MOBILE ROBOT NAVIGATION WITH DEEP REINFORCEMENT LEARNING

DERİN PEKİŞTİRMELİ ÖĞRENME İLE MOBİL ROBOTLARDA OTONOM HAREKET

Hüseyin Pullu [1 *], Cihan Karakuzu [2]

[1,2] Bilecik Şeyh Edebali University, Computer Engineering, Turkey

[1] ORCID ID: 0000-0001-9124-9597, huseyinpullu@outlook.com

[2] ORCID ID: 0000-0003-0569-098X, cihan.karakuzu@bilecik.edu.tr

**** Corresponding Author***

Abstract

Reinforcement learning is a learning method based on choosing actions to maximize rewards in an environment. This form of behavioral learning, which is widely observed in nature, gives successful results for autonomous tasks in environments where there is no information about it. This study is about mobile robots trained for autonomous tasks using reinforcement learning, a branch of machine learning. Necessary environments were created with the Robot Operating System (ROS) and Gazebo simulation environment, which is a 3D modeling and processing tool that is frequently used in the ROS platform. The robot, which has no knowledge about the environment, tries to learn how to find the target points with Reinforcement Learning. In our study, the autonomous mobility of the TurtleBot3 mobile robot in simulation environments that contains fixed or dynamic barriers, was gained by reinforcement learning. Due to the limitations of reinforcement learning, Deep Reinforcement Learning, which uses neural networks instead of memory for complex problems, was preferred in our study. Training was conducted with the TurtleBot3 mobile robot in two different environments with fixed and dynamic barriers created in the gazebo environment. According to the epsilon greedy policy, one of the main features of reinforcement learning is a model that can find the target point repeatedly and can increasable the total reward was created by leaving the movements that were made randomly at the beginning to the movements decided by the neural network model in the later stages, and the results were shared with graphs.

Keywords: *Deep reinforcement learning, autonomous navigation, mobile robot, Intelligent route planning.*

1. Giriş

Pekiştirmeli Öğrenme (Reinforcement Learning), hareketi yapan unsurun (ajan) bir çevre ile etkileşime girerek ortamı keşfettiği ve mümkün olan en yüksek toplam ödülü elde etmek amacıyla bu etkileşimlerin değerlendirildiği makine öğrenimi alanıdır. Bu yöntem canlıların ortamdan olumlu ya da olumsuz geri bildirim alarak; öğrenmek için kullandıkları deneme yanılma yöntemini taklit eder.

Robotik alanında pekiştirmeli öğrenme, karmaşık ve mühendisliği zor davranışların tasarımı için bir çerçeve sunar [1]. Buradaki zorluk, bu makine öğrenimi tekniklerinin doğrulanabileceği ve daha sonra gerçek bir senaryoda uygulanabileceği basit bir ortam oluşturmaktır. Bu sorunun üstesinden gelmek için çalışmamızda, robot uygulamaları için yazılım geliştirme amacıyla çok kullanılan Robot İşletim Sistemi (ROS) ve 3 boyutlu modelleme ve işleme aracı olan Gazebo benzetim ortamı ile gerekli ortam oluşturulmuştur.

Çalışmamızda, Derin Pekiştirmeli Öğrenme (DQN) yöntemi kullanılarak Gazebo benzetim ortamında, TurtleBot3 mobil robotu ile engellerden kaçınma ve hedefi bulma görevlerini yerine getirmek için eğitim yapılmıştır. Pekiştirmeli öğrenme de iki temel özellik, hareketi yapan unsur "ajan" ile bulunulan ortam "çevre" dir. DQN ajanı, robotu engellere çarpmadan hedefine ulaştırmak için, hedefine yaklaştığında pozitif bir ödül ve hedefinden uzaklaştığında ise negatif bir ödül yani ceza verir. Burada amaç toplam ödülü en yükseğe çıkarmaktır. TurtleBot3 robotu, hedefe ulaşmadan ve engellerden başarılı bir şekilde kaçınmadan önce çok sayıda eğitim tekrarı gerektirir [2].

Çalışmanın geri kalanı şu şekilde düzenlenmiştir. Bölüm 2'de, çalışmanın altyapısında kullanılan ROS, Gazebo, DQN gibi teknoloji ve yöntemlerden bahsedilmiştir. Bölüm 3'te kullanılan mimariden, ROS, Gazebo ve DQN bağlantısından bahsedilmiştir. Bölüm 4'de DQN modelinde oluşturulan çerçeve yapıdan bahsedilmiş ve detaylandırılmıştır. Bölüm 5'te DQN modeli ile yapılan çalışmanın parametreleri ve benzetim sonuçları paylaşılarak analiz edilmiştir. Son olarak, Bölüm 6'da sonuç ve araştırma için gelecekteki bazı olası çalışmalar önerilmektedir.

2. Teknoloji ve Yöntemler

2.1. Robot İşletim Sistemi

ROS (Robot Operating System) [3], robotların tasarımı, geliştirilmesi ve kontrolü için kullanılan açık kaynaklı yazılım platformudur. Robotlar için modüler yazılım geliştirme için bir araç seti sağlar ve robotik uygulamaları hızlandırmak için hazır modülleri içerir. Açık kaynaklı olması ve geniş bir kullanıcı tabanına sahip olması nedeniyle, araştırmacılar ve geliştiriciler tarafından tercih edilir. ROS, robotik uygulamaların hızlı bir şekilde prototip edilmesine ve test edilmesine izin verirken, aynı zamanda daha karmaşık sistemler için ölçeklenebilir bir platform sunar.

2.2. Gazebo

Gazebo, gerçekçi bir benzetim ortamında çoklu robot sistemlerini modellemek, test etmek ve görselleştirmek için kullanılan bir araçtır [4]. ROS ile senkron çalışması ve kullanımı kolay olması nedeniyle yapay zeka ve robotik araştırmacıları tarafından çok tercih edilir.

Gazebo, robotik alanında gerçekçi bir deneyim sunmak için fizik tabanlı benzetim motorunu kullanır [4]. Bu, robotların gerçek dünyada nasıl davranacağını daha doğru tahmin etmeyi sağlar. Görsel olarak zengin ve etkileyici bir benzetim sunmak için güçlü bir grafik ve görselleştirme motoruna sahiptir. Çeşitli sensör türlerini benzetme ve entegre etme yeteneğine sahiptir. LIDAR (Light Detection and Ranging), kameralar, hız ölçerler ve diğer sensörler gibi farklı algılayıcıları benzeterek, robotların çevrelerini algılama ve tepki verme yeteneklerini değerlendirmek mümkündür. Gerçek dünya senaryolarını taklit etmek için benzetim ve kontrol arasında senkronizasyon sağlar. Bu, gerçek robotların kontrol algoritmalarını benzetim ortamında test etmek için kullanışlıdır. Aynı anda birden fazla robotun benzetimini destekler.

2.3. TurtleBot3

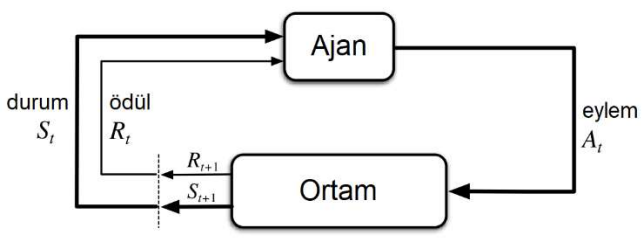
TurtleBot3, düşük maliyeti ve taşınabilirliği ile öne çıkan robot platformudur [5]. Açık kaynaklı ROS platformuyla uyumludur ve birçok farklı uygulama alanında kullanılabilmektedir. Özellikle robotik ve yapay zeka alanında öğrenim ve araştırma faaliyetleri için tasarlanmıştır. TurtleBot3, çeşitli sensörleri ve aksesuarları ile özelleştirilebilir. Burger modelinin kilometre sayacı, Imu ve LIDAR sensörleri sayesinde çevre ve konum bilgilerini öğrenebiliriz [6].

2.4. Pekiştirmeli Öğrenme

Pekiştirmeli öğrenme (Reinforcement Learning), problemle ilgili eğitim verisi ya da probleme özgü yeterince uzmanlık olmadığında kullanılan, çevre ile etkileşime giren makine öğrenmesi yöntemidir. Pekiştirmeli Öğrenme de ihtiyacımız olan şey bir ödül mekanizmasıdır. Yeterince tekrar etmesine izin verilen model ödülü nasıl en üst düzeye çıkaracağını bulacaktır.

2.4.1. Q-Öğrenme

Q-Öğrenme, makine öğrenimi alanında güçlü bir öğrenme algoritmasıdır [7]. Pekiştirmeli öğrenmenin temelinde Q-Öğrenme yer alır. Ajanın yapacağı eyleme karar vermek için Q-değerleri kullanılır. Ajan (agent), bir durumda (state) bir eylem (action) gerçekleştirerek ödül alır ve Q-değerini günceller. Q-değeri, eylem gerçekleştirildiğinde beklenen toplam ödülü tahmin eder. Q-değerleri, ajanın hedefe en hızlı şekilde ulaşmak için hangi eylemi seçmesi gerektiğini gösterir. Bu şekilde, ajan zamanla optimal bir davranış stratejisi geliştirir.



Şekil 1. Markov Karar Süreci [8]

Q-Öğrenme'nin merkezinde Bellman denklemi ve Markov Karar Süreçleri (MDP) yer alır [8]. MDP, pekiştirmeli öğrenmenin modellenmesi için kullanılan matematiksel çerçevedir. Pekiştirmeli Öğrenmeyi Şekil 1'de görülen Markov Karar Süreci ile ifade etmek mümkündür. Markov özelliğine göre gelecek durum (S_{t+1}), yalnızca ortamın olası durumlarını içeren (S) uzayında bulunulan mevcut duruma ve (A_t) eylem uzayından seçilen eyleme bağlıdır. Ayrıca yapılan eylemden sonra olumlu ya da olumsuz bir ödül (R_{t+1}) alınır. Önce ki durumların ve eylemlerin bir etkisi yoktur. Bellman denklemleri, Markov Karar Süreçleri çerçevesinde en iyi kararları almak için kullanılan matematiksel formül olarak ortaya çıkar.

Denklem (1)'de verilen Bellman denklemi, durum-eylem çifti değerlerini "$Q(s,a)$" tahmin etmek için kullanılır [8]. Denklem, gelecekteki durumların maksimum değerlerini kullanarak bir durumdaki değeri geriye doğru güncellemeyi sağlar. Bu iteratif güncelleme süreci, durum değerlerini veya durum-eylem çifti değerlerini kararlı bir şekilde yaklaşık olarak bulmaya yardımcı olur.

$$Q(s,a) = R(s,a) + \gamma * max_{a'}(Q(s',a')) \quad (1)$$

Zamansal fark öğrenimi (Temporal Difference Learning) [8], bir durumda bir eylem gerçekleştirildiğinde ajanın gerçekleştirdiği eylemin kalitesini (değerini) tahmin etmek için kullanılır. Algoritma, gerçekleştirilen eylemin beklenen değeri ile gerçekte elde edilen değer arasındaki farkı hesaplar ve bu farkı, gerçekleştirilen eylemin kalitesini Q değerini güncellemek için kullanır. Denklem (2)'de, zamansal fark öğrenimi fonksiyonu görülmektedir.

$$Q'(s,a) = Q(s,a) + \alpha * \left(r_t + \gamma * max_a Q(s',a') - Q(s,a)\right) \quad (2)$$

Denklem (2) de:

(s, a): durum-eylem çifti

(s', a'): bir sonraki durum-eylem çifti

$Q(s, a)$: Q değeri, s durumunda a eylemi ile elde edilmesi beklenen toplam ödülü ifade eder.

r_t: mevcut durumda alınan ödülü ifade eder

γ: İndirim faktörü gelecekte kazanılması muhtemel ödüllerin ne kadar önemli olduğunu belirlememize yardımcı olur.

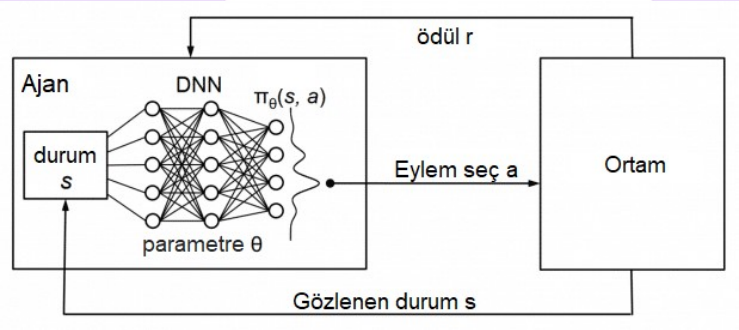
α: Öğrenme oranı, modelin öğrenme hızını ayarlar.

Q değerlerini saklamak için Q tablosu denilen bir tablo kullanılır. Tablonun her satırı bir durumu, her sütunu bir eylemi temsil eder. Her hücre, o durumda o eylemin beklenen toplam ödülünü temsil eden Q değeridir. Q tablosu, ajanın öğrenirken bu değerleri güncellediği veri yapısıdır.

Q tablosu, küçük problemler için uygulanabilir bir yöntemdir. Büyük ölçekli problemlerde boyutu çok büyük olabilir ve ağır hesaplama gerektirebilir. Bu nedenle, daha büyük problemler için Q-Öğrenme algoritması genellikle derin öğrenme yöntemleriyle birleştirilir ve Deep Q Network (DQN) gibi yöntemler kullanılır. DQN, derin sinir ağı kullanarak Q değerlerini tahmin etmek için yapay sinir ağı kullanır [9].

2.4.2. Derin Pekiştirmeli Öğrenme

Mnih ve arkadaşları [9], Pekiştirmeli Öğrenmeyi derin sinir ağları ile birleştiren, Derin Pekiştirmeli Öğrenme modelini geliştirdiler. Çalışma da kurdukları ağ yapısına Deep Q Network (DQN) adını verdiler. Bu ağın genel yapısı Şekil 2'de gösterilmiştir.



Şekil 2. Derin Pekiştirmeli Öğrenme (DQN) yapısı

Derin sinir ağları, yüksek boyutlu verilerle başa çıkabilme ve özellikleri daha iyi öğrenebilme yeteneğine sahipken, pekiştirmeli öğrenme bir sistemi yeni görevlerle ve yeni ortamlarla harekete geçirebilmeyi sağlamaktadır [10].

DQN bir durumda alınabilecek tüm aksiyonların Q-değerlerini tahmin etmek için derin sinir ağını kullanır. Bu Q-değerleri, bir durumda belirli bir aksiyonun yapılması durumunda beklenen toplam ödülü temsil eder.

DQN'in çalışması, ajanın çevreyle etkileşimde bulunduğu ve belirli aksiyonlar aldığı bir döngüye dayanır. Ajan, çevreden bir gözlem alır, gözlemi derin sinir ağına besler ve Q-değerlerini tahmin etmek için derin sinir ağından çıktı alır. Alınan çıkış değerlerinden en büyüğü beklenen toplam ödülü temsil ettiği için ilgili çıkışa ait eylem seçilir. Ardından epsilon-greedy (ε) stratejisine göre belirlenen eylemi uygular. Epsilon-greedy ajanın keşfetme olasılığıdır. Genellikle 1 veya 1'e yakın yüksek bir değerle başlar ve her bölümde azaltılır. Bu, ajanın eğitimin başında çevreyi keşfetmesine yardımcı olur. Ajan, eğitim sırasında ε 'yi azaltır ve ilerleyen bölümlerde daha çok modeli kullanarak eyleme karar vermeye başlar [11].

Modeli eğiten durumlar arasındaki korelasyonu azaltmak için Deneyim Yeniden Oynatma (Experience Replay) Mekanizması kullanılır [12]. Mekanizma geçmişteki ödül, eylem ve durumları, Replay Memory olarak da adlandırılan tampon belleğe koymak ve eğitim için tampondan bazı rastgele örnekler almak şeklinde çalışır.

Q-Öğrenme'de kullandığımız 2 nolu denklemi, Q-tablosunun yerine kullandığımız sinir ağı sayesinde basitleştirebiliriz. Geri yayılan optimize edicimiz zaten buna sahip olacağından, öğrenme oranına artık ihtiyaç yoktur. Öğrenme oranı kaldırıldığında, birbirlerini sıfırladıkları için iki $Q(s, a)$ terimini de kaldırırız ve Q değerini hesaplayan Denklem (3)'ü elde ederiz.

$$Q(s, a) = R(s, a) + \gamma * max_a Q(s', a') \quad (3)$$

Denklem (3) Q fonksiyonu olarak isimlendirilir. Denklem de:

s: t anındaki durum verisi.

a: Eylem uzayından seçilen eylem.

s': $t+1$ anında a eylemi ile geçilen yeni durum.

$R(s,a)$: s durumunda a eylemi ile alınan ödülü ifade eder.

γ: İndirim faktörü gelecekte kazanılması muhtemel ödüllerin ne kadar önemli olduğunu belirlememize yardımcı olur.

$Q(s,a)$: Q değeri, s durumunda a eylemi ile elde edimesi beklenen toplam ödülü ifade eder.

Denklem (4) ağın çıkış fonksiyonudur. Burada $(1-done)$ kısmı dikkat çeker. "done", bölümü sonlandırmak için kullanılan boolean değişkendir. Robot engele çarptığında ya da zaman aşımında "true" değerini alır böylece bölüm sonlanır. Bu durumda çıkış, alınan ödüle eşit $(y=r)$ olur. Bunun nedeni bölüm sonlandığında başa dönüleceğinden sonraki durum olmayacaktır. Bu nedenle sonraki durum hesaba katılmaz çıkış değeri ödül değerini alır.

Hesaplanan gerçek Q değerleri, sinir ağının tahmini Q değerleriyle karşılaştırılır ve ağı eğitmek için Denklem (5)'de verilen kayıp fonksiyonu kullanılır. Bu kayıp fonksiyonu, ağın ağırlıklarını güncellemek için kullanılır ve ağın tahmini Q-değerini gerçek Q-değerine yaklaştırmaya çalışır [11].

$$y = r + \gamma * max_a Q(s', a'; \theta) * (1 - done) \quad (4)$$

$$L = \left(y - Q(s, a; \theta)\right)^2 \quad (5)$$

Algoritma: Deneyim yeniden oynatma (Experience Replay) mekanizması ile Derin Pekiştirmeli Öğrenme [9]

Yeniden oynatma belleği D'yi N kapasitesiyle tanımla

model'i rastgele ağırlıklarla başlat

for bölüm = 1, M **do**

Ortamı sıfırla ve durum değerini al = s_1, *done* = *false*

for t = 1, T **do**

a_t eylemini, ε değeri olasılığınca rastgele ya da $a_t = max_a Q(s_t, a; \theta)$ hesabı ile seç

Ortamda a_t eylemini uygula ve r_t, s_{t+1} ve *done* (boolean) değerlerini al

Yeniden oynatma belleği D 'ye $(s_t, a_t, r_t, s_{t+1}, done_t)$ verilerini ekle

D 'den rastgele mini yığın örneği al $(s_j, a_j, r_j, s_{j+1}, done_j)$

$y_j = r_j + \gamma\, max_{a'} Q(s_{j+1}, a'; \theta) * (1 - done_j)$

$L_j = (y_j - Q(s_j, a_j; \theta))^2$ kayıp fonksiyonu ile modeli eğit

$s_t = s_{t+1}$

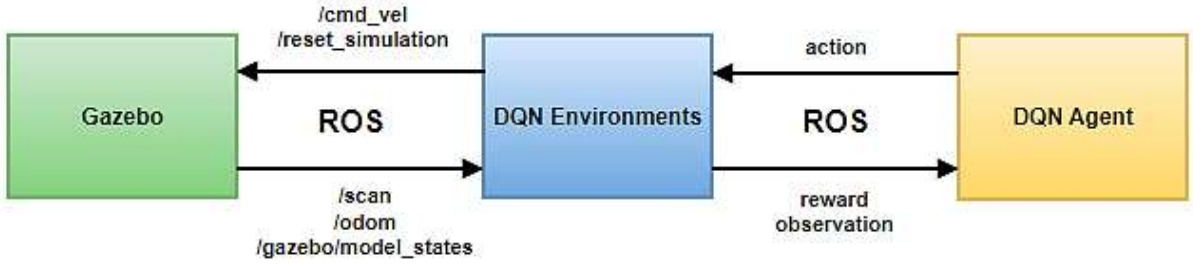
end for

$\varepsilon = \beta\varepsilon, \varepsilon'yi\ \beta$ ile güncelle

end for

3. Mimari

Gazebo ve diğer kullanılan yöntemler arasındaki ilişkiyi gösteren öbek şema Şekil 3'de sunulmaktadır. Bu mimari, ROS ortamında düğümlerin, belirli bilgiler sağlayan konular aracılığıyla etkileşime girdiği bir yapıdır. İlk olarak, Gazebo benzetim ortamının iletişim arayüzü düğüme bağlanır. Daha sonra, DQN modeli, denetim stratejisine dayalı eylem talimatlarını "/cmd_vel" konusu aracılığıyla "/gazebo" ya geri bildirim olarak gönderir. Son olarak, Gazebo TurtleBot3'ü çalıştırır ve TurtleBot3'ün tüm durumlarını ve benzetim ortamını günceller [13].



Şekil 3. Uygulama da kullanılan mimari

ROS'da çalıştırılabilen her birim, düğüm olarak tanımlanır. Düğümler arası veri alışverişi konular üzerinden mesajlar ile yapılır. Örneğin robotu yönlendirmek için, "/cmd_vel" konusu üzerinden "twist" ile açısal ve doğrusal hız mesajları motorları denetleyen düğüme iletilir. Böylece robot istenilen hareketi yapar.

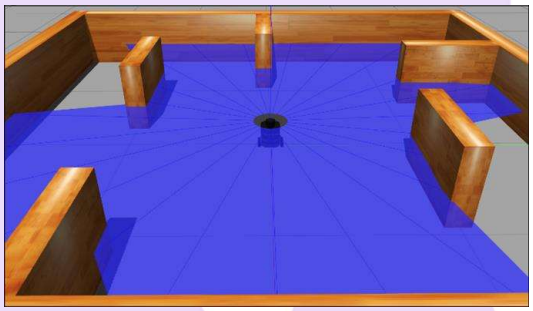
"/reset_simulation", ROS içinde Gazebo benzetim ortamını sıfırlamak için kullanılan servistir. Gazebo dünyasını, robotların konumlarını, hızlarını ve durumlarını başlangıç değerlerine geri döndürür. Robot bir engele çarptığında olumsuz ödül verir ve bölüm sonlandılarak benzetim ortamı sıfırlanır.

"/gazebo/model_states" hedef noktanın konumunu ve benzetim ortamı hakkında diğer bilgileri sağlayan servistir.

"/scan" ile robotun LIDAR verilerini alırız. TurtleBot3'de 360 derece LIDAR sensörü bulunur [6]. LIDAR sensörü tarafından elde edilen veriler benzetim ortamı hakkında bilgi sağlar. TurtleBot3 Burger modelinde LIDAR sensörün minimum algılama mesafesi yaklaşık 12cm, maksimum ise 3,5m'dir.

"/odom" konusu üzerinden, odometry bilgileri alınır. Odometry, robotun konum, hız ve rotasyon bilgilerini tahmin etmek için kilometre sayacı, imu sensörü, hız sensörü ve kamera vb. sensör verilerini kullanır.

Şekil 4'de sanal ortamda LIDAR kullanarak otonom olarak gezinmeyi öğrenen TurtleBot3 görülmektedir.



Şekil 4. Sanal bir ortamda TurtleBot3

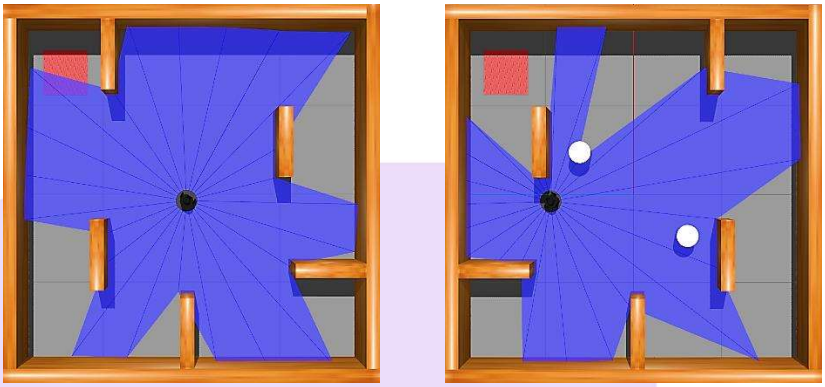
4. Çerçeve

Amacımız, robotu bulunduğu konumdan hedef noktaya minimum çabayla yönlendirmek için en uygun stratejiyi öğrenen DQN ajanı elde etmektir. Pekiştirmeli Öğrenmede durum uzayı, eylem uzayı ve ödül mekanizması en temel özelliklerdir. Doğru kurulan ödül mekanizması ile ajan kendi kendine öğrenecektir. Şekil 5'de Ajanın çalışma süreci model ile gösterilmiştir. Bu süreç aşağıdaki temel adımlar ile özetlenebilir.

- Ajan, eylemi DQN modelinden aldığı Q-değerine göre seçer ve ortama yansıtır.
- Ajan bir ortamda elde edilen ödülü ve diğer durum verilerini tekrar oynatma hafızasında (Experiance Replay) saklar.
- Ajan, saklanan verileri, tekrar hafızasından rasgele örnekleyerek modeli günceller.
- Ajan'ın amacı, TurtleBot3'ü engellerden kaçınarak sabit ya da rastgele değişen hedef noktalara ulaştırmaktır.

Şekil 5. DQN ajanının çalışması [2]

TurtleBot3 Burger modeli için Şekil 6'da görülen ortamlar oluşturulmuştur. Şekil 6 (sağ)'da ortama iki adet hareketli engel eklenerek problem zorlaştırılmıştır.



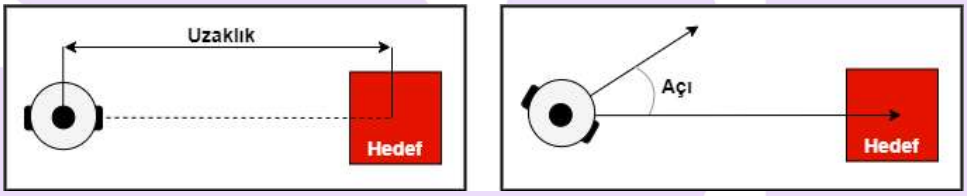
Şekil 6. Çalışma için oluşturulan sabit engelli (sol) ve hem sabit hem de hareketli engelli (sağ) benzetim ortamları

4.1. Durum Uzayı

Durum (State), çevrenin bir gözlemidir ve mevcut durumu açıklar. Bu, pekiştirmeli öğrenme ajanı için hayati önem taşır, çünkü durum verisine göre model ile Q değeri hesaplanır ve robot hareket eylemi seçilir. Modelimizde durum 28 veriden oluşan boyuta sahiptir. Bu verilerin 24 tanesi Lazer Mesafe Sensörü (LDS) değeridir. Diğerleri ise hedefe olan açı, hedefle arasında ki mesafe, engel ile arasındaki mesafe ve engel ile arasındaki açıdır. Bu verilerin matematiksel bir yaklaşım ile ifadesi aşağıdaki gibidir [14]:

$$\text{Durum (28)} = \text{LDS (24)} + \text{Hedefe Açı (1)} + \text{Hedefe Mesafe (1)} + \text{Engele Mesafe (1)} + \text{Engele Açı} \quad (4)$$

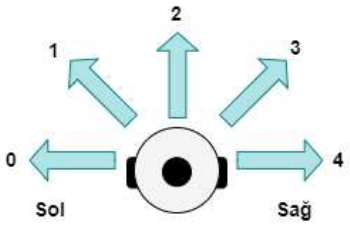
Engele olan mesafe, LDS den alınan verilere göre en yakın engel ile olan mesafedir. Engele olan açı ise en yakın engeli gören LDS'yi temsil eden değerdir. Şekil 7'de hedefe olan mesafe ve robot ile hedef arası açı resmedilmiştir.



Şekil 7. Robot ile hedef arasındaki mesafe (sol), robot ile hedef arasındaki açı (sağ) [2]

4.2. Eylem Uzayı

Robotun, durum türüne bağlı olarak hareket edebilmesini temin için beş eylemi (Action) vardır. Bu eylemler Şekil 8'de gösterilmiştir. Beş farklı eylemde sırasıyla açısal hızlar -1.5, -0.75, 0, 0.75, 1.5 rad/sn'dir. Bu çalışmada robotun sabit doğrusal hızı 0.15m/s'dir ve açısal hızı model tarafından belirlenir. DQN modelinin maksimum Q değeri eylem seçimini belirler.



Şekil 8. Eylem uzayı [2]

4.3. Ödül Sistemi

Ödül tasarımı öğrenme için çok önemlidir. Robot bir eylem gerçekleştirdiğinde ödül (Reward) alır. Ödül olumlu ya da olumsuz olabilir. Robot hedefi bulunca, 13 farklı hedef noktadan biri rastgele seçilerek yeni hedef nokta olur. Turtlebot3 hedefe ulaştığında, büyük bir pozitif ödül alır, engelle çarpıştığında ise büyük bir negatif ödül alır. 14 cm altında ki LIDAR verisi çarpışma olarak değerlendirilir. Ayrıca TurtleBot3 hedefe yaklaştığında pozitif bir ödül alır ve uzaklaştığında negatif bir ödül alır. Bölüm, TurtleBot3 bir engele çarptığında veya belirlenen adım sayısından sonra sona erer.

5. Uygulama

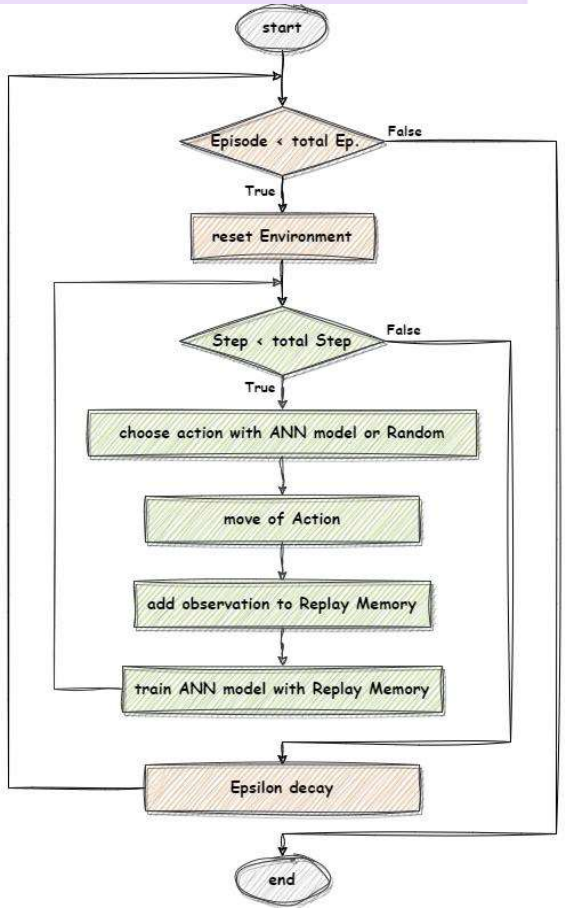
Benzetim senaryoları için, Şekil 6'da iki tür ortam oluşturulmuş ve bu ortamlarda pekiştirmeli eğitim yapılmıştır.

Öğrenme oranı (α): 0 olarak ayarlamak, robotun öğrenmeyeceği ve 0,9 gibi yüksek bir değer, öğrenmenin hızlı bir şekilde gerçekleşebileceği anlamına gelir. Ancak çok hızlı öğrenme-de optimum noktayı bulamama riski de mevcuttur.

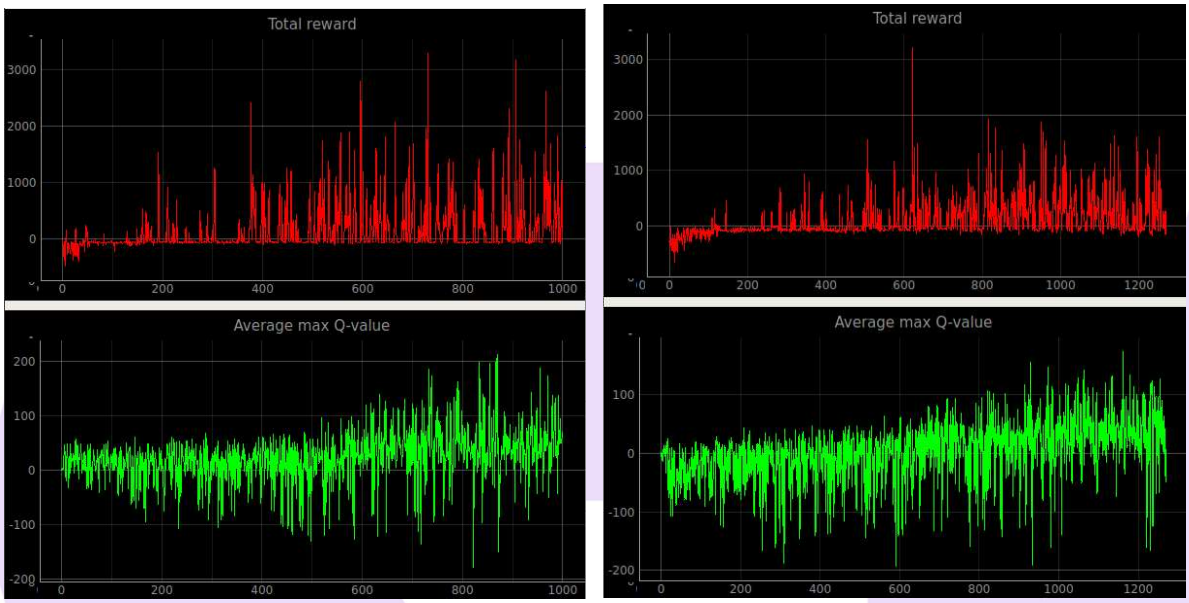
İndirim faktörü (γ): 0 değeri ajanın yalnızca mevcut ödülleri dikkate almasını sağlarken, 1'e yaklaşan bir faktör, ajanın uzun vadeli yüksek bir ödül için çabalamasına neden olacaktır.

Keşif sabiti (ε): 0,99 gibi yüksek bir değer ayarlamak, eylemlerin %99'unu stokastik yapacaktır. Keşif aşaması için yüksek bir değerle başlar, bölümler ilerledikçe epsilon azaltma tekniği ile 0'a yaklaştırırız. Böylece giderek deneyimlerden daha fazla faydalanırız.

Seçilen başlangıç parametreleri, $\alpha = 0.001$, $\gamma = 0.95$ ve $\varepsilon = 1$'dir. Bu durumda minimum ε değerimiz olan 0,01'e ulaşana kadar her bölümde ε * 0,995 olan azaltma tekniği kullanılmıştır.



Şekil 9. Uygulamada kullanılan DQN akış şeması



Şekil 10. Birinci ortamda (GazeboStage1)(sol sütun) ve ikinci ortamda (GazeboStage2) eğitilen DQN Modelinin eğitim sürecindeki toplam (üst satır) ve ortalama (alt satır) ödül değişimi

Şekil 10'daki grafiklerde DQN kullanılarak GazeboStage1 ve GazeboStage2 ortamının izlenmesinden elde edilen kümülatif ödül kırmızı ile, model tarafından tahmin edilen Q değeri yeşil ile gösterilmektedir.

GazeboStage1'de alınan kümülatif ödülün daha yüksek olduğu grafiklerde görülmektedir. GazeboStage2'de hareketli engellerin problemi biraz daha zorlaştırdığı kümülatif ödüllerden anlaşılmaktadır. Oluşturulan ortamlarda DQN modeli ile yapılan eğitimler sonucunda robotun hedef noktaları başarılı bir şekilde bulduğu gözlemlenmiştir.

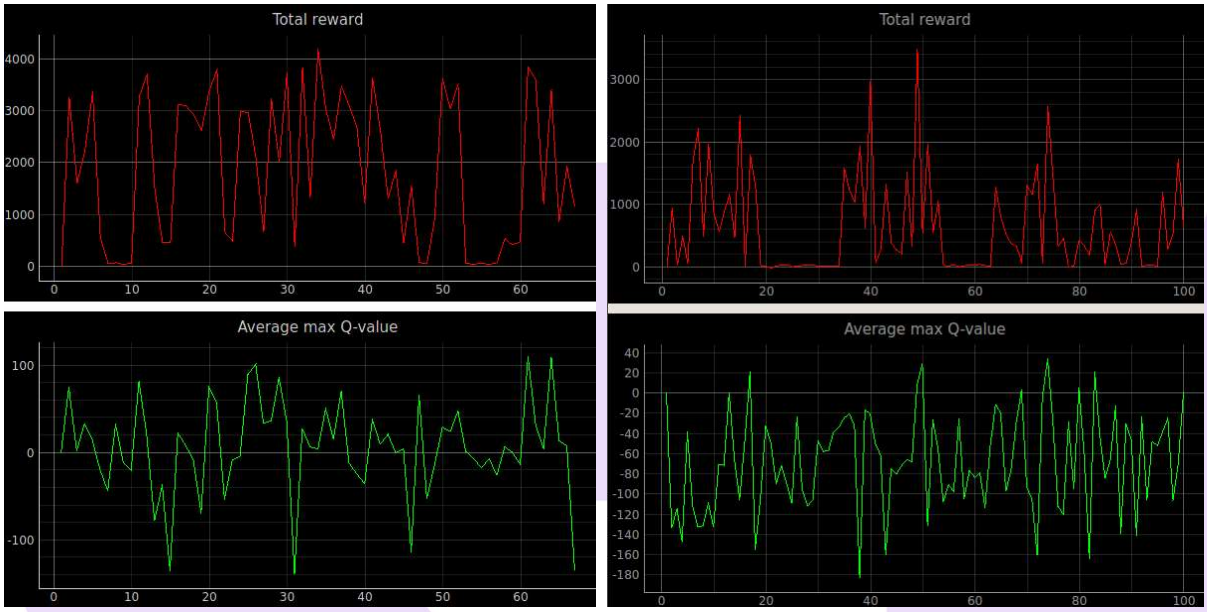
6. Sonuçlar

Bu çalışmada, benzetim ortamında Derin Pekiştirmeli Öğrenme ile robotlarda otonom hareket ele alınmıştır. Pekiştirmeli öğrenme yöntemi birçok deneme yanılma araştırması gerektirir. Öğrenme aşamasını doğrudan gerçek dünya-da gerçekleştirmek çok zordur, hatta imkânsızdır. Bu nedenle iyi bir model eğitmek ve eğitilen modeli gerçek cihaza taşımak için benzetim ortamına ihtiyaç vardır.

Benzetim çalışmasında, Derin Pekiştirmeli Öğrenme yöntemi ile eğitim yapmak ve doğrulamak için ROS'un Gazebo simülatörü ile TurtleBot3 burger robot modeli kullanılmıştır. Süreç boyunca TurtleBot3'ün kilometre sayacı ve imu sensörü tarafından toplanan pozisyon ve mevcut konum bilgisi ile LIDAR sensörü tarafından taranan benzetim ortamı bilgileri kullanılarak DQN ajanı çevre ile etkileşime girer.

Gazebo simülatöründe oluşturulan iki farklı ortamda Derin Pekiştirmeli Öğrenme yöntemi ile robot modeli eğitilmiştir. Robot, ortamda engellere çarpmadan rastgele yer değiştiren hedef noktaları buldukça ödülleri toplar. Amaç toplanan ödülü maksimize etmektir. Eğitilen robotumuz test ortamında, bir bölümde engellere çarpmadan arka arkaya 10 dan fazla hedef noktayı bulmayı başarmıştır. Derin Pekiştirmeli Öğrenme ile belirlediğimiz iki ortam için de mobil robot hedef noktaları bularak başarılı sonuçlar almıştır. Eğitim sonrası yapılan iki ortamdaki birer testin sonucu Şekil 11'de verilmiştir. Bu çalışmanın kodlarına "https://github.com/computeachp/Mobile-Robots-Navigation-with-Deep-Reinforcement-Learning" bağından erişilebilir.

Bu çalışmada benzetim ortamında eğitilen modeli gerçek dünyada kullanımı için çeşitli ayarlamalar yapmak gerekecektir. Ayrıca hedef noktaları belli olmayan, hedefin de ajan tarafından bulunduğu bir senaryo gerçek dünya problemlerine daha uygundur. Bir sonra ki araştırma konusu bu olabilir.



Şekil 11. Birinci ortamda (GazeboStage1)(sol sütun) ve ikinci ortamda (GazeboStage2) DQN Modelinin test sürecindeki toplam (üst satır) ve ortalama (alt satır) ödül değişimi

7. Referanslar

[1] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero, "Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo," *arXiv preprint arXiv:1608.05742*, 2016.
[2] L. Garrido Alvarez, "DQN: Deep Q-Learning for Autonomous Navigation," 2019.
[3] M. Quigley *et al.*, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009, vol. 3, no. 3.2, p. 5.
[4] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, 2004, vol. 3, pp. 2149–2154.
[5] R. Amsters and P. Slaets, "Turtlebot 3 as a robotics education platform," in *Robotics in Education: Current Research and Innovations 10*, 2020, pp. 170–181.
[6] TurtleBot3 Burger Specifications, https://emanual.robotis.com/docs/en/platform/turtlebot3/features/, (accessed Dec. 20, 2022).
[7] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.
[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[9] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
[10] I. B. de A. Santos and R. A. Romero, "A deep reinforcement learning approach with visual semantic navigation with memory for mobile robots in indoor home context," *Journal of Intelligent & Robotic Systems*, vol. 104, no. 3, p. 40, 2022.
[11] M. G. Kabataş and S. İ. Omurca, "Setting Reward Function of Sensor Based DDQN Model," *Avrupa Bilim ve Teknoloji Dergisi*, no. 28, pp. 539–544, 2021.
[12] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[13] J. Tsai, C.-C. Chang, Y.-C. Ou, B.-H. Sieh, and Y.-M. Ooi, "Autonomous driving control based on the perception of a lidar sensor and odometer," *Applied Sciences*, vol. 12, no. 15, p. 7775, 2022.
[14] TurtleBot3 Machine Learning, https://github.com/ROBOTIS-GIT/turtlebot3_machine_learning, (accessed Dec. 21, 2022).