŞLEMİNİN SİMÜLE EDİLMESİ

Öncelikle basit araba kullanım işlemlerine bakalım. Örneğin sola veya sağa dönme işlemi. Daha önce araba kullandıysanız direksiyonun dönme işlemi zamanı aniden sağa veya aniden sola kırılmadığını fark etmişsinizdir. Sola dönme bir süreçtir ve düzenli olarak süreç boyunca devam etmesi gereken olaylar silsilesinden oluşur. Aynısı frene basma ve hızlanma işlemi için de geçerlidir. Bu mantığı araba kontrol sistemine eklemek için "Continuous Actions" özelliği kullanılır.



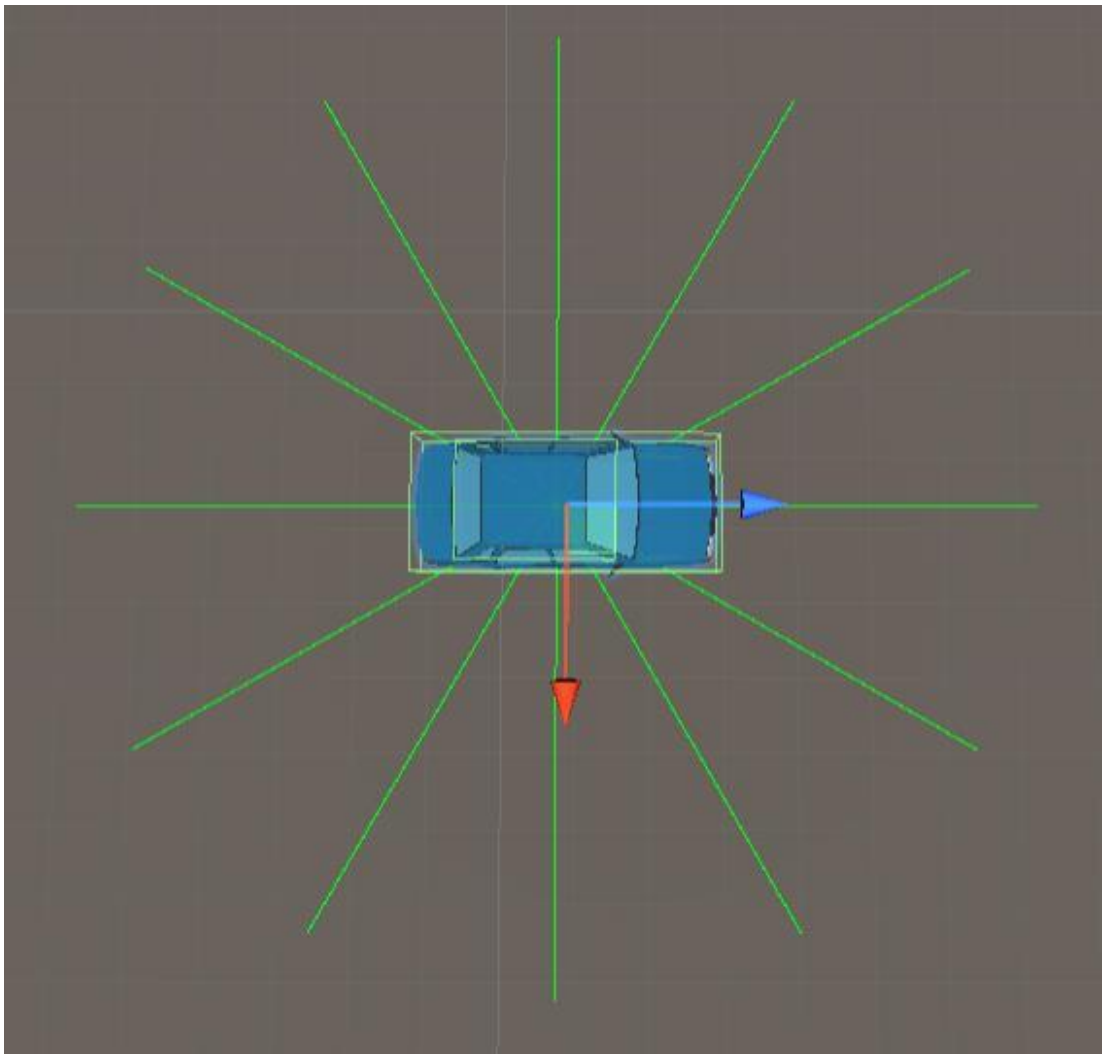
Şekil 4.1. Devam Eden İşlemler

4.1. Sensörler ve Kameralar

Araç üzerinde 12 adet sensör bulunur. Bu sensörler sol tarafta 5, sağ tarafta 5, arka tarafta bir ve ön tarafta da bir olacak şekilde yerleştirilmiştir. Araç üzerinde bulunan kamera sayısı ise 8dir. Kameralar da sol tarafta 3, sağ tarafta 3, önde ve arkada birer tane olmak üzere yerleştirilmiştir.

4.1.1. Uzaklık Sensörleri

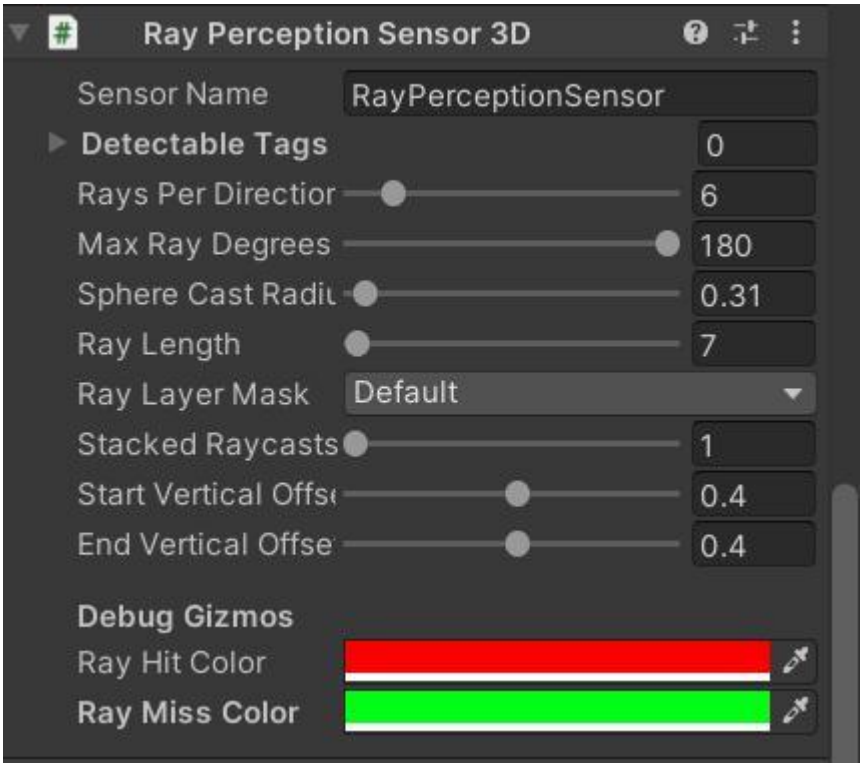
Ray Perception Sensor 3D kullanılarak uzaklık sensörü benzetimi yapılan ve araba üzerine yerleştirilen sensörlerin son hali bu şekildedir.



Şekil 4.2. Araç Sensörleri

Normal durumda yeşil olarak gösterilir yakında bir cisim tespit edildiğinde yavaş yavaş kırmızıya dönüşür.

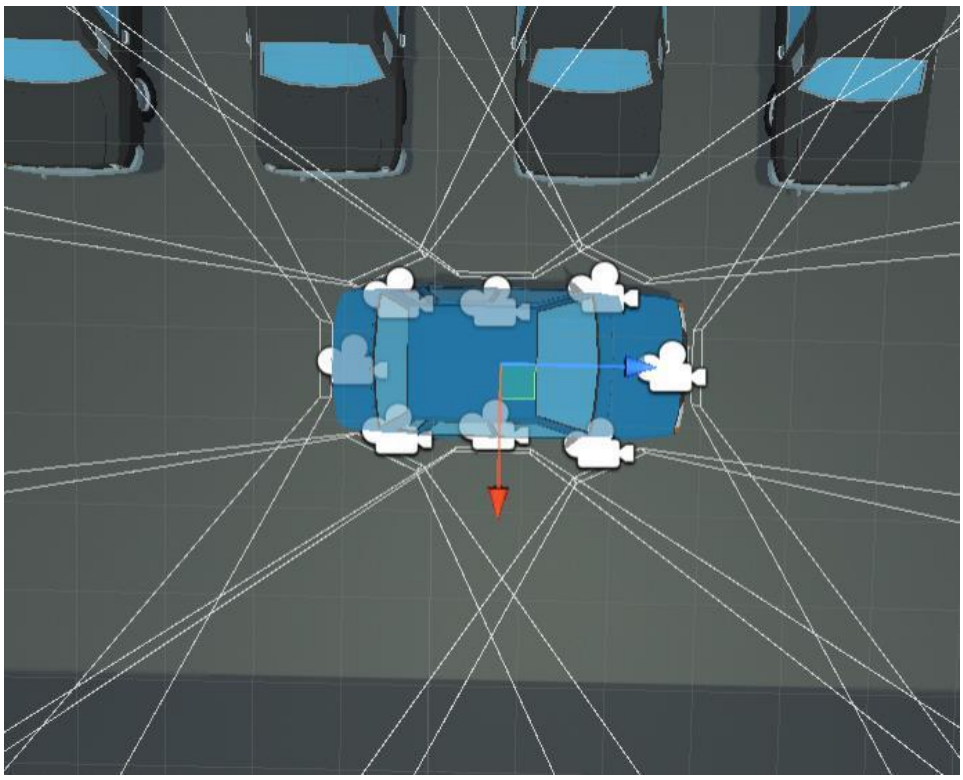
Sensör ayarları bu şekildedir.



Şekil 4.3. Sensör Ayarları

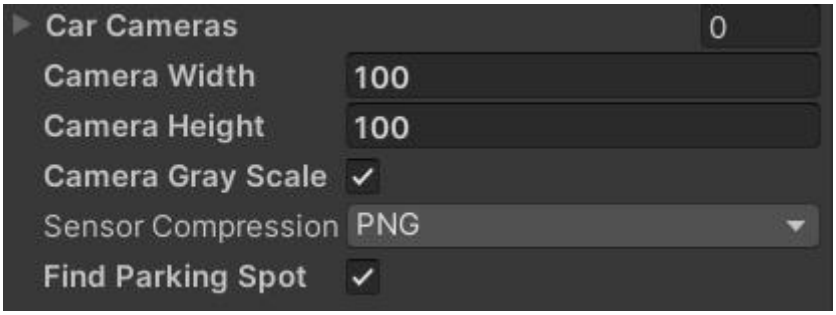
4.1.2. Araç Üzerindeki Kamerlar

Araç üzerinde sekiz adet kamera bulunmaktadır. Bu kameralar eğitim için yapay zeka modeline veri sağlamak için kullanılır.



Şekil 4.4. Araç Kameraları

Kameralar yerleştirilirken araç çevresinde hiçbir kör nokta kalmamasına özen gösterilmiştir. Kameraların ayarlarında da özelleştirmeler yapılmıştır. Kamera boyutları ve "Gray Scale" özelliği işlenecek veri boyutunu düşürmek ve performansı artırmak için güncellenmiştir. Kamera ayarlarının son hali bu şekildedir:



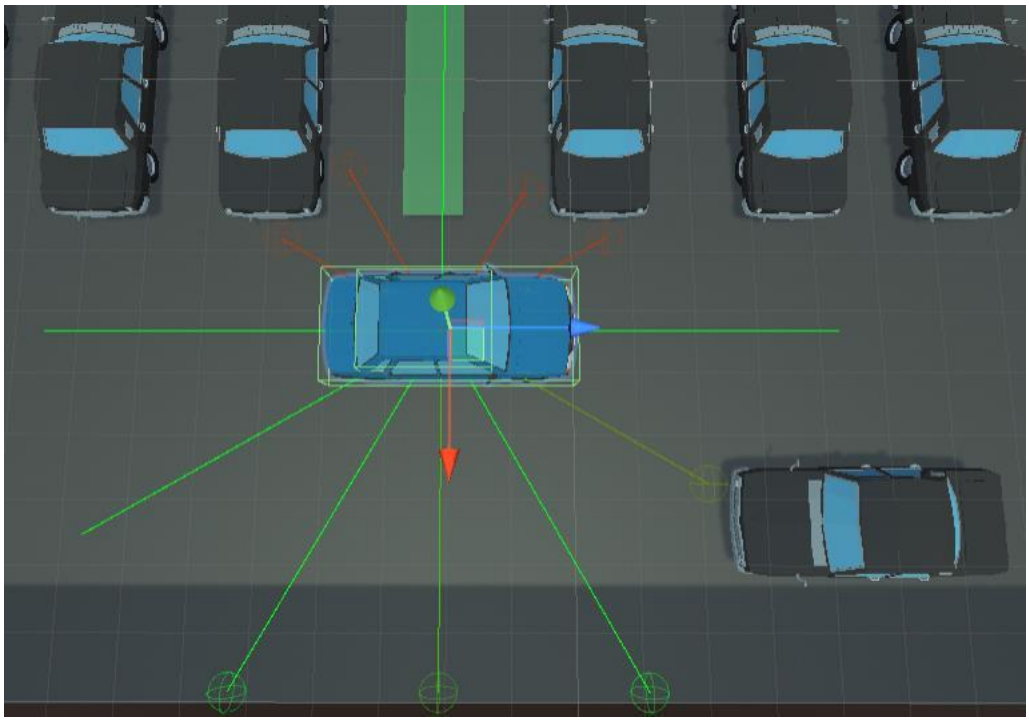
Şekil 4.5. Kamera Ayarları

4.2. Park İçin Uygun Alanın Belirlenmesi

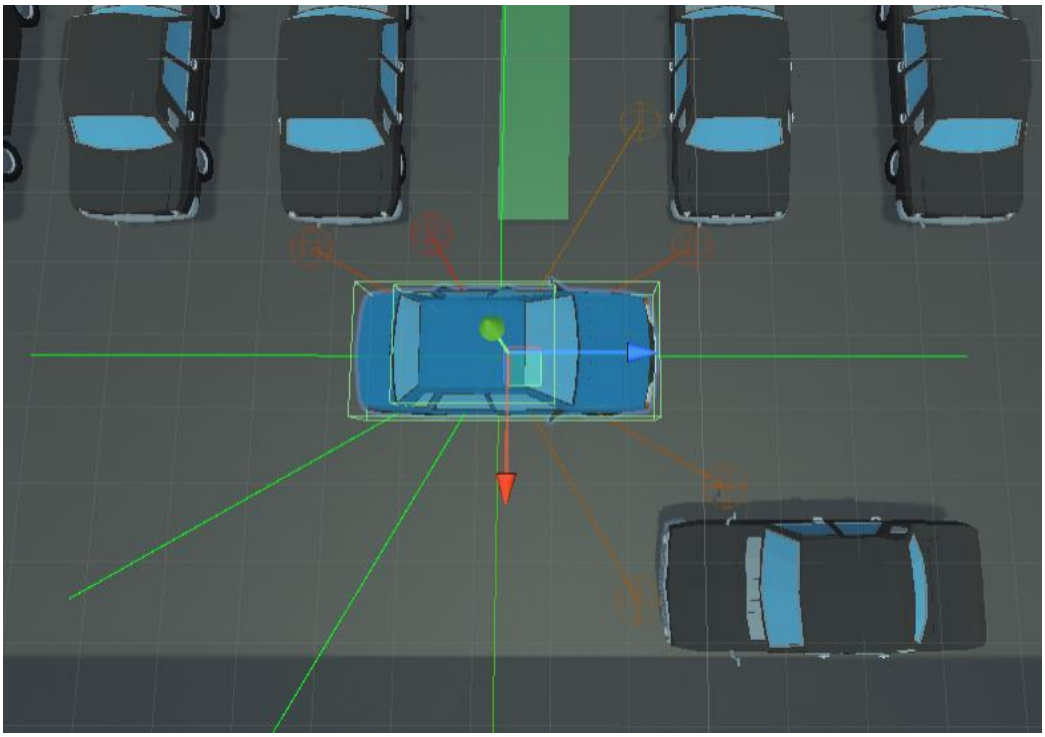
Araç park alanını dolaşır olası park alanlarını denetler. Denetleme işlemi "CarAgent.cs" script'i içerisindeki FindParkingSpot() fonksiyonu ile gerçekleştirir. Öncelikle aracın sağ ve sol merkezindeki uzaklık sensörleri her iki tarafta da aracın sığabileceği kadar boşluk var mı kontrol eder. Eğer boşluk varsa park için uyumlu olduğunu gösteren ve üç kriterle ölçülen her kriter sağlandığında bir değer arttırılan değişkenin değerini artırır. Bu ilk koşulun sağlandığını gösterir. Daha sonra aracın en oranının sığıp sığmadığı köşelerdeki sensörler yardımıyla denetlenir ve bu koşul da sağlanırsa değişkenin değeri bir arttırılır. Araç park alanının tam merkezine ulaştığında ve diğer iki koşul sağlandığında park etme işlemi başlatılabilir. Bunun için de aracın bulunan park alanına göre tam konumu kontrol edilir. Bunu da yine uç noktalardaki sensörler arasındaki fark yardımıyla hesaplarız. Bu koşul da sağlandığında değişkenin değeri bir arttırılır. Her üç koşul sağlanmışsa işlemler sonunda değişkenin değeri 3'e eşit olur bu da uygun park alanın bulunmuş olduğunu gösterir. Park alanı bulunduğu için "isLookingForSpot" değişkeni false yapılır. Bunu yapmaktaki amaç daha fazla park alanı aramaya gerek kalmadığından park alanı arama işleminin durdurulmasını sağlamaktır.

4.3. Dış Etkenlerin Denetimi

Araç uzaklık sensörlerindeki değişimin kendi kontrolünde nasıl değişeceğini bilir ama farklı bir cisim hareket halinde olduğunda uzaklık sensörlerindeki değişim beklenildiği gibi olmaz, bu da başka bir cisimin hareket ettiğini anlamasını sağlar.



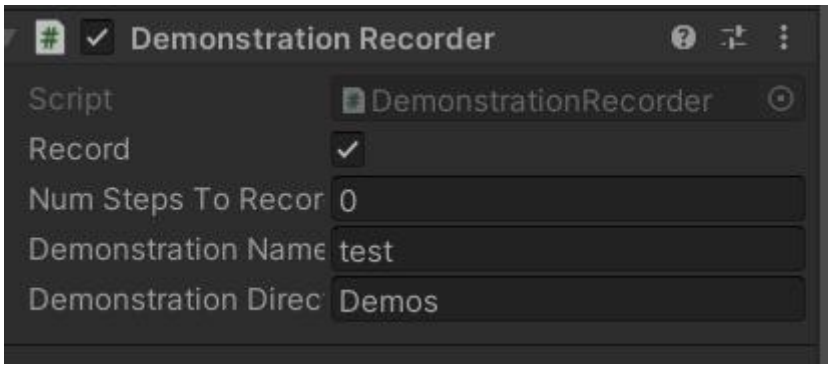
Şekil 4.6. Dış Etken Kontrolü



Şekil 4.7. Dış Etken Kontrolü 2

4.4. Oluşturulan Ortamın Öğrenmeye (Training) Hazırlanması

"Demonstration Recorder" kullanılarak eğitim sırasında örnek olarak verilecek park etme işlemi oluşturulur. Bu sayede yapay zeka modelinin sadece ileri gitmeyi öğrenmek için bile fazladan neredeyse 10000 adım harcaması önlenir. Burada yapılan işlem kullanıcı kontrolünde araç park ettirmek ve bu park etme işlemi sırasında gerçekleşen tüm olayları (hızlanma, frene basma, sola dönme, park etme) kayıt altına almak ve eğitim sırasında kullanmaktır.



Şekil 4.8. Eğitim Hazırlığı

BÖLÜM 5. SONUÇLAR VE ÖNERİLER

Eğitim sonrası oluşturulan model projenin hedeflediği gibi, dış etkenleri dikkate alarak park alanı bulma ve park etme işlemlerini gerçekleştirmeyi başardı. Bunu yaparken oluşturulan ortam ve araç mekanikleri gerçek hayat benzetilmesi yapılarak oluşturuldu. Araç sensörleri ve kameralar kullanılarak yapay zeka eğitimi için veri sağlandı.

Projeyi geliştirmek ve daha detaylı bir park sistemi oluşturmak için birden fazla bağımsız dış etken eklenebilir. Örneğin park alanı aranırken farklı bir aracın park alanını boşaltmaya başlaması durumu, park etme sırasında yayaların park alanından geçmesi durumu veya park alanında küçük cisimlerin bulunması durumu gibi dikkate alınması gereken yeni olaylar eklene bilir.

KAYNAKLAR

[1] https://docs.conda.io/projects/conda/en/stable/

[2] https://pytorch.org/features/

[3] https://aws.amazon.com/tr/what-is/python/

[4] https://unity.com

[5] https://github.com/Unity-Technologies/ml-agents

[6] https://www.techtarget.com/searchenterpriseai/definition/reinforcement-learning

[7] https://sites.google.com/view/icml2018-imitation-learning/

[8] A reduction of imitation learning and structured prediction to no-regret online learning (Ross, Gordon & Bagnell, AISTATS 2011)

[9] Ng, Andrew Y., and Stuart J. Russell. "Algorithms for inverse reinforcement learning." ICML. 2000.

ÖZGEÇMİŞ

BSM498 BİTİRME ÇALIŞMASI
DEĞERLENDİRMEVE SÖZLÜ SINAVTUTANAĞI

KONU : Makine Öğrenmesi Kullanarak Yapay Zekaya Otonom Araç Park Ettirme
ÖĞRENCİLER (B181210553/OMAR/GARİBOV):

Değerlendirme Konusu	İstenenler	Not Aralığı	Not
Yazılı Çalışma			
Çalışma klavuza uygun olarak hazırlanmış mı?	x	0-5	
Teknik Yönden			
Problemin tanımı yapılmış mı?	x	0-5	
Geliştirilecek yazılımın/donanımın mimarisini içeren blok şeması (yazılımlar için veri akış şeması (dfd) da olabilir) çizilerek açıklanmış mı?			
Blok şemadaki birimler arasındaki bilgi akışına ait model/gösterim var mı?			
Yazılımın gereksinim listesi oluşturulmuş mu?			
Kullanılan/kullanılması düşünülen araçlar/teknolojiler anlatılmış mı?			
Donanımların programlanması/konfigürasyonu için yazılım gereksinimleri belirtilmiş mi?			
UML ile modelleme yapılmış mı?			
Veritabanları kullanılmış ise kavramsal model çıkarılmış mı? (Varlık ilişki modeli, noSQL kavramsal modelleri v.b.)			
Projeye yönelik iş-zaman çizelgesi çıkarılarak maliyet analizi yapılmış mı?			
Donanım bileşenlerinin maliyet analizi (prototip-adetli seri üretim vb.) çıkarılmış mı?			
Donanım için gerekli enerji analizi (minimum-uyku-aktif-maksimum) yapılmış mı?			
Grup çalışmalarında grup üyelerinin görev tanımları verilmiş mi (iş-zaman çizelgesinde belirtilebilir)?			
Sürüm denetim sistemi (Version Control System; Git, Subversion v.s.) kullanılmış mı?			
Sistemin genel testi için uygulanan metotlar ve iyileştirme süreçlerinin dökümü verilmiş mi?			
Yazılımın sızma testi yapılmış mı?			
Performans testi yapılmış mı?			
Tasarımın uygulamasında ortaya çıkan uyumsuzluklar ve aksaklıklar belirtilerek çözüm yöntemleri tartışılmış mı?			
Yapılan işlerin zorluk derecesi?	x	0-25	
Sözlü Sınav			
Yapılan sunum başarılı mı?	x	0-5	
Soruları yanıtlama yetkinliği?	x	0-20	
Devam Durumu			
Öğrenci dönem içerisindeki raporlarını düzenli olarak hazırladı mı?	x	0-5	
Diğer Maddeler			
Toplam			

DANIŞMAN(JÜRİADINA):
DANIŞMANİMZASI: