MAKİNE ÖĞRENMESİ İLE KAÇINMA HAREKETLERİNİN ÜRETİLMESİ

GENERATING AVOIDANCE MOTIONS WITH MACHINE LEARNING

İSMAİL ABDULLAH ÖZOĞUR

Dr. Öğr. Üyesi ZÜMRA KAVAFOĞLU

Tez Danışmanı

Hacettepe Üniversitesi

Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin

Bilgisayar Grafiği Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS TEZİ

Olarak Hazırlanmıştır

2019

ÖZET

MAKİNE ÖĞRENMESİ İLE KAÇINMA HAREKETLERİNİN ÜRETİLMESİ

İsmail Abdullah Özoğur
Yüksek Lisans, Bilgisayar Grafiği Anabili Dalı
Tez Danışmanı: Dr. Öğr. Üyesi Zümra Kavafoğlu
Eylül 2019, 42 sayfa

Kaçınma hareketleri özellikle video oyunları olmak üzere, sanal insan modellerinin kullanıldığı bilgisayar animasyonu uygulamalarında önemli yer tutmaktadır. Diğer hareketlerde olduğu gibi gerçekçi kaçınma hareketleri elde etmek için genellikle hareket yakalama teknikleri kullanılmaktadır. Fakat gerçek hayatta bir insanın yapabileceği sonsuz miktarda hareket vardır ve bunların hepsinin hareket yakalama teknikleri ile elde edilmesi imkansızdır. Yeteri kadar hareket yakalama verisi elde edilmiş olsa bile çok miktarda hareket içinden uygun olanın seçilmesi kolay değildir.

Çok miktarda hareket yakalama verisi içinden uygun olanını gerçek zamanlı olarak seçmek için yapılan bazı çalışmalar vardır fakat bunlar genelde çok masraflı işlemler içermektedir ve çok miktarda sistem kaynağı kullanmaktadır. Bu sebeple, bu tip yöntemlerin çok sayıda karakter içeren oyunlar için kullanılması zordur.

Yukarıda bahsedilen nedenlerden dolayı, birçok oyunda gelen saldırılardan kaçınmak için büyük sıçramalar, maksimum miktarda eğilme gibi abartılı hareketler birden fazla saldırıya karşı kullanılmaktadır. Fakat sanal insan modelinin küçük bir sıyrılma ile kurtulabileceği saldırılardan yüksek efor gerektiren hareketler ile kaçınması ve sürekli aynı hareketlerin kullanılması gerçekçiliği azaltmaktadır.

Bu çalışmayla hiçbir hareket yakalama verisi kullanılmadan kaçınma hareketlerinin sanal insan modeline derin pekiştirmeli öğrenme ile sıfırdan öğretilmesi amaçlanmaktadır. Öğrenme

algoritmasının bileşenleri karakterin hareket için fazla efor harcamaması ve fiziksel kısıtlara uygun davranması gibi kriterler göz önüne alınarak tasarlanmıştır. Böylece öğrenilen hareketlerin gerçekçi olması amaçlanmıştır. Çevrimdışı çalışan öğrenme aşamasında sanal insan modelinin vücudunun rastgele noktalarına yine rastgele bölgelerden küreler fırlatılmış ve modelin yaptığı hareketlerin küreden kaçınma bakımından başarısına göre ödüller verilmiştir. Çevrimdışı aşamada öğrenilen politika kullanılarak gerçek zamanlı kaçınma hareketleri üretilmiştir. Bu hareketlerin başarımını ölçmek için kullanıcı testleri yapılmış ve testlerin sonuçları yorumlanmıştır.

Anahtar Kelimeler: Kaçınma Hareketi, Derin Pekiştirmeli Öğrenme, Sanal İnsan Animasyonu.

ABSTRACT

GENERATING AVOIDANCE MOTIONS WITH MACHINE LEARNING

İsmail Abdullah Özoğur

Master of Science, Department of Computer Graphics

Supervisor: Asst. Prof. Dr. Zümra Kavafoğlu

September 2019, 42 pages

Avoidance motion plays an important role in computer animation applications which have virtual human models, especially in video games. Motion capture techniques are often used to achieve realistic avoidance motions. However, there is an unlimited number of motion that a person can perform in real life, which are obviously impossible to be achieved with motion capture techniques. Even if adequate amount of motion capture data is available to emulate human motion, it won't be easy to find the appropriate motion from a large amount of data.

There are studies to determine the most appropriate motion within a large pool of motion capture data in real time, but these studies often involve costly processes and use a large amount of system resources. Therefore, such methods are difficult to use for games with a large number of characters.

Due to the reasons mentioned above, unrealistic motion such as a big jump and an extreme bending is used against multiple attacks in many games. However, the fact that the virtual human model avoids the attacks in an unrealistic way, which could in fact be easily warded off, and continuously employs the same motion lead the virtual human model to lose touch with reality.

This study seeks to teach avoidance motion to virtual human model through deep reinforcement learning without using any motion capture data. The components of the learning algorithm in this study are designed in a way that the virtual model won't spend an unrealistic

effort and will act according to physical constraints. Therefore, this study aims to teach real-life motion to the virtual human model. In the offline learning phase, spheres were thrown from random areas to the random points of the virtual human model. The rewards of reinforcement leraning algorithm are set based on the virtual human model's success on guarding off the spheres.

Accordingly, real-time avoidance motion was generated through this learned policy at the offline learning phase. In order to measure the virtual human model's performance of avoidance motion, user tests were conducted and their results were interpreted.

Keywords: avoidance, deep reinforcement learning, Virtual Human Animation.

TEŞEKKÜR

Lisansüstü eğitimim boyunca engin bilgi ve tecrübelerinden yararlandığım, sadece bilimsel anlamda değil sahip olduğu eşsiz bilgisiyle hayatıma yön veren, desteğini benden esirgemeyerek her zaman yanımda olduğunu hissettiren değerli hocam Sayın Dr. Öğr. Üyesi ZÜMRA KAVAFOĞLU'na,

Hayatım boyunca her koşulda bana destek veren ve sabır gösteren, önceliklerini her zaman benim önceliklerime göre değiştiren ve bunun karşılığını hiçbir zaman tam olarak ödeyemeyeceğim, bugünlere gelmemde en büyük katkıları olan, bu hayatta hiç bir şeye asla değişmeyeceğim canım aileme, Sonsuz Teşekkürler...

İÇİNDEKİLER

ŞEKİL LİSTESİ

1 GİRİŞ

1.1 Motivasyon ve Çalışmanın Kapsamı

Sanal insan modelleri bilgisayar animasyonu uygulamalarında sıkça kullanılmaktadır ve uygulamanın kalitesi için sanal insan modellerinin gerçekçiliği önemli bir ölçüttür. Bir sanal insan modelinin gerçekçi görünmesi için gerçek insanların hareketlerini başarılı bir şekilde taklit etmesi gerekir. Başarılı taklitler elde etmek için genelde hareket yakalama verileri kullanılmaktadır.

Günümüzde hareket yakalama verisi pahalı sistemlerden mobil cihazlara çok çeşitli gereçlerle elde edilebilmektedir. Gerekli gereçler ve uygulamalar bir şekilde sağlanmış olsa bile hareketin yakalanması, işlenmesi ve uygun koşullar için uygun hareket yakalama verilerinin seçilmesi oldukça zahmetli işlerdir. Modellerin düz bir yolda yürümesi ya da koşması gibi basit hareketler elde etmek kolaydır ve çok fazla hareket yakalama verisi gerektirmez. Fakat yapılacak hareketlerin çeşitli değişkenlere bağlı olduğu dinamik hareketler üretmek o kadar kolay değildir. Çünkü bu değişkenlerin çeşitli kombinasyonlarına göre farklı hareketler üretmek gerekmektedir. Bu ise çok fazla hareket yakalama verisine ihtiyaç olacağı anlamına gelir.

Yukarıda sayılan sebeplerden dolayı literatürde hareket üretme ile ilgili bir çok çalışma yer almaktadır. Bunların arasında hareket yakalama verilerini parçalayıp birleştirerek yeni hareketleri çalışma zamanında üreten sistemler vardır. Ayrıca makine öğrenme yöntemlerinin gelişmesi ve günümüz bilgisayar performanslarının artması ile makine öğrenme yöntemlerinden faydalanan hareket üretme çalışmaları da oldukça yaygınlaşmıştır. Bu çalışmaların birçoğu hareket yakalama verilerini taklit ederek benzerlerini üretmek üzerinedir. Bununla beraber sıfırdan ortam durumuna göre uygun hareket üretmeyi öğrenen sistemler de geliştirilmiştir.

Kaçınma hareketleri üretmek de yapılması gereken hareketin kaçınılacak saldırıya göre şekillenmesi gerektiği için literatürdeki zor konulardandır. Örneğin bir dövüş oyunundaki sanal insan modelinin rakibinin yumruğundan kurtulmak için yapması gereken hareket yumruğun hızı, hacmi, hedef aldığı vücut bölgesi gibi birçok değişkene bağlıdır. Elbette kısıtlı miktarda hareket yakalama verisi kullanılarak, gelen bütün darbeler için büyük adımlar, sıçrayışlar ve

yere yatma gibi hareketler yaparak da saldırılardan kaçınılabilir. Fakat ufak bir sıyrılma hareketiyle az bir efor sarf edilerek kurtulmanın mümkün olduğu bir durumda böyle abartı hareketler yapmak gerçekçiliği bozmaktadır. Ayrıca sürekli aynı kaçınma hareketlerinin yapılması tekdüze bir görünüme sebep olmaktadır.

Bu çalışmanın amacı gerçekçi kaçınma hareketlerini bir sanal insan modeline hareket yakalama verisi olmadan sıfırdan öğretmektir. Geliştirilen sistem çevrimdışı öğrenme sistemi ve çevrimiçi hareket üretme sistemi olmak üzere iki parçadan oluşmaktadır.

Çevrimdışı öğrenme sisteminde, sanal insan modeli pekiştirmeli öğrenme ile vücudunun rastgele bölgelerine, rastgele konumlardan fırlatılan kürelerden gerçekçi bir şekilde kaçınmayı öğrenmektedir. Bir kaçınma hareketinin gerçekçi olması için aşağıdaki kriterler belirlenmiştir.

- Sanal insan modeli, eklemlerin haddinden fazla dönmesi, ya da aşırı hızlı hareket etmesi gibi gerçek bir insan vücudunun yapısına aykırı hareketler yapmamalıdır.
- Hareketler fizik kurallarına uygun olmalıdır. Model dengede duramayacağı bir hareketi tercih etmemelidir.
- En az efor gerektiren hareketler seçilmelidir. Model gereğinden fazla hareket etmemelidir.
- Farklı koşullar için farklı hareketler üretilmelidir. Bir çok koşula uyan bir kaçınma hareketi tekrar tekrar kullanılmamalıdır.

Sistemde Unity makine öğrenme ajanları araç setinin (Unity ML-Agents Toolkit) sağladığı pekiştirmeli öğrenme yöntemi kullanılmaktadır.

Çevrimiçi hareket üretme sisteminde, çevrimdışı öğrenme sisteminde öğrenilmiş olan politikanın kaydedildiği öğrenme dosyası kullanılır. Bu sistemde küreler, sanal insan modelinin kullanıcı tarafından işaretlenen vücut kısmına fırlatılmaktadır.

1.2 Tez İçeriği

Bu bölümden sonra tezin üç bölümü daha vardır. İkinci bölümde pekiştirmeli öğrenme ve karakter animasyonu ile ilgili genel bilgiler verilmektidir Üçüncü bölümde çalışmada yapılanlardan detaylı bir şekilde bahsedilmektedir. Son olarak ise sonuç bölümü gelmektedir.

İkinci bölümün ilk kısmı pekiştirmeli öğrenmenin detaylı anlatımı ve kullanım alanlarından örneklerle başlamaktadır. Sonrasında politika gradyan metotlarından (Policy Gradient Methods) bahsedilmekte ve çalışmada kullanılan politika gradyan metodu olan proksimal politika optimizasyonu (PPO - Proximal Policy Optimization) yöntemi açıklanmaktadır. Son olarak ise Unity ML-Agents eklentisinin açık bir şekilde tanıtımı gelmektedir. İkinci kısımda ise kaçınma hareketinin tanımı sonrasında dinamik kaçınma hareketleri üretmek için yapılmış bazı çalışmalardan ve pekiştirmeli öğrenme ile hareket üretme çalışmalarından bahsedilmektedir.

Üçüncü bölüm çalışmamızın içeriğinin detaylı bir şekilde anlatıldığı bölümdür. Pekiştirmeli öğrenme ortamından, ödül, gözlem ve hareketlerden bahsedilmektedir. Çalışmanın test sonuçları da bu bölümde yer almaktadır.

Son olarak sonuç bölümünde ise genel olarak çalışmanın başarısından ve kısıtlarından bahsedilmektedir. Ayrıca sistemi geliştirmek için gelecekte yapılabilecek çalışmalar da bu bölümde anlatılmaktadır.

2 GENEL BİLGİLER

2.1 Pekiştirmeli Öğrenme

2.1.1 Temel Bilgiler

Makine öğrenmesi, bir sistemin açıkça programlanmadığı bir işi tıpkı insanlarda olduğu gibi öğrenerek gerçekleştirmesini amaçlamaktadır. Yapılması gereken işin belli bir formülü hiç var olmadığında ya da karmaşık ve masraflı olduğu durumlarda tercih edilmektedir. Yüz tanıma, metin çeviri, bilgisayar oyunları vb. alanlar kullanım alanlarına örnek olarak verilebilir.

Şekil 2.1: Temel makine öğrenmesi çeşitleri

Makine öğrenmesi yöntemleri aşağıdaki gibi sınıflandırılabilir (bkz. Şekil 2.1).

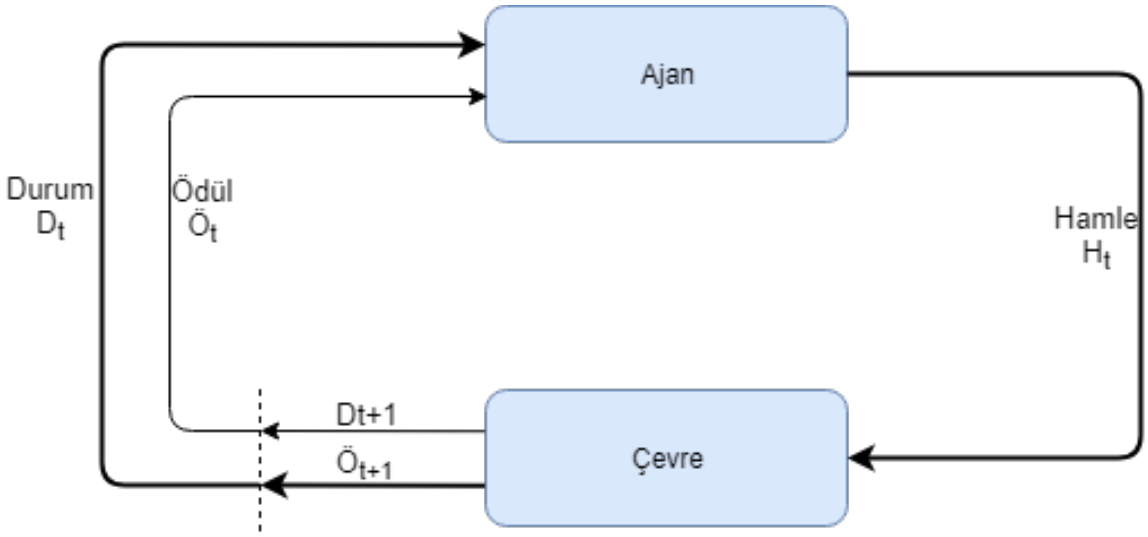
Gözetimli Öğrenme (Supervised): Daha önceden hazırlanan işaretli ve sınfılandırılmış verilerden öğrenilen bilginin yeni verilere uygulanması işlemidir. Eldeki girdi ve çıktılar arasında bir fonksiyon oluşturur ve bu fonksiyonu bilinmeyen yeni girdilerin çıktısını hesaplamak için kullanır.

Gözetimsiz Öğrenme (Unsupervised): İşaretlenmemiş ve sınıflandırılmamış verilerdeki bilinmeyen bağıntıyı bulan bir fonksiyon üretmeye çalışır.

Pekiştirmeli Öğrenme (Reinforcement): Gözetimli ve gözetimsiz makine öğrenmesinden farklı olan **pekiştirmeli öğrenme**de başlangıçta elimizde etiketli ya da etiketsiz veriler yoktur, yalnızca çevre etkileşimi ile bir hedefe ulaşmaya çalışılır. Ajan (agent) adı verilen öğrenen yapı çevre etkileşimi ile uzun vadede alacağı ödül toplamını maksimize etmeye çalışır

(bkz. Şekil 2.2). Bunu yapabilmek için ajan çevresindeki ilgili durumları gözlemleyebilmeli ve çeşitli davranışlarla bu durumları değiştirebilmelidir. Davranışları ve çevre durumuna göre aldığı ödülleri değerlendiren ajan en yüksek ödül miktarı ile hedefe ulaşmaya çalışır. İnsan ve hayvanların öğrenmesine benzer, tamamen deneme-yanılmaya dayalı bir yöntemdir.

Deneme yanılmaya dayalı olması pekiştirmeli öğrenmede diğer öğrenme yöntemlerinde olmayan bir ikileme yol açmaktadır. Ajan aldığı toplam ödülü yükseltmek istediği için yüksek ödüller kazandığı durum ve davranışları tekrarlama (exploitation) eğiliminde olursa düşük ödüllerden korunmuş olurken daha yüksek ödülleri keşfetme (exploration) şansını da kaçırmış olacaktır. Fakat sürekli daha yüksek ödüller ararsa da elinde olandan daha düşük bir ödülle bitirme ihtimali ortaya çıkar. Yıllardır çözülmemiş bir problem olan bu dengenin sağlanması sistemin başarısını yakından etkilemektedir.



Şekil 2.2: Pekiştirmeli öğrenme ajan-çevre etkileşimi.

Pekiştirmeli öğrenmenin temel öğeleri şunlardır:

Politika (Policy): Ajanın içinde bulunduğu durumları değerlendirip uygun davranışları gerçekleştirmesi olarak tanımlanabilir. Durum bilgilerine karşılık davranış bilgilerinin denk geldiği bir referans tablosu olabileceği gibi ciddi ve yoğun hesapların yapıldığı politikalar da söz konusu olabilir.

Ödül (Reward): Ortamın her adımda ajana verdiği sayısal değerdir. Ajanın tek amacı bu değerlerin toplamı(veya ağırlıklı toplamı) en büyük olacak şekilde turu tamamlamaktır. Bunu sağlamak için ajan aldığı puanlara göre politikasını güncellemektedir.

Durum Değeri (Value): Ödül bir davranışın belirli bir andaki başarısını belirlerken durum değeri uzun vadedeki değerlendirmedir. Gelecekteki ödül toplamının tahmini denilebilir.

Bir durumda alınan ödül ile o durumun değeri arasında birebir olmayan bir ilişki vardır. Bir durumun ödülü düşük olmasına rağmen sonraki durumlarda daha yüksek ödüllere sebep olacaksa değeri yüksek olur ve ajan aynı politika ile devam edebilir. Aksine, bir davranışın bir durumdaki ödülü yüksek olabilir fakat o durumdan bu davranışla ilerleyerek sonraki durumlarda yüksek ödüller almak mümkün olmuyorsa ajan politikasını değiştirmelidir. Burada duruma verilecek değerler çeşitli hesaplarla sonraki durumların değerinin tahminlerine bağlıdır ve bu tahminlerin başarısı pekiştirmeli öğrenme algoritmasının başarısını göstermektedir.

Çevre Modeli (Model): Pekiştirmeli öğrenmenin son öğesi ise bazı pekiştirmeli öğrenme sistemlerinde kullanılan, çevre koşullarını ve davranışlarını taklit eden bir çevre simülasyonudur.

Sıkça kullanılan bir diğer yapay zeka yöntemi de **yapay sinir ağları**dır. Bu yöntem sinir hücrelerinin yapılarından hareketle, beynin çalışma şeklini taklit eder. Eldeki girdi ve çıktıları analiz ederek geliştirilen algoritmalar ile çıktısı belli olmayan girdilerin sonuçlarını bulmayı amaçlar. Pekiştirmeli öğrenmede çok fazla durum sinyali ve davranış ihtimali olduğunda bunları bir tabloda tutmak çok büyük miktarda belleğe ihtiyaç doğurduğu için yapay sinir ağları ile birleştirilerek **derin pekiştirmeli öğrenme** algoritmaları bulunmuştur. Günümüzde yapılan çalışmaların çoğunda pekiştirmeli öğrenme algoritmaları derin öğrenme algoritmaları ile birlikte kullanılmaktadır.
Çok çeşitli problemlere uygulanması mümkün olan pekiştirmeli öğrenmenin kullanımı her geçen gün artmaktadır. Aşağıda çeşitli alanlardan güncel bazı örnekler verilmiştir.

Bilgisayar Sistemlerinde Kaynak Yönetimi: Ağ yönetimi ve bilgisayar kümelerinde kaynakların etkin kullanımı maliyet ve hız açısından oldukça önemli ve zorlayıcı bir problemdir. Çevrimiçi karar algoritmaları iş yükü ve ortamın durumuna bağlıdır. Bu algoritmalar genelde insan tecrübe ve gözlemlerine dayalı tahminlerle planlanmaktadır. Mao v.d. [1] pekiştirmeli öğrenme yardımı ile kaynakların beklemedeki görevlere, işlemedeki minimum yavaşlama ile uygun şekilde nasıl dağıtılabileceğini göstermiştir.

Trafik Lambası Kontrolü: Arel v.d. [2] çoklu ajanlı pekiştirmeli öğrenme algoritmalarını kullanarak kurulan sistemle trafik yoğunluğunu azaltma amaçlı bir çalışma gerçekleştirdi. Bu

çalışma gerçek ortamda denenmedi fakat simülasyon ortamında yapılan testler sonucunda günümüzde kullanılan yöntemlerden çok daha etkin olduğu kanıtlandı.

Robotik: Robotlar günümüz endüstrisinde büyük yer tutmaktadır. Fakat bunların büyük bir kısmı sadece belirli işlevler için geliştirilmiş ve önceden programlanmış kabiliyetleri dışında bir işte kullanılamayacak robotlardır. Farklı yetenekler geliştirebilen ve bulunduğu ortama uyum sağlayabilen robotlar üretmek için çok miktarda makine öğrenmesi çalışması yapılmaktadır . Bu çalışmaların büyük bir kısmında ise pekiştirmeli öğrenme kullanılmaktadır ve umut veren sonuçlar ortaya çıkmaktadır. Long vd. [3] robotların özel sensörlerden gelen gözlem verileri yardımıyla, kalabalık ortamlarda yayaların hız, yön ve uzaklık gibi özelliklerini algılayarak gerektiğinde yön değiştirmeyi öğrenmelerini sağlamışlardır. Çalışmanın sonucunda robotlar yayalara takılmadan düzgün bir yol çizerek ilerlemeyi başarmıştır. Winfried vd. [4] kolların sürekli davranış uzayındaki hareketlerinde karşılaşılan problemleri ele alarak bunları derin pekiştirmeli öğrenme yöntemi ile çözmeye çalışan bir çalışma yapmışlardır.

Web Sistemi Parametrelerinin Yapılandırılması: İnternet trafiğinin değişkenliği ve sistemin çalıştığı dinamik sanal makine ortamlarının durumuna göre web sistemlerinin parametrelerinin tekrar yapılandırılması kritik önem taşımaktadır. Çok katmanlı mimarideki sistemlerde bu işlem parametre sayısı çoğaldığı için daha zorlayıcı hale gelmektedir. Bu v.d. [5] pekiştirmeli öğrenmenin de dahil olduğu bir yaklaşımla bu işlemi otomatik hale getirmiştir.

Kişiselleştirilmiş Haber Önerileri: Site önerileri için genelde kullanıcıların oylama verileri kullanılır. Fakat genelde kullanıcılar incelediği her içeriği oylamazlar. Bu yüzden kullanıcının tıkladığı içerik baz alınarak kullanıcının ilgilenebileceği haberleri oylama yapmasını gerektirmeden kestirmeye çalışan sistemler vardır. Zheng v.d. [6] kullanıcının sadece haber bağlatılarına tıklamalarını değil tıklamamalarını da gözlem verilerine dahil etmişlerdir. Kullanıcının daha önce okumuş olduğu haberlerle benzerlik oranını da hesaba katarak yaptıkları pekiştirmeli öğrenme çalışması ile başarılı sonuçlar elde edilmiştir.

Reklam: Alibaba Group araştırmacıları Jin v.d. [7], satıcıların belli teklifler vererek reklamlarını müşterilerine gösterebilecekleri bir çalışma yapmışlardır. Çalışmada çoklu ajan içeren pekiştirmeli öğrenme algoritmaları kullanılmıştır. Umut veren sonuçları dolayısıyla TaoBao platformunda canlı teste sunacakları duyurulmuştur.

Oyunlar: Oyun sektörü yapay zeka uygulamalarının kullanımında şüphesiz ilk sıralardadır. Çeşitli yapay zeka teknolojileri ile geliştirilen sanal oyuncular artık insanları yenebilmektedir. Gözetimli öğrenme yardımıyla gerçek oyuncuların verileri kullanılarak yapılan çalışmaların yanı sıra pekiştirmeli öğrenme ile yapılan çalışmalar da günümüzde oldukça yaygınlaşmıştır. David vd. [8] tarafından geliştirilen AlphaGo projesinde yapay sinir ağları ilk önce uzman oyuncuların oynamış olduğu oyunların verileriyle gözetimli öğrenme yöntemi kullanılarak eğitilmiştir. Daha sonra sistem kendisine karşı oynayarak pekiştirmeli öğrenme ile daha da geliştirilmiştir. AlphaGo, diğer yapay zeka programlarına karşı oynadığı oyunların yüzde 99.8'ini kazanmıştır ve Avrupa go şampiyonunu 5-0 yenerek bir ilki başarmıştır. Araştırmacılar daha sonra hiç insan bilgisi kullanmadan, oyunu pekiştirmeli öğrenme ile sıfırdan öğrenen AlphaGo Zero'yu [9] geliştirmiştir. AlphaGo Zero ise AlphaGo'yu büyük bir farkla yenmeyi başarmıştır. Yine pekiştirmeli öğrenme kullanılarak geliştirilmiş olan OpenAI Five [10] isimli Dota 2 takımı Dota 2 dünya şampiyonlarını yenmiştir.

2.1.2 Proksimal Politika Optimizasyonu (Proximal policy optimization)

Diğer pekiştirmeli öğrenme yöntemlerinden farklı olarak politika gradyan metotları (policy gradient methods) daha önce öğrendiği davranışların değerlerinden yola çıkarak geliştirdiği algoritmalarla değer tahmini yaparak davranış seçmez. Yani politikaları davranış-değer tahminine dayalı değildir. Bunun yerine doğrudan, hiçbir değer fonksiyonuna başvurmadan bir davranış dönen parametrik politikaları öğrenmeye çalışır. Bu politikaların parametrelerini yine değer fonksiyonları yardımıyla bulabilir fakat davranış seçimi için değer fonksiyonlarına ihtiyaç yoktur [11].

Diğer yöntemlere göre birçok avantajı vardır:

- Politika gradyan metotları daha yakınsaktır. Politikadaki bir değişikliğe göre seçilen davranıştaki değişim daha yumuşak olur. Sonucun bir yerel optimum değere yakınsaması garantidir, hatta en iyi durumda genel optimum değere de yakınsayabilir. Değer fonksiyonu temelli yöntemlerde ise değişiklikler daha keskindir, değer fonksiyonundaki ufak bir değişiklik seçilen davranışta çok büyük değişikliğe sebep olabilir ve bu da sonucun belli bir değere yakınsamasına engel olur.

- Daha karmaşık ve mümkün olan davranış uzayının sürekli olduğu çalışmalarda, durumları davranışlarla doğrudan eşleyecek fonksiyonlar bulmak hem zordur hem de

çok fazla kaynak ve zaman tüketir. Politika gradyan metotları kullanılarak bu eşleştirmeye gerek duymadan geliştirilen politika sayesinde davranış seçilebilir.

- Değer temelli yöntemler genelde deterministiktir fakat deterministik çözümler her zaman, özellikle de durum bilgilerinin yeterli olmadığı veya benzer olduğu problemleri çözmek için yeterli olmaz. Politika gradyan metotları ise stokastik politikalar da üretebilir.

Politika gradyan metotlarında, politika optimizasyonu genelde alt sınır yükseltme (MM - Minorize Maksimization) problemi olarak görülür. Adımları şu şekildedir:

- Bir politika tahmini yapılır ve bu politikaya göre değer fonksiyonu için bir alt sınır belirlenir.
- Belirlenen alt sınır optimize edilir.
- Optimize edilmiş politika yeni politika tahmini olarak kullanılarak aynı adımlar tekrarlanır.

Burada ortaya çıkan en büyük problem politikanın güncellenme (öğrenme) sıklığına karar vermektir. Çünkü politikanın çok sık güncellenmesi politikanın etkisinin kaybolmasına (vanishing) az güncellenmesi ise şişerek kaynakların yetmemesine (exploding) sebep olmaktadır.

Bu sorun Schulman vd. tarafından [12] ele alınmıştır. Geliştirmiş oldukları güvenli bölge politika optimizasyonu (trpo - trust region policy optimization) yönteminin temeli bir sınır belirlenerek politika güncellemelerinin güvenli bir bölge içerisinde kalmasını sağlamaktır. Bu sayede her bir döngüde bölge içerisindeki yerel optimum politikalar bulunarak sonuçta genel optimum politikaya ulaşılır. Bulunan tahminlerin kalitesine göre bölge genişletilip daraltılabilmektedir[13].

Güvenli bölge politika optimizasyon yöntemi ile güncelleme sıklığına karar verme ve yakınsama sorunları çözülmüştür. Fakat güvenli bölge kontrolü için kullanılan kısıtlama fonksiyonları sisteme fazladan yük bindirmektedir. Bu sorun ise bölge kontrolü için fazladan bir problem

çözmek yerine, Schulman vd. [14] tarafından proksimal politika optimizasyonu (ppo - proximal policy optimization) yön*temi ile kısıtlama probleminin de optimizasyona dahil edilerek bir tek fonksiyon elde edilmesiyle çözülmüştür[15].

Proksimal politika optimizasyon yöntemi günümüzde en çok kullanılan politika gradyan metodudur.

2.1.3 UNITY ML-Agents

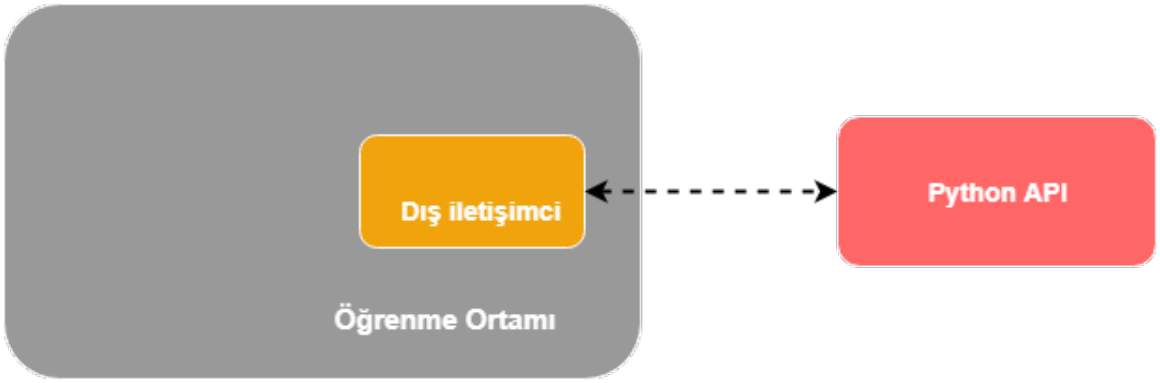
TensorFlow Google tarafından Python dili ile geliştirilen açık kaynak kodlu bir derin öğrenme çerçevesidir[sayfaya referans verelim]. TensorFlow bir çok platform ve programlama dilini desteklemektedir. Unity Makine Öğrenme Ajanları Araç Seti (The Unity Machine Learning Agents Toolkit - Unity ML-Agents) ise TensorFlow temelli, yine açık kaynak kodlu, bir Unity3d eklentisidir. Oyun ve simülasyonların akıllı ajanların eğitilebileceği bir ortam olarak kullanılmasını sağlar.

ML-Agents [16] ajanların pekiştirmeli öğrenme, imitasyon öğrenme, yapay sinir ağları gibi çeşitli makine öğrenme yöntemleri ile eğitilmesini sağlar. Eğitilen ajanlar NPC (Non Playable Character - Oyuncu tarafından kullanılamayan karakter) kontrolü, oyunların otomatik testi gibi amaçlarla kullanılabilir. Ajanın eğitilmesinde üç temel öğe vardır.

- **observations(gözlemler):** Ortamın belirli bir andaki durumunun ajan ile ilgili olan bölümüdür. Kesikli veya sürekli olabilir.
- **actions(davranışlar):** Ajanın yapabileceği hareketlerin listesidir. Yine kesikli veya sürekli olabilir.
- **reward(ödül):** Ajan davranışlarının istenilen doğrultuda olup olmadığını gösteren skalar değerlerdir.

Bu üç öğe probleme uygun bir biçimde ayarlandıktan sonra ajan eğitime hazır hale gelmektedir. Ajanlar gözlemlediği çevre sinyallerine göre seçtiği davranışlara karşılık ortam tarafından ödüllendirilir. **Eğitim(Training)**, ortam simülasyonunun defalarca tekrarlanarak bir turdaki ödül miktarını maksimuma çıkaracak bir politika belirlenmesi işlemidir.

ML-Agents üç temel bileşenden oluşmaktadır.



Şekil 2.3: ML-Agents Ana Bileşenleri

- **Öğrenme Ortamı:** Kullanılan Unity sahnesi ve içindeki tüm GameObject örneklerini içerir. GameObject Unity'nin temel sınıfıdır. Ortamdaki bütün karakterler ve cansız varlıklar bu sınıf ile ifade edilir. GameObject örnekleri içerdikleri bileşenlere göre özelleşir.
- **Python API:** Politika ya da davranış öğrenmede kullanılan tüm makine öğrenmesi algoritmalarını içerir. ML-Agents veya Unity'nin bir parçası değildir. Unity'den bağımsızdır ve Unity ile iletişimi dış iletişimci aracılığı ile sağlanır.
- **Dış İletişimci:** Öğrenme ortamı ile Python API'sini birbirine bağlar. Öğrenme ortamının içinde barınır.

Öğrenme ortamının üç bileşeni vardır.

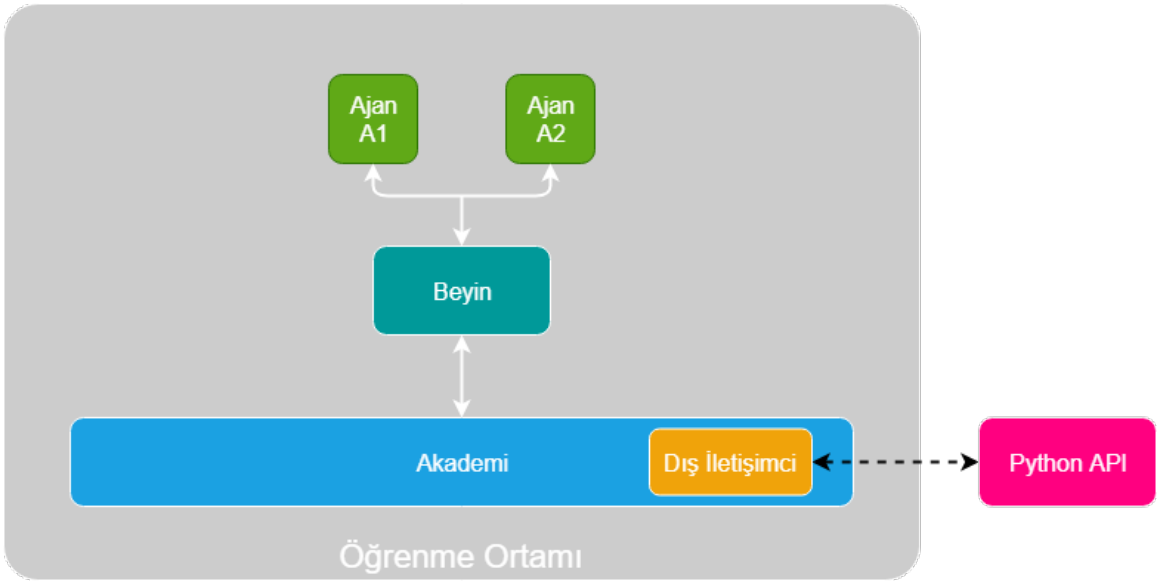
- **Ajan (Agent):** Öğrenmeyi gerçekleştirecek olan herhangi bir GameObject'e bağlı olabilir. Gözlemleri düzenleyip iletir. Kendisine bildirilen davranışı gerçekleştirir ve uygun bir ödül atar. Her ajan tam olarak bir beyne bağlıdır.
- **Beyin (Brain):** Politikaya bağlıdır ve ajanın belli bir anda ne yapması gerektiğine karar verir. Ajandan gelen gözlem ve ödül bilgilerine göre ajana yapması gereken davranış bilgisini döndürür. Üç beyin çeşidi vardır.
  - **Öğrenen (learning):** Kararlar Tensorflow modeli tarafından verilir. Tensorflow modeli öğrenilen bir politika gibi davranır ve beyin mevcut duruma uygun davranışı bu model yardımıyla seçerek ajanlara iletir. Harici ve dahili olmak üzere iki farklı öğrenen beyin tipi vardır. **Harici (external)** beyin, öğrenmenin devam

ettiği beyindir ve politika güncellenmeye devam eder. **Dahili (internal)** beyin ise öğrenme işlemi bitmiş beyindir ve belirlenmiş olan politika son halindedir.

  - **Kullanıcı(player):** Kararlar klavye gibi gerçek girdi aygıtlarını kullanan kullanıcıya bağlıdır. Beynin topladığı gözlem ve ödül bilgilerinin karakter davranışına bir etkisi yoktur.
  - **Bulgusal(heuristic):** Kararlar kodlanmış algoritmaya göre verilir.

  Bu üç beyin türü ve çeşitli miktarlarda ajanlar kullanılarak farklı türlerde öğrenme ortamları hazırlanabilir.

- **Akademi (academy):** Gözlem ve karar işlemlerini yönetir ve ajanlar ile beynin senkronizasyonunu sağlar. Python API ile öğrenme ortamı arasındaki bağlantıyı sağlayan dış iletişimci burada bulunmaktadır. Ortamla ilgili hız, çözünürlük gibi bazı parametreleri barındırır.



Şekil 2.4: Örnek öğrenme ortamı tasarımı

Bir öğrenme ortamında mutlaka bir adet akademi bulunur. Öğrenen tüm karakterler içinse bir beyne bağlı bir ajan olmalıdır. Eşit sayıda gözlem ve davranışı bulunan benzer ajanlar aynı beyne bağlanabilir.

Öğrenme sonucu istediğimiz performans ve öğrenme kalitesini elde etmek için Unity ML-Agents algoritmalarının sağladığı bazı parametreleri ayarlamak gerekebilir. Bu parametrelerin en başında **trainer** parametresi gelir ve temel öğrenme yöntemini belirler. Bu parametre

imitation (imitasyon) ve PPO değerlerinden birini alabilmektedir. Çalışmada Proksimal Politika Optimizasyonu tabanlı pekiştirmeli öğrenme kullanıldığı için bu trainer parametresi için PPO değeri seçilmiştir.
ML-Agents paremetrelerinden bu çalışma için önemli olanlar aşağıda açıklanmıştır. Diğer parametrelerin açıklamaları ML-Agents dökümanlarının ilgili sayfasında [17] bulunabilir.

Gamma: İndirim çarpanına (discount factor) karşılık gelmektedir. Yani ilerideki ödüllerin ne kadar önemli olduğunu belirler. Ajanın mevcut durumdaki hareketinin seçiminde gelecekteki ödüller de önemli ise bu değer büyük seçilmelidir. Eğer ajanın hareketi çoğunlukla anlık ödüllere bağlı ise bu değer küçük tutulmalıdır. Genellikle 0.8 ile 0.995 arasındaki değerler kullanılır.

Lambda: Genelleştirilmiş avantaj tahminine (generalized advantage estimate) karşılık gelir. Yeni bir değer tahmininin oluşumunda mevcut değer tahminin etkisini belirler. Düşük değer seçilmesi mevcut değer tahmininin etkisini artırır, değer yükseldikçe mevcut tahminin etkisi azalır ve yeni değer tahmini oluşumunda ortamın verdiği gerçek ödüller önem kazanır. Genellikle 0.9 ile 0.95 arası değerler kullanılır.

buffer_size: Bir öğrenme olması için yani modelin güncellenmesi için ne kadar tecrübe (durum, davranış, ödül) kazanılması gerektiğini belirler. batch_size parametresinin katı olmalıdır. 2048 - 409600 aralağındaki değerler uygundur.

batch_size: Bir gradyan güncellemesi için gerekli tecrübe miktarıdır. Kesikli davranış uzayı kullanıldığında 32 ile 512, sürekli davranış uzayı kullanıldığında ise 512 ile 5120 arasında bir değer seçilmelidir. Değerin küçük olması gradyanın sık güncelleneceği anlamına gelir.

num_epoch: Gradyan güncelleme sırasında tecrübe ara belleğine kaç kere girileceğini belirtir. Büyük değer alması daha kararlı bir öğrenme sağlar fakat öğrenmeyi yavaşlatır. Büyüklüğü batch_size parametresi lle orantılı olmalıdır. 3 ile 10 arasında seçilebilir.

learning_rate: Gradyan güncelleme gücüne karşılık gelir. Öğrenme kararsız ve ödüller çok tutarsız ise küçültmek gerekir. 1e-5 ve 1e-3 arası değer alabilir.

time_horizon: Ara belleğe alınmadan önce toplanacak tecrübe miktarını belirtir. Eğer bir turun tamamlanması için gereken adım sayısından küçük bir değerde olursa beklenen toplam

değer bir değer tahmini ile hesaplanır. Yüksek tutulursa ise sapma az olur fakat değişkenlik fazla olur. 32 ile 2048 arası değer alabilir.

max_steps: Eğitim süresindeki maksimum adım sayısını belirler. Büyüklüğü problemin karmaşıklığına göre artırılmalıdır. Normal aralığı 5e5 ve 1e7 arasıdır.

beta: Entropy büyüklüğünü ayarlar. Değer yüksek olduğunda politikanın rastgeleliliğini artırır, bu da sistemin exploration (daha yüksek değer arama) eğilimli olmasını sağlar. Yani eldeki ödül ile yetinilmeden başka yollar aranır. Normal şartlarda ödülün artması ile entropy yavaş bir düşüş göstermelidir. Eğer entropy çok hızlı düşüyorsa beta değeri arttırılmalı, gereğinden fazla yavaş düşüyorsa beta değeri yükseltilmelidir. Uygun değerler 1e-4 ile ile 1e-2 aralığındadır.

epsilon: Mevcut politika gradyanı ile yenisi arasındaki ıraksama eşiğine karşılık gelir. Değeri küçük tutmak daha kararlı ama daha yavaş bir öğrenmeye sebep olur. 0.1 ile 0.3 arasında olmalıdır.

normalize: Gözlem verilerinin normalize edilip edilmeyeceğini belirtir. Normalizasyon ortalama ve varyansa göre yapılır. Sürekli ve karmaşık bir gözlem uzayı olduğunda işe yarar fakat kesikli ve basit problemlerde zararlı olabilir.

num_layers: Kullanılan yapay sinir ağları için gözlem girdilerinden sonra gelen gizli katman sayısını belirtir. Basit gözlemlerde sayının az olması daha hızlı öğrenme anlamına gelir. Daha karmaşık problemlerde sayının artması faydalı olur. 1 ile 3 arasında tam sayı değerleri alabilir.

hidden_units: Sinir ağlarının her bir katmanındaki nöron sayısın belirtir. Basit problemlerde sayı küçük olabilir fakat dönecek olan davranışın gözlem verilerinin karmaşık etkileşimleri ile elde edildiği sistemlerde değer yüksek tutulmalıdır. 32 ile 512 arasında olmalıdır.

2.2 Karakter Animasyonu

2.2.1 Kaçınma Animasyonu

Kaçma, bir organizmanın kendisini rahatsız eden bir etkenden uzaklaşmaya çalışmasıdır. Kaçınma ise organizmanın, henüz rahatsız edici etken gelmeden bu etkenin yaklaştığını belirten ve çeşitli duyularla algılanabilen uyarıcıları farkederek ondan kurtulmaya çalışması işidir. Yani kaçma içinde bulunulan bir durumdan kurtulmayı ifade ederken kaçınma o duruma hiç düşmeme çabasını ifade eder. Kaçınmaya örnek olarak bir kedinin henüz köpek gelmeden, sesini duyduğunda ortamı terketmesi ya da bir insanın kendise doğru yaklaştığını gördüğü bir taştan, taşın kendisine temas etmesine engel olacak hareketleri örnek verilebilir. Kaçınma bilinçli olarak ya da gelişmiş refleksler ile otomatik olarak yapılabilir.

Kaçınma hareketleri, dövüş oyunları gibi karakterlerin birbirleriyle fiziksel etkileşim kurduğu animasyon uygulamalarında büyük öneme sahiptir. Bu tip uygulamalarda sanal karakterlerin hareketlerinin rakibinin hareketlerine göre dinamik olarak şekillenmesi gerekmektedir. Bu sebeple gerçekçi kaçınma hareketlerinin üretilmesi karmaşık bir problem teşkil etmektedir. Günümüz bilgisayar oyunlarında karakterlerin hareketleri önceden hazırlanmış sınırlı sayıdaki hareketten uygun olanı seçerek üretilebilmektedir. Bu yöntem ise gerçek dövüşçülerin saldırılardan minimum hareketle kaçınmalarını canlandırmada yetersizdir ve bilgisayar oyunlarındaki karakterler saldırılardan kaçınmak için hala büyük adımlar ve uzun sıçrayışlar kullanmaktadır.

Literatürde gerçekçi kaçınma hareketleri elde etmek için yapılmış çeşitli çalışmalar yer almaktadır.

Zordan vd. çalışmalarında bir gözetimli öğrenme çeşidi olan destek vektör makinesi (support vector machine) yöntemini kullanarak kaçınma hareketi üretmiştir[18] . Gelen saldırının yön, hız ve konum bilgilerini içeren bir özellik vektörü oluşturularak bu vektöre karşılık en uygun animasyon, hareket yakalama verilerinin bulunduğu veritabanından seçilmiştir. Başlangıç pozu olarak karakterin düz bir şekilde ayakta durduğu hali alınmış ve gelen saldırılara karşı kaçınma hareketi seçerken dengenin bozulmayacağı hareketlerin seçimine dikkat edilmiştir. Bununla beraber uzuvlar hayati önemlerine göre derecelendirilmiş ve kaçınmada bu derecelendirme dikkate alınmıştır. Çalışma genel olarak başarılı olmuştur fakat saldırı hacminin başlangıç ve bitiş konumları ile vücudun kaçınma hareketi boyunca başlangıç ve

bitiş konumlarının kesiştiği durumlar olduğu için saldırıdan kaçınılamadağı durumlar da görülmüştür. Ayrıca sistemin yeterince gerçekçi olabilmesi için çok miktarda hareket yakalama verisine ihtiyaç vardır ve bu hem masraflıdır hem de gözetimli öğrenme eğitim süresini uzatır.

Shum vd. çalışmalarında yine hareket yakalama verileri kullanmışlardır [19]. Amaçları sadece kaçınma hareketleri üretmek değil karşılıklı dövüşen iki karakterin hareketlerini sentezlemektir. Bunun için öncelikle tek bir karakter için çeşitli hareketler yakalanmış ve bu hareket yakalama verileri tekme, yumruk, çökme, sıyrılma gibi gruplara ayrılmıştır. Bu gruplar ise kaçınma ve saldırı olmak üzere iki ana grubun altında toplanmıştır. Hamleler seçilirken min-maks ağaç yapısı kullanılmıştır. Hamlelerin ağaç yapısında derecelendirilmesi için konum, değer ve kontrol gibi fonksiyonlar kullanılmıştır. Konum; hareket sonrası istenilen konuma uzaklığı, değer; verilen ile alınan zarar arası farkı ifade eder. Kontrol ise manuel olarak kullanıcının istediği hareketleri derecelendirmesi işidir. Karşı hamle seçiminde saldırı-savunma tablosu kullanılmıştır. Bu tablo ise saldırı ve savunma hareketi konumlarına göre düzenlenmiş olmakla birlikte kullanıcı tarafından manuel olarak da seçilebilir. Sistem hareket yakalama verilerinin gruplanması dışında tamamen otomatik olarak çalışabilmektedir. Çalışma genel olarak başarılı olsa da gerçekçiliği hareket yakalama verilerinin çeşit ve miktarına bağlıdır. Saldırı ve savunma hareketlerinin sadece önceden düzenlenmiş hareket yakalama verileri içinden seçilmesi ise yeni bir hareket eklenmesini zorlaştırmaktadır. Ayrıca her saldırıya karşılık sadece bir savunma hareketi seçilmesi tekdüze bir görünüme sebep olup gerçekçiliği azaltmaktadır.

Oshita vd. çalışmalarında yine hareket yakalama verilerinden faydalanmıştır [20]. Eldeki hareket yakalama hareketleri ile bir hareket çizgesi (motion graph) oluşturulmuş ve bu hareketler kaçınma kısmı tek parça olacak şekilde bölümlere ayrılıp işaretlenmiştir. Daha sonra gelen saldırılara en uygun hareketler seçilerek uygulanmıştır. Uygun hareketin seçiminde hareket boyunca saldıran hacmin ve vücudun temas etmemesi temel alınmıştır. Bunun için üç hacim ele alınmıştır. Saldırı hacmi, saldıran cismin hacminin hareketi boyunca kapladığı toplam hacmi ifade eder. Vücut hacmi, kaçınacak olan karakterin toplam hacmidir ve kaçınma hacmi ise karakterin ortamda kaçınma hareketinden önce bulunan fakat sonrasında başka yere taşınmış olan vücut kısmıdır. Bu hacimler, kesişme kontrolü daha kolay olan kürelere bölünerek GPU destekli algoritmalarla kesişme kontrolü yapılmış ve temas etmeyen en uygun hareket seçilmiştir. Kesişmeyi engellemek için gerektiğinde kaçınma hareketlerinin hızı

yükseltilip düşürülebilmektedir. Literatürdeki diğer hareket çizgesi yöntemlerinde olduğu gibi, bu yöntemde de sistem veriye çok bağımlıdır ve başarısı, hareket yakalama verisinin sayısına ve kalitesine bağlıdır. Veri sayısı arttıkça kalite artar fakat tüm verilerin hareket verisi oluşturulurken incelikle işlenmesi çok fazla emek gerektirir.

Bir başka çalışmada ise, Oshita vd. verilen bir animasyonu deforme ederek, hareketi yapan karakterin engellerden kaçınmasını sağlayan bir yöntem geliştirmişlerdir [21]. Bu yöntemde girdi olarak alınan hareket, bir kafesin (lattice) içine alınmaktadır. Daha sonra bilinen çözgü (warping) teknikleri kullanılarak kafes uygun şekilde deforme edildiğinde karakterin, kafes kesişim noktalarına denk gelen vücut bölümleri bükülmektedir. Bu sayede, kafes engelin uzay zaman hacmi baz alınarak deforme edildiğinde, animasyon boyunca karakterin engelle çakışması engellenerek yeni bir kaçınma hareketi üretebilmektedir. Yöntem, elde bulunan hareket yakalama verilerinin değiştirilerek farklı şekillerde kullanılmasınama imkan verdiği için önemlidir. Bu sayede her bir engel için ayrı hareket yakalanması ihtiyacı ya da birden fazla engel için aynı kaçınma hareketinin yapılması sorunu çözülmüş olmaktadır.

2.2.2 Pekiştirmeli Öğrenme İle Karakter Animasyonu Üretme

Animasyon üzerine yapılan çalışmaların çok büyük bir kısmı hareket yakalama verilerini kullanmaktadır. Fakat hareket yakalama işlemleri zahmetlidir ve gerekli araçlar pahalı olduğu için herkesin edinmesi mümkün değildir. Ayrıca birçok hareket için uygun ortam ve uygun aktör bulmak da oldukça zordur. Bahsedilen sebeplerden ötürü, pekiştirmeli öğrenmenin de yaygınlaşması ile hareket yakalama verilerini taklit ederek yeni hareketler üretmek ve hatta hiç hareket yakalama verisi olmadan animasyon üretmek için pek çok çalışma yapılmıştır.

Tan vd. çalışmalarında pekiştirmeli öğrenme kullanarak sanal insansı bir modele insanlarda olduğu gibi pedal çevirerek bisiklet sürmeyi öğreten bir sistem geliştirdi[22]. Bu sistemin sonucu olarak, insansı model düz bir yolda bisiklet sürebilmekte ve arka teker üstünde gitme, sıçrama gibi özel hareketler yapabilmektedir. Sadece bisiklet sürmek için ve özel hareketlerin her biri için sistem ayrı ayrı eğitilmektedir.

Çimen vd. üretilmesi daha kolay olan yürüme, ayakta durma ve erişme hareketlerini birleştirerek dinamik yakalama hareketleri üreten bir sistem geliştirmişlerdir [23]. Sistemde kinematik bir sanal insan modeli kullanılmıştır ve yürüme, ayakta durma ve erişme hareketleri

üst seviye kontrol mekanizmaları ile üretilmektedir. Bu hareketlerin başlama zamanları, konumları ve hızları ise pekiştirmeli öğrenme yardımı ile deneme yanılma ile öğrenilmektedir. Çalışma sonucunda rastgele fırlatılan toplar için gerçekçi bir şekilde yakalama hereketleri üretebilen bir sistem ortaya çıkmıştır.

Peng vd. pekiştirmeli öğrenme ile iki boyutta fizik temelli sanal karakter modelleri için sürekli gezinme hareketleri üreten bir çalışma yapmışlardır [24]. Düz yollarda gezinme hareketleri üreten sistemlerin aksine, çalışmalarındaki hareketler dinamik ve momentum temellidir. Hareketlerin dinamik olarak üretilmesi sebebiyle boşluklar, basamaklar ve duvarlar içeren arazilere uyum sağlamaktadır. Çalışmada 21 eklemli köpek modeli ve 7 eklemli iki ayaklı model kullanılmıştır. Derin öğrenme teknolojilerinin gelişmesiyle Peng vd. derin pekiştirmeli öğrenme kullanarak daha etkin bir yöntem geliştirmişlerdir [25]. Bu çalışmalarında 21 eklemli köpek modeli ile birlikte 19 eklemli dinazor modeli kullanılmıştır.

Peng vd. bir diğer çalışmalarında [26] gezinme hareketlerini yine pekiştirmeli öğrenme ile bu kez 3 boyutta geliştirmişlerdir. İki bölümden oluşan öğrenme aşamasının ilk bölümünde kabaca yürüme hareketleri öğrenilmiş, ikinci bölümde ise arazi şekillerine göre zamanlı bir şekilde hareketler öğrenilmiştir. iki ayaklı model kullanılan sistem engelli arazi koşullarında sorunsuz gezinme yapabilecek bir başarı sağlamıştır.

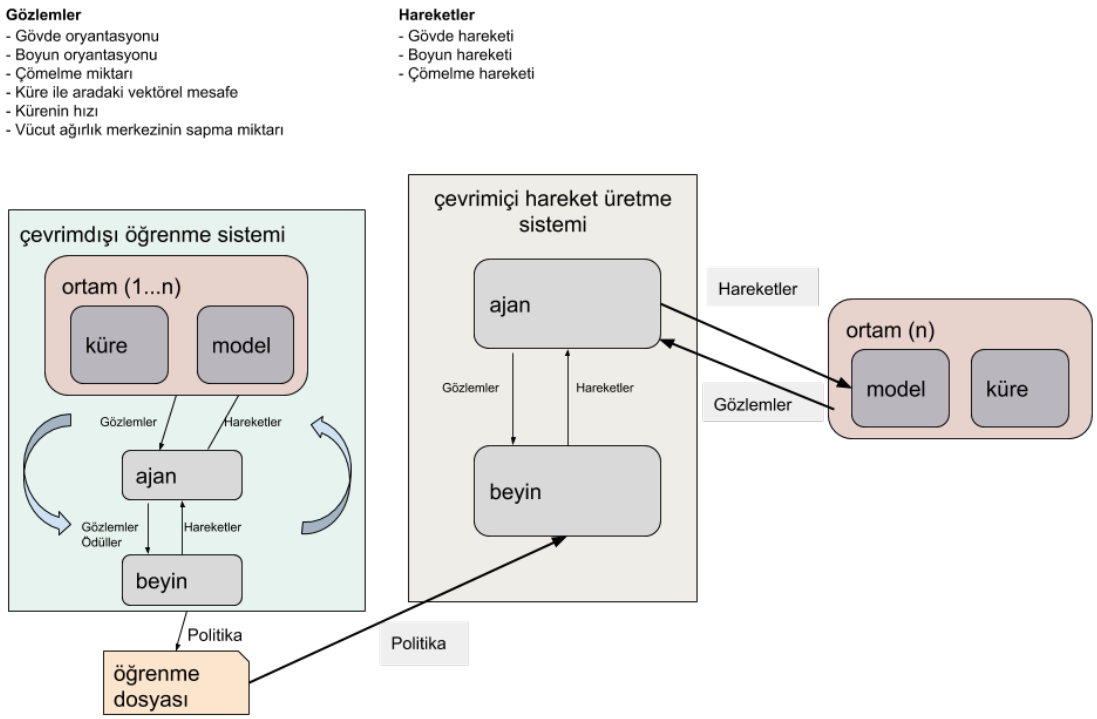
Sıfırdan pekiştirmeli öğrenme yöntemleri ile animasyon üretilen çalışmaların yanı sıra gerçek hareket verilerinden yola çıkarak pekiştirmeli öğrenme ile benzer hareketler üretmeyi öğrenen çalışmalar da yaygındır. Peng vd. tarafından geliştirilen hareket yakalama verilerini taklit eden sistem [27] ve doğrudan videolardaki karakter davranışlarını taklit eden sistem [28] bu tip çalışmalara örnek olarak verilebilir.

Hareket yakalama verilerini taklit eden bir diğer çalışmada [29] Chentanez vd. iki farklı ajan kullanmışlardır. Bu ajanlardan biri hareket yakalama verilerini taklit etmeyi öğrenirken diğeri kullanılan sanal insan modelinin yapılan hareketler sonrasındaki pozunun fiziksel olarak doğruluğunu kontrol etmektedir. Çalışmada taklit edilen hareketler, halka açık birçok hareket yakalama veritabanı birleştirilerek elde edilen havuzdan seçilmektedir. Sistem genel olarak başarılı olsa da takla atma, break dans gibi fiziksel takibi ve denge kontrolünün zor olduğu hareketleri taklitlerde yetersiz kalmıştır.

3 DERİN PEKİŞTİRMELİ ÖĞRENME İLE DOĞAL KAÇINMA HAREKETLERİNİN ÜRETİLMESİ

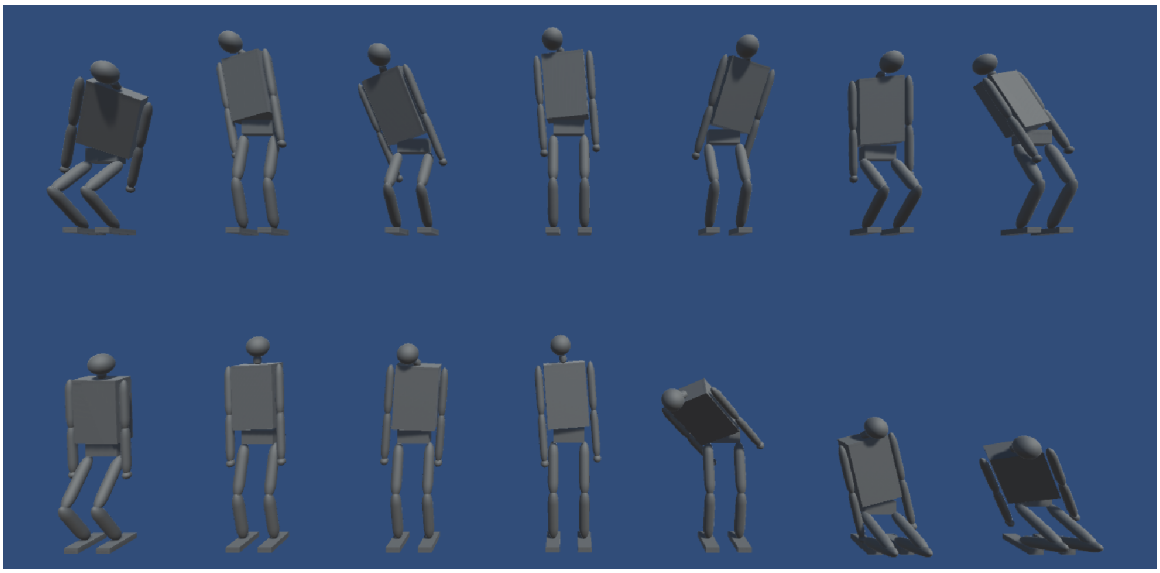
3.1 Sisteme Genel Bakış

Sistemin çevrimdışı öğrenme ve çevrimiçi hareket üretme olmak üzere iki ana parçası vardır. Bu iki parça için ayrı sahneler kullanılmıştır. Çevrimdışı öğrenme sisteminin çıktısı olan öğrenme dosyası çevrimiçi hareket üretme sisteminde kullanılmaktadır. Şekil 3.1 de genel bir sistem diagramı gösterilmektedir.



Şekil 3.1: Sisteme genel bakış.

Her iki sistem parçası da temelde Unity ML-Agents beynine bağlı bir Unity ML-Agents ajanı içeren kinematik bir sanal insan modeli ile bir küreden oluşmaktadır. Sanal insan modelinin başlangıç pozu olarak ayakta durduğu ve tüm eklem açılarının sıfır olduğu hali seçilmiştir. Modelin ayakları yerde sabittir ve hiçbir şekilde konumu ya da doğrultusu değişmemektedir. Modelin uzuvları insan vücudunun fizyolojik yapısına aykırı olmayacak şekilde bilek, diz, kalça, bel ve boyun eklemlerini kullanarak çeşitli dönme hareketleri yapabilmektedir.

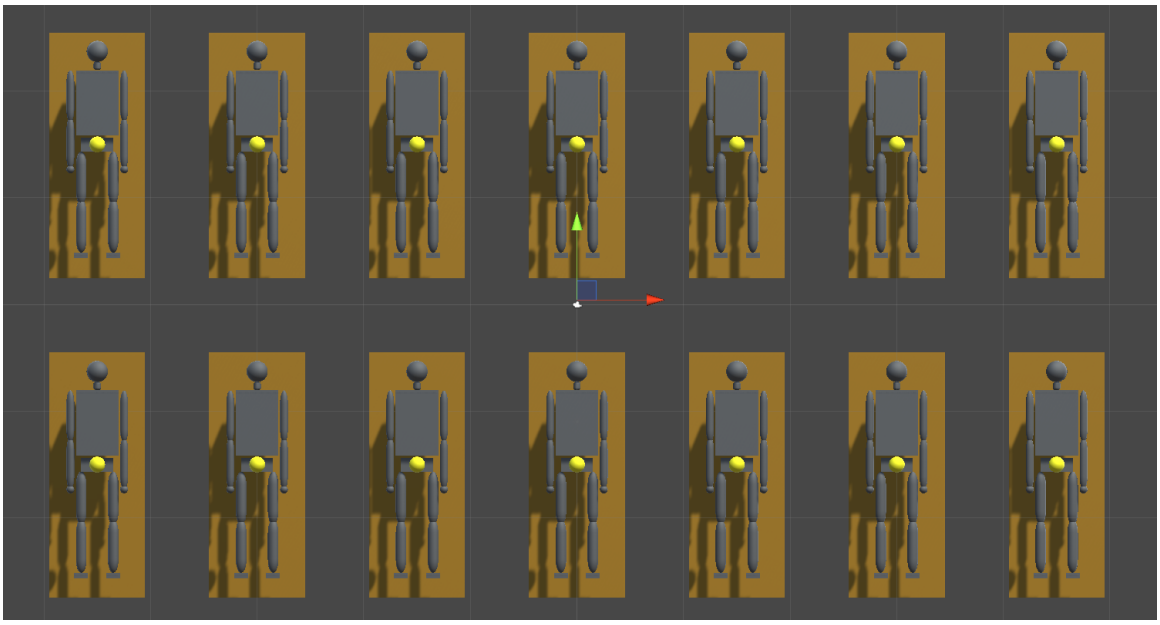


Şekil 3.2: Sanal insan modelinin çeşitli hareketleri.

Çevrimdışı öğrenme sisteminde sanal karakterin derin pekiştirmeli öğrenme ile fırlatılan kürelerden kaçınmayı öğrenmesi amaçlanmaktadır. Bu sistemde küre, sanal insan modeline doğru rastgele konumlardan, rastgele doğrultu ve hız ile fırlatılmaktadır. Küre ile sanal insan modelinin herhangi bir uzvunun temas etmesi ya da kürenin temas etmeden sanal insan modelini geçmesi durumunda öğrenme sürecinin bir turu tamamlanmaktadır ve sanal insan modeli diğer tur için başlangıç pozuna dönmektedir. Öğrenmenin başlangıcında, öğrenme algoritması model ve kürenin durumu ile yapılacak hareketler arasında rastgele bir politika belirler. Belirlenen politika öğrenme aşaması boyunca, alınan ödüllere göre güncellenir ve güncel politika ile ilgili bilgiler bir öğrenme dosyasına kaydedilir.

Çevrimiçi hareket üretme sistemi, çevrimdışı öğrenme sisteminde üretilen politikanın kullanılması aşamasıdır. Bu aşamada küre sanal insan modelinin kullanıcı tarafından seçilen bölgesine doğru fırlatılmakta ve tepki hareketi çevrimdışı sistem tarafından öğrenilen politikaya göre gerçek zamanlı olarak üretilmektedir.

3.2 Doğal Hareket Üretimi İçin Pekiştirmeli Öğrenme Sistemi



Şekil 3.3: Eğitim ortamı başlangıç görünümü.

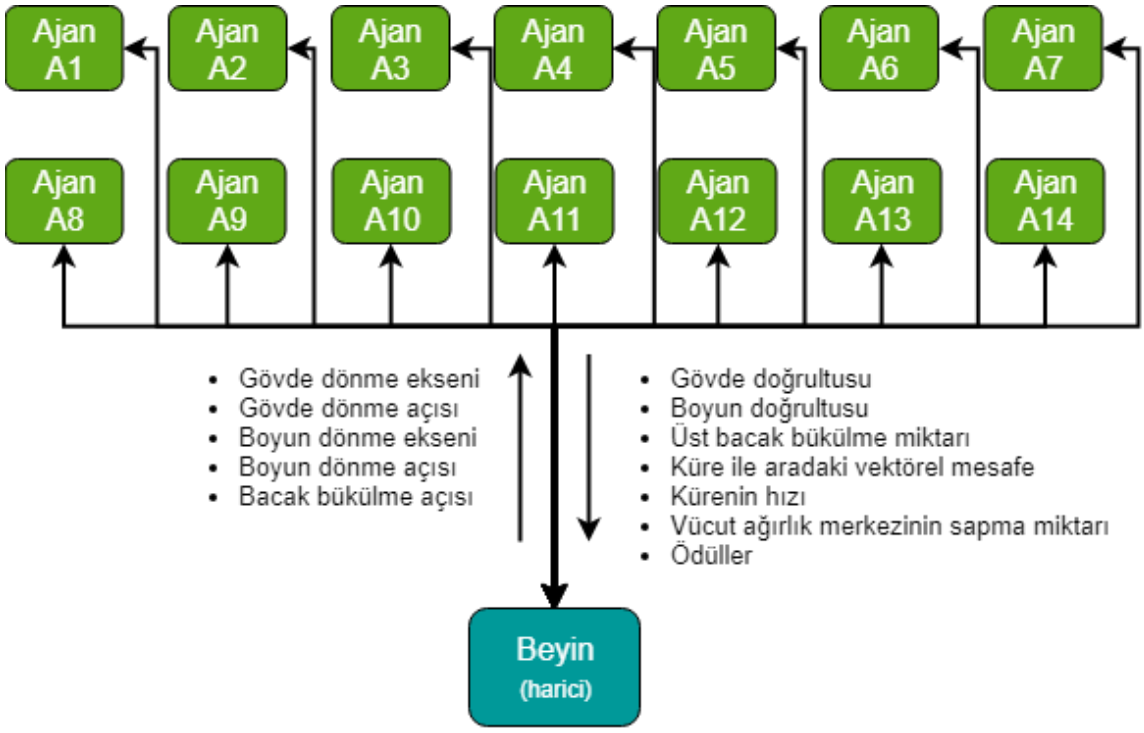
Sistemimizin çevrimdışı pekiştirmeli öğrenme aşaması için hazırlanan sahnede 1 beyine bağlı 14 ajan bulunmaktadır. Her bir ajanın eğitimi ile aynı öğrenme dosyası güncellenmektedir ve bu şekilde eğitimin süresi kısalmaktadır (bkz. Şekil 3.4). Henüz öğrenme başlamadan sistemin başlangıç aşamasında, sanal insan modelinin tüm uzuvlarının kütleleri hacimleri cinsinden hesaplanır ve bellekte tutulur. Bu işlem, her adımda ağırlık merkezi hesaplanırken sistemi matematiksel işlemler ve masraflı Unity metotlarından kurtarmaktadır. Adım ajan ile beyin arasındaki hareket ve gözlem alışverişi olarak tanımlanabilir. Tur ise kürenin iki kez fırlatılması arasındaki zamandır.

Sanal bir insan modelinin kendisine doğru gelen cisimlerden gerçekçi bir şekilde kaçınabilmesi için sağlaması gerekenler aşağıdaki gibi sıralanabilir.

- Modeldeki uzuvların hareketleri, normal bir insan vücudunda kendisine karşılık gelen uzvun bağlı olduğu eklemin müsaade ettiği eksen ve açı sınırlarını aşmamalıdır.
- Yer çekimine aykırı pozlara izin verilmemelidir (vücut ağırlık merkezi izdüşümü ayakların köşelerinin sınırlarından çıkmamalıdır.)
- Kaçınma sırasında gerekenden fazla hareket edilmemelidir.
- İnsanüstü bir hızla hareket edilmemelidir.

- Sürekli aynı ya da benzer hareketler tekrarlanmamalıdır.

Çevrimdışı öğrenme aşamasında yapılacak gözlemler ve gözlemlere karşılık verilecek ödüller seçilirken, modelin belirtilen kriterlere uygun hareketleri yapmayı öğrenmesi amaçlanmıştır.



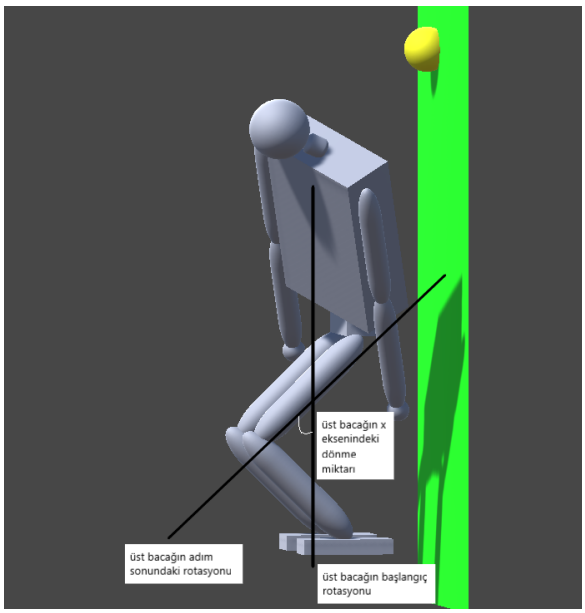
Şekil 3.4: Eğitim diagramı

3.2.1 Gözlemler

Gözlemler, pekiştirmeli öğrenme algoritmasının ortamın durumunu anlamak için ortamdan edinmesi gereken bilgilerdir(bkz 2.1.3) Gözlemlerin nasıl seçilmesi gerektiğinin belirli bir yöntemi yoktur ve probleme göre değişiklik göstermektedir. Gözlemlerin gerekenden fazla olması politikanın belirlenme süresini uzatırken eksik olması durumunda başarılı bir politika belirlemek mümkün değildir. Öğrenme süresiyle başarımı dengelemek için yaptığımız denemeler sonucunda, problemimiz için aşağıda açıklanan 17 adet gözlem değerinin kontrol edilmesine karar verilmiştir.

- **Çökme miktarı:** Sanal insan modeli dizlerini insan fizyolojisine uygun olarak, yalnızca bir eksende bükebilmektedir. Çökme hareketi gerçekleştirilirken modelin dengesinin ve hareketin doğal görünümünün bozulmaması için her iki kalçadan ayaklara doğru

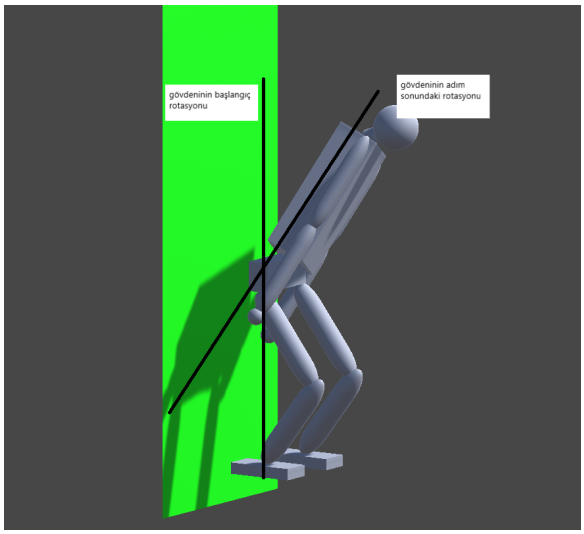
bir seri dönme işlemi yapılır. Ayakların konum ve oryantasyonu değişmediği için diz, bilek ve kalça eklemlerinin dönme ekseni de değişmez ve dönme açıları orantılı olduğundan çökme miktarı gözlemi için bu açılardan sadece biri yeterlidir. Çalışmada çökme miktarının kontrolü için üst bacağın x ekseninde her bir öğrenme adımı sonundaki dönme miktarı gözlemlenmektedir. Adım başlangıcında üst bacağın oryantasyonunun bütün bileşenleri sıfır olduğu için dönme miktarı şekil 3.5'te görüldüğü gibi üst bacağın oryantasyonunun x bileşenine eşittir.



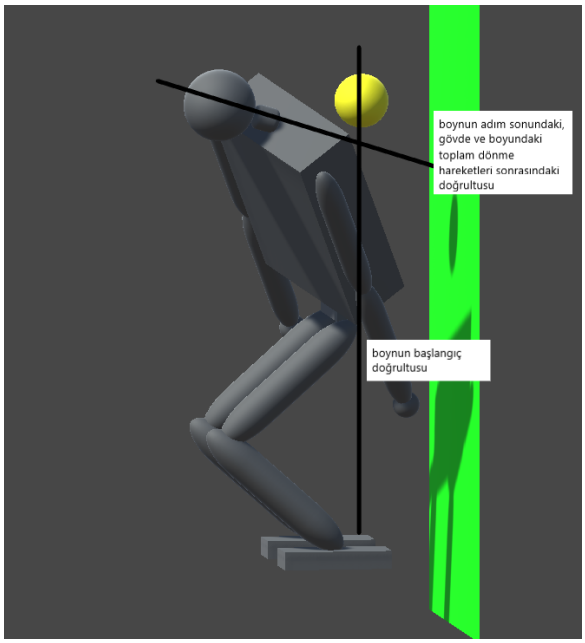
Şekil 3.5: Çökme miktarı gözlemi

- **Gövde oryantasyonu:** Modelin gövdesi belden, normal insan vücudunun müsaade ettiği açılarla x,y ve z eksenlerinde dönebilmektedir. Dönme miktarının kontrolünde, pelvis oryantasyonu değişmediği için, Şekil 3.6 da gösterildiği gibi gövdenin o anki oryantasyonunun gözlemi yeterli görülmüştür. Çalışmadaki diğer üç serbestlik derecesine sahip vücut parçalarının oryantasyonunda olduğu gibi, gövde oryantasyonu quaternionlar ile temsil edilmektedir. Bu sebeple quaternion sınıfının her bir bileşeni için bir tane olmak üzere toplamda dört değer gözlemlenmektedir.

- **Boyun oryantasyonu:** Benzer biçimde üç serbestlik derecesine sahip boyun oryantasyonu da quaternion olarak gözlemlenmektedir (bkz. Şekil 3.7).

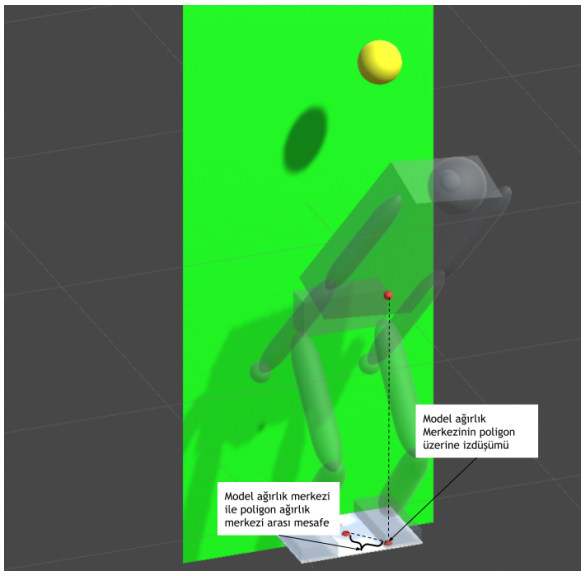


Şekil 3.6: Gövde oryantasyonundaki değişimin gözlemi



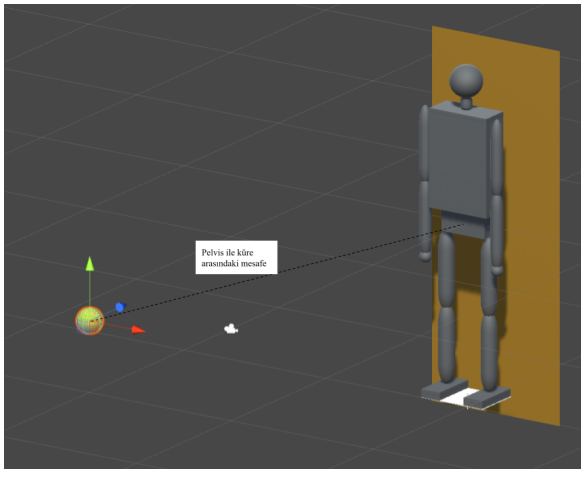
Şekil 3.7: Boyun doğrultusundaki değişimin gözlemi.

- **Ağırlık merkezi konumu:** Gerçekçi hareket elde etmek için sadece eklemlerin izin verdiği sınırları kontrol etmek yeterli değildir. Doğru eklem sınırlarını aşmadığı halde, fizik kurallarına aykırı hareketler de üretilebilir. İki ayağı yerde olacak biçimde ayakta duran bir insan modelinin dengesini koruyabilmesi için kütle merkezinin izdüşümünün ayakların yerde oluşturduğu poligonun(destek poligonu) içinde kalması gerekmektedir [30]. Bu sebeple her bir öğrenme adımında, vücudun kütle merkezinin izdüşümünün destek poligonunun kütle merkezine olan vektörel uzaklığının x ve z bileşenleri gözlemlenmektedir (bkz. Şekil 3.8).



Şekil 3.8: Model ağırlık merkezinin izdüşümü ile destek poligonu ağırlık merkezi arasındaki uzaklık

- **Küre ile aradaki mesafe:** Kaçınma hareketlerinin zamanlamasının doğru ayarlanabilmesi için ve kürenin modele hangi yönden yaklaştığını bilmek için sanal insan modelinin küre ile arasındaki vektörel mesafe bilgisine ihtiyaç vardır. Bunun dışında fırlatılan kürenin veya sanal insan modelinin konumlarının ayrı ayrı gözlemine gerek yoktur. Ayrıca hareketli uzuvların oryantasyonları gözlem verilerine dahil olduğu için her birinin konumu için ayrı gözleme de gerek yoktur. Bu sebeplerle her adımda fırlatılan küre ile pelvis arasındaki mesafe vektörü gözlemlenmektedir (bkz. Şekil 3.9).



Şekil 3.9: Pelvis ile küre arasındaki mesafe

- **Kürenin hızı:** Kaçınma hareketlerinin zamanlamasının ayarlanması ve topun yaklaşma

yönü bilgisi için fırlatılan kürenin hızı da vektörel olarak takip edilmektedir. Kürenin hız vektörünün üç bileşeni ile birlikte toplamda 17 adet değer gözlenmektedir.

3.2.2 Ödüller

Pekiştirmeli öğrenme algoritmasının optimum politikayı etkili bir biçimde bulmasını sağlamak için ödül tasarımının iyi yapılması çok önemlidir ((bkz 2.1.3)). Verilen ödülleri adım başına verilen ödüller ve tur sonunda verilen ödüller olmak üzere iki başlık altında inceleyebiliriz. Adım başına verilen ödüller, her bir adımda yapılmış hareketler ve o anki duruma göre verilen küçük değerli ödüllerden oluşmaktadır. Tur sonuna ulaşıldığı durumlarda ise çok daha büyük ödüller verilmektedir. Adım ödülleri ve tur sonu ödülleri aşağıda detaylı olarak açıklanmıştır.

Adım ödülleri: Çalışmamızda adım başına verilen ödüller sanal insan modelinin her bir adımda yaptığı hareketlere verilen ödüller olarak tasarlanmıştır. Gerçekçilik için temel kriterlerden birinin de minimum efor gerektiren hareketler üretmek olduğundan bahsedilmişti. Bu sebeple hareketlere karşılık olarak verilen ödüller gereksiz hareketi engellemek amacıyla negatif ayarlanmıştır. Bununla birlikte adımlarda verilen ödüllerin toplamının tur sonundaki başarılı yahut başarısız olma ödüllerini bastırmaması için adım ödüllerini hesaplamada $R = \frac{e^{\frac{\alpha}{100}}}{100}$ eşitliğinin kullanılması uygun görülmüştür. Burada α yerine gelecek değerler aşağıda açıklanmaktadır.

- **Çökme miktarı:** Aslında sanal insan modeli, vücudun üst kısmına gelen kürelerden her durumda eklemlerin izin verdiği kadar çökme hareketi yaparak da kurtulabilir. Fakat normalde ufak bir gövde hareketi ile sıyrılabileceği bir durumda tamamen çökmeyi seçmesi gerçekçiliği bozar. Bu yüzden çömelme hareketi için seçilen negatif ödülün çok küçük olmamasına özellikle dikkat edilmiştir. Ödül hesaplama eşitliğinde α değeri olarak, üst bacağın içinde bulunulan duruma kadar x ekseni etrafında yapmış olduğu toplam açının üç katı verilerek ödül hesaplanır. Sanal insan modeli hiç çökme hareketi yapmadıysa negatif ödül verilmez. Bu ise modelin gereksiz yere çökmesinin önüne geçmektedir.

- **Gövde açısı:** Gövdenin belden hareketleri daha az efor ile kaçınmayı mümkün kıldığı için gerçek dövüşçüler de mümkün olduğunda gövde hareketleri ile sıyrılmayı tercih

etmektedir. Bu nedenle gövde hareketleri ile sıyrılma gerçekçi kaçınma hareketleri elde etmek için oldukça önemlidir. Bunu sağlamak için hem çökme hem de gövde hareketleri ile kaçınmanın mümkün olduğu durumlarda gövde hareketlerinin seçilmesi için bacak hareketlerinden daha az negatif ödül verilmiştir. Ödül, gövde ile bel arasındaki açının α değeri olarak ödül hesaplama fonksiyonuna gönderilmesiyle hesaplanır. Çökme hareketlerinde olduğu gibi gereksiz hareketlerin önüne geçmek için negatif ödül yalnızca gövdenin hareket ettiği adımlarda verilir.

- **Boynun açısı:** Boyun hareketleri doğrudan modelin kafasını hedefleyen küreleri savuşturmada gerçekçi bir görünüm sağlamaktadır. Ayrıca diğer kaçınma hareketleri gerçekleştirilirken kafanın da hareket etmesi sabit durmasından daha gerçekçi bir görünüm sağlamaktadır. Bu sebeple en küçük negatif ödül boyun hareketlerine verilmektedir. Ödül, boyun ile gövde arasındaki açının yarısı ödül hesaplama fonksiyonuna α değeri olarak verilerek hesaplanmıştır.

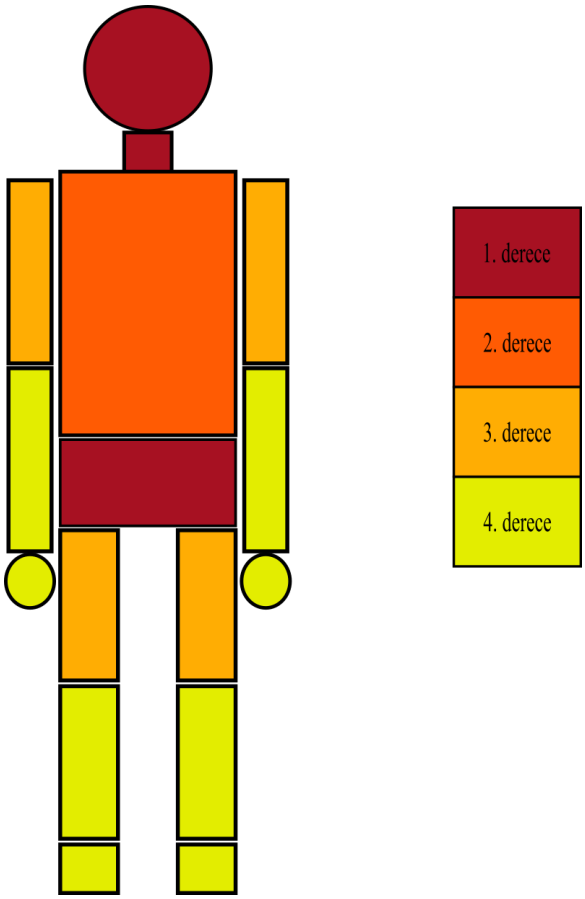
- **Ağırlık merkezi konumu:** Sanal insan modelinin dengede kalmasını sağlamak için denge konumundan uzaklığı ile orantılı olarak negatif ödül verilmektedir. Negatif ödül, modelin ayaklarının oluşturduğu destek poligonunun kütle merkezi ile sanal insan modelinin kütle merkezinin bu poligona iz düşümü arasındaki mesafe ödül eşitliğine verilerek hesaplanmıştır. Bu ödül, modelin denge durumuna gelmesini sağlayacak hareketler yapmasını zorlamak amacıyla sadece bir hareket olduğunda değil her adımda verilmiştir.

Tur sonu ödülleri:

- Tur sonunda kaçınmanın başarısız olması durumunda verilecek negatif ödüller seçilirken kürenin çarptığı bölgelerin hayati önemi baz alınmıştır. Bunun için vücut bölümleri Zordan vd. çalışmasında olduğu gibi[18] hayati öneme göre derecelendirilmiştir (bkz. Şekil 3.9).

- Modelin kütle merkezinin izdüşümü destek poligonunun dışına çıktığında sistem başarısız kabul edilerek -5 negatif ödül verilmiştir.

- Kürenin modele temas etmeden geçmesi durumunda kaçınma hareketi başarılı olduğu için 5 pozitif ödül verilmiştir.



Şekil 3.10: Vücut bölümlerinin önem dereceleri

3.2.3 Hareketler

Çalışmamızda sürekli hareket uzayı kullanılmıştır. Ajanlar beyinden, üç serbestlik derecesine sahip eklemler için bir eksen ve bir açı değeri olmak üzere dört değer, bir serbestlik derecesine sahip eklemler için tek bir açı değeri almaktadır. Yapılan hareketler aşağıdaki gibidir.

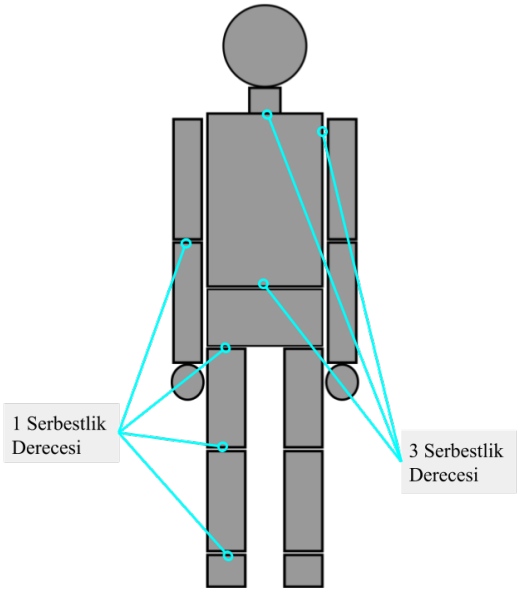
- **Çökme hareketi:** Çökme hareketi tek eksende yapıldığı için bir değere göre hareket yapılır. Ajana α kadar bir değer geldiğinde üst bacaklar kalça eklemine göre x ekseninde α kadar bir dönme gerçekleştirir. Alt bacak üst bacağın α miktarı dönmesinden de etkilendiği için dize göre başta ilk durumuna ulaşmak için $-\alpha$ kadar, sonra ise bükülme için bir $-\alpha$ kadar daha dönme yapar. Yani alt bacaklar x ekseninde toplamda -2α kadar dönme yapmaktadır. Son olarak ayaklar, oryantasyonunda oluşan farkı ($\alpha - 2\alpha = -\alpha$) düzelterek ilk durumuna dönmek için yine x ekseninde bileklere göre α kadar bir dönme yapar.

- **Gövde hareketleri:** Gövde belden her üç eksende de dönüş yapabilmektedir. Bu yüzden ajan dönme ekseni için üç, dönme açısı için bir olmak üzere toplamda 4 değer alır. Gövde, dönme açısı sıfırdan farklı ise dönme ekseni etrafında bele göre dönme açısı kadar dönüş gerçekleştirir.

- **Boyun hareketleri:** Modelin boynu gövde-boyun ekleminden her üç eksende de dönüş yapabilmektedir. Bu sebeple yine dönme ekseni ve miktarı için toplam dört değer alınır.

3.3 İmplementasyon ve sonuçlar

Bu bölümde implementasyon detayları ve sistemin başarısını ölçmek için uyguladığımız kullanıcı testlerinden bahsedilmektedir. Çalışmada eklemli yapıdaki, kinematik bir sanal insan modeli kullanılmıştır. Modelin uzuvları gerçek insanda olduğu gibi eklemlerinden tek eksende ya da üç eksende dönebilmektedir. Modelin vücut bölümleri hayati önem derecelerine göre gruplanmıştır. Eklemlerdeki dönme serbestlik dereceleri Şekil 3.11'te gösterilmiştir.



Şekil 3.11: Kullanılan sanal insan modelinin eklemlerinin serbestlik dereceleri

3.3.1 Çevrimdışı sistem

Çevrimdışı sistemde kullanılan beyin harici (external) beyindir. Bu beyin tipi, rastgele bir politika belirleyerek çalışmaya başlar. Harici beyin, eğitim süresi boyunca ajan aracılığıyla edindiği gözlemlere karşılık hareketleri bu politika ile seçer. Beyin ajandan aldığı ödüllere göre politikayı belli sıklıklarla iyileştirir. Eğitim bittiğinde ise politikanın son hali bir öğrenme dosyasına kaydedilir.

Çevrimdışı öğrenme sisteminde aşağıdaki ML-Agents ayarları kullanılmıştır. Aşağıda belirtilmeyen parametreler için öntanımlı değerler kullanılmıştır.

Gamma: Sistemin öğrenme adımlarında negatif ödül kullanılması sebebiyle, ilerideki ödüllerin öneminin azalması içinde bulunulan durumun önemini arttırmaktadır. Bu ise beynin içinde bulunulan durumdaki negatif ödülden kaçınarak gerekli hareketleri yapmamasına ve

ileride daha büyük miktarda negatif ödül almasınaa sebep olmaktadır. Bu nedenle sistemde mümkün olan en yüksek indirim çarpanı değeri olan 0.995 değeri kullanılmıştır.

Lambda: Mevcut değer tahmini ve gelecek ödüllerle belirlenecek değer tahminleri arasında bir denge olması için ortalama bir genelleştirilmiş avantaj tahmini değeri olan 0.93 değeri kullanılmıştır.

batch_size: Sürekli hareket uzayı kullanıldığı için değer 512 ile 5120 arasında olmalıdır. Fakat gradyanın sık güncellenmesi öğrenme hızını yavaşlatacağı için ortalama bir değer olarak 2048 seçilmiştir.

buffer_size: batch_size değerinin 10 katı olan 20480 kullnılmıştır. Yani politika, gradyanın her 10 kez güncellenmesinde 1 defa güncellenmektedir.

num_epoch: batch_size değeri ile orantılı olması için değer aralığının ortasındaki 5 değeri seçilmiştir.

time_horizon: Gözlem miktarı fazla olduğu için 1000 değeri uygun bulunmuştur.

max_steps: 5.0e6 değeri seçilmiştir.

beta: Daha yüksek değerleri araştırma eğilimini artırmak için maksimum 1.0e-2 değeri uygun görülmüştür

epsilon: Mevcut gradyan ile güncellenen arasındaki değişikliğinin yüksek olması için maksimum değer olan 0.3 seçilmiştir.

normalize: Gözlem uzayı karmaşık olduğu için true olar ayarlanmıştır.

num_layers: Gözlem uzayı karmaşık olduğu için maksimum değer 3 seçilmiştir.

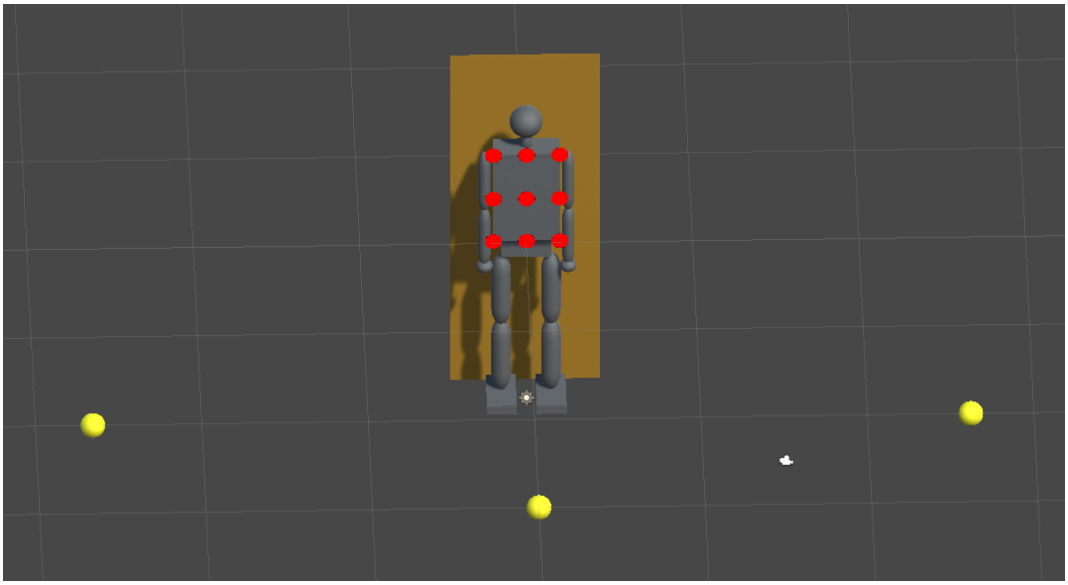
hidden_units: Gözlem uzayı ile hareket uzayı arasındaki bağıntı karmaşık olduğu için maksimum değer olan 512 seçilmiştir.

Eğitim i7-7700HQ işlemcili ve 6gb harici Nvidia GeForce GTX 1060 grafik kartlı bir bilgisayarda CUDA kullanılarak GPU üzerinde yapılmıştır. Beş milyon adımlık eğitim yaklaşık bir gün sürmüştür.

3.3.2 Çevrimiçi Sistem ve Kullanıcı Testleri

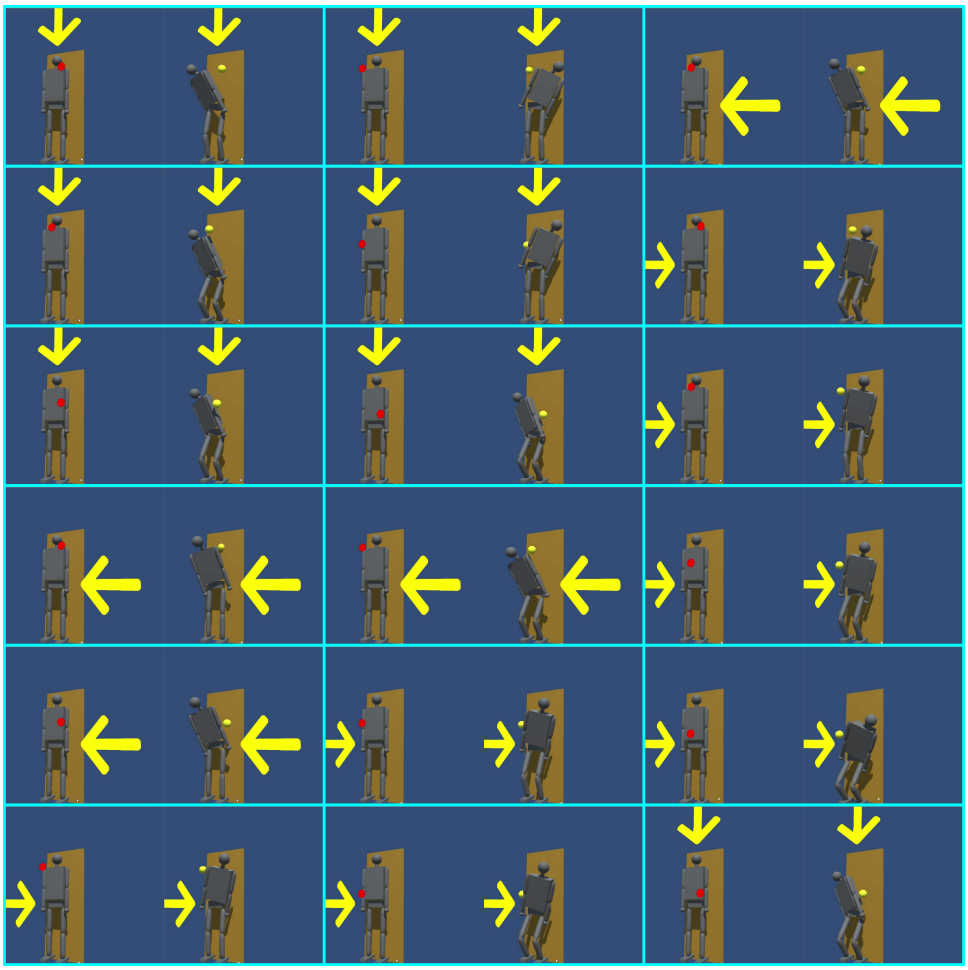
Çevrimiçi hareket üretme sisteminde kullanılan beyin dahili (internal) beyindir. Beyin kendisine ajan tarafından iletilen küre ve model eklem durumu bilgilerine karşılık hareket sinyallerini yine ajan aracılığı ile sanal insan modeline iletir. Dahili beyin gözlemlere karşılık hareket sinyallerini çevrimdışı öğrenme sisteminin çıktısı olan öğrenme dosyasından edindiği politikayı kullanarak seçer.

Çevrimiçi hareket üretme sisteminde küre çevrimdışı öğrenme sisteminde olduğu gibi, sanal insan modelinin rastgele bölgelerine otomatik olarak fırlatılmaz. Bunun yerine kürenin fırlatılacağı konum kullanıcı etkileşimi ile belirlenmektedir. Kullanıcının sanal insan modelinde bir bölgeyi işaretlemesi ile küre işaretlenen bölüme fırlatılır ve model bundan sonra kaçınma hareketlerine başlar.



Şekil 3.12: Test hareketleri için seçilmiş vücut bölgeleri ve küre başlangıç konumları.

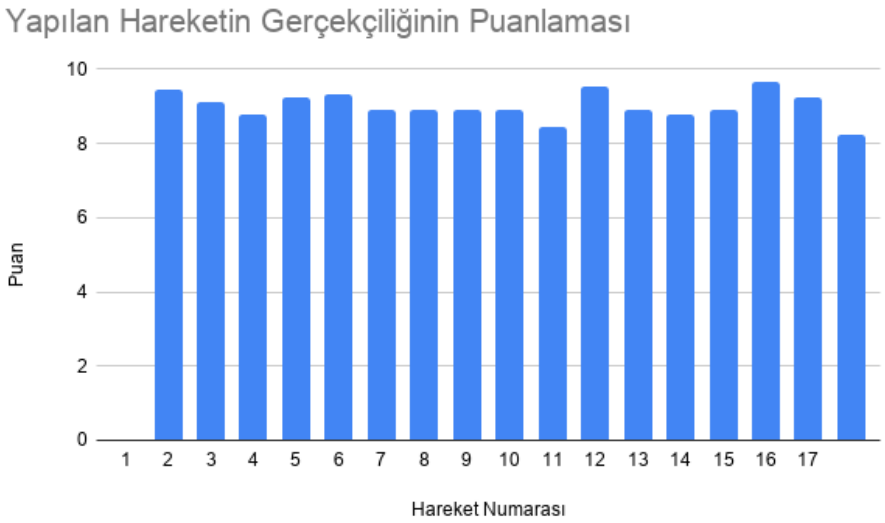
Test aşamasında çevrimiçi hareket üretme sistemi kullanılmıştır. Şekil 3.12'teki işaretli vücut bölgelerinden ve küre konumlarından birer tanesi seçilerek, küre toplam 17 farklı şekilde sanal insan modeline doğru fırlatılmıştır. Fırlatılan kürelere karşılık, sanal insan modelinin yaptığı hareketlerin videosu kaydedilmiştir. Şekil 3.13'teki görüntülerde görüldüğü gibi kaydedilen videoda her bir küre fırlatılmasından önce, kürenin fırlatılacağı vücut bölgesi işaretlenmektedir ve topun fırlatılacağı konum (sağ, sol ya da karşıdan) ok ile gösterilmektedir. Şekil 3.13'te sanal insan modelinin kaçınma hareketi gerçekleştirildikten sonraki konumu da görülmektedir.



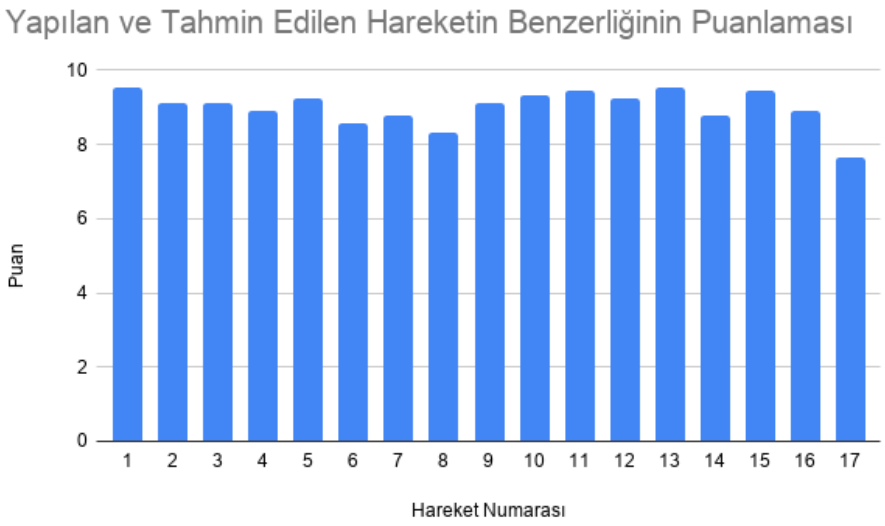
Şekil 3.13: Sanal insan modelinin kaçınma öncesi ve sonrası pozları.

Kullanıcı testleri yaşları 13 ile 70 arasında değişen 10 kullanıcı ile gerçekleştirilmiştir. Sanal insan modelinin kollarının hareket etmediği ve ayaklarının yerde sabit olduğu teste başlamadan önce, kullanıcıların kararlarının etkilenmemesi için özellikle belirtilmiştir. Kaydedilen hareketler izletilmeden önce kullanıcılardan kürelerin başlangıç konumları ve işaretli vücut bölgelerine bakarak bir kaçınma hareketi tahmini yapması istenmiştir. Daha sonra sanal insan modelinin hareketleri izletilmiştir ve yapılan hareketin kendi tahmin ettikleri harekete benzerliği ile gerçekçiliğini 10 üzerinden değerlendirmeleri söylenmiştir.

Şekil 3.14'teki grafikte görüldüğü üzere, yapılmış olan testlere göre üretilen hareketler yüzde 90 gerçekçiliktedir. Şekil 3.15'teki grafik üretilmiş olan hareketlerin test kullanıcılarının yapılmasını beklediği hareketler ile yüzde 89 oranında benzer olduğunu göstermektedir. Her iki grafik de değerlendirilecek olursa sistem bütün hareketlerde mantığı uygun hareketler üretmiştir.



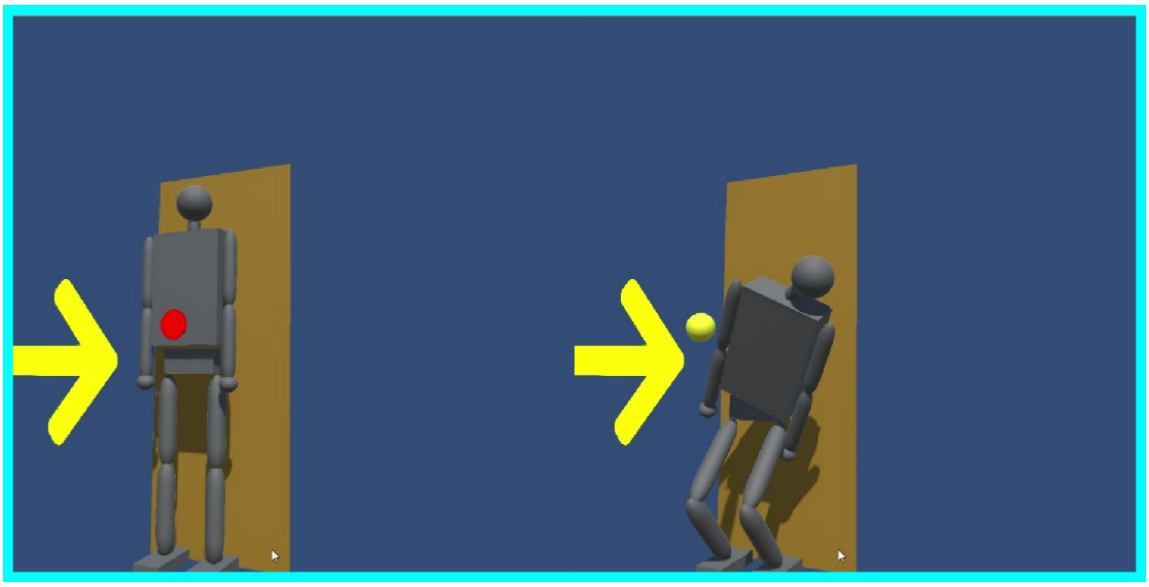
Şekil 3.14: Yapılan hareketlerin gerçekçiliklerinin değerlendirilmesi.



Şekil 3.15: Yapılan hareketlerin tahmin edilen hareketlerle benzerliğinin değerlendirilmesi.

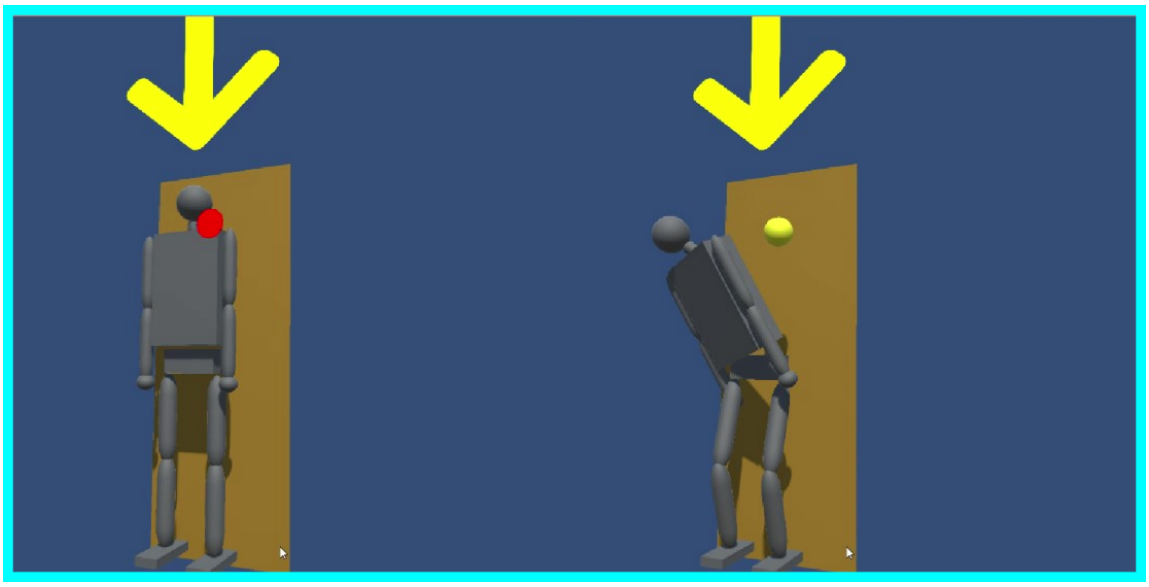
Yapılan hareketlerde hem tahmin edilenle benzerlik hem de gerçekçilik değerlendirmesinde en düşük puanı alan 17 numaralı hareket olmuştur. Bu hareket kaçınmanın en zor olduğu, kürenin sol taraftan gövdenin alt-orta kısmına fırlatılması durumunda üretilen harekettir. Burada sanal insan modelinin Şekil 3.16'de görüldüğü gibi dizlerini ve gövdesini yüksek miktarda bükmesi fiziksel olarak doğru bir hareket olsa da sıradan bir insanın yapması çok da kolay olmayacağı için testçiler böyle bir hareketi tahmin etmekte zorlanmış ve gerçekçiliğini diğer hareketlere nazaran daha düşük bulmuştur.

Her iki değerlendirmede de en yüksek puanı alan hareket ise 1 numaralı hareket olmuştur.



Şekil 3.16: Test sonucunda en düşük puanı alan 17 numaralı hareket.

Bu hareket kürenin karşıdan sanal insan modelinin sol omzuna fırlatıldığı durumda üretilen hareket olmuştur. Bunun sebebi olarak sanal insan modelinin Şekil 3.17'de gösterildiği gibi dizlerini neredeyse hiç bükmeden basit bir omuz hareketiyle sıyrılmış olması düşünülebilir.



Şekil 3.17: Test sonucunda en yüksek puanı alan 1 numaralı hareket.

4 SONUÇ

Çalışmada, pekiştirmeli öğrenme kullanılarak bir sanal insan modelinin kendisine fırlatılan kürelerden kaçınmayı öğrenmesi sağlanmıştır. Kaçınma hareketlerinde gerçekçiliğini sağlamak için sanal insan modelinin;

- Minimum efor kullanacak şekilde küçük hareketler yapmasına özen gösterilmiştir.
- Denge konumundan uzaklaşarak yerçekimine aykırı bir hareket yapması engellenmiştir.
- İnsan vücudunun limitlerine uygun şekilde gerçekçi hız ve eklem açılarıyla hareket etmesine dikkat edilmiştir.

Çalışmada 17 değer gözlemlenmekte ve bunlara karşılık 9 hareket değeri dönmektedir. Gözlem ve hareket sayısının fazla olması ve de sürekli hareket uzayı kullanılması doğru bir politika belirlenmesi için gerekli adım sayısının da fazla olmasına sebep olmaktadır. Ayrıca kürenin başlangıç konumu ve fırlatılacağı vücut bölgesi, öğrenme boyunca geniş aralıklardan rastgele seçilmektedir. Bu nedenle öğrenmenin tamamlanması için gereken süre biraz uzundur.

Elde edilen sistem başarılı şekilde kaçınma hareketleri üretse de kollar mevcut durumda hareket etmemektedir. Ayrıca sanal insan modelinin ayakları yerde sabittir. Kolların kürenin geliş yönüne göre daha önemli vücut bölgelerine siper edilmesi ve gerektiğinde küçük adımlarla kaçınma hareketleri yapılması gerçekçiliği arttırabilir.

Çevrimdışı öğrenme sisteminde elde edilen öğrenme dosyası sadece çalışmada kullanılan sanal insan modeli ile birebir bir aynı modeller için başarılı kaçınma hareketleri üretebilecek bir politika içermektedir. Bu nedenle modelin herhangi bir vücut bölümünün boyutunun değişmesi çevrimiçi hareket üretme sisteminin doğru çalışmamasına neden olacaktır.

Çevrimdışı öğrenme sistemi ile öğrenilen politika sayesinde model kendisine fırlatılan kürelerden kaçabilmektedir. Bu politika sayesinde, yumrukları ya da tekmeleri küre şeklinde olan sanal insan modellerinin karşılıklı dövüşmelerinde kaçınma hareketleri yeni bir eğitime gerek duymadan mevcut öğrenme dosyası kullanılarak üretilebilir.

Gelecekte mevcut sistem üzerinde bazı geliştirmeler planlanmaktadır. Bunlardan en önemlisi sanal insan modelinin kolları ile daha önemli vücut bölümlerini korumayı öğrenmesidir. Daha sonra ise bazı ayak oyunları ile ufak adımlar atarak kaçınabilmesi düşünülmektedir. Son olarak modelin farklı geometrideki cisimlerden kaçınmayı da öğrenmesi ile sistem dövüş oyunlarında kullanılmaya hazır olacaktır.

Kaynaklar

[1] Mao, H., Alizadeh, M., Menache, I., Kandula, S., Resource management with deep reinforcement learning, *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, ACM, New York, NY, USA, **2016**, HotNets '16, 50–56. URL http://doi.acm.org/10.1145/3005745.3005750.

[2] Arel, I., Liu, C., Urbanik, T., Kohls, A.G., Reinforcement learning-based multi-agent system for network traffic signal control, *IET Intelligent Transport Systems*, 4(2), 128–135, **2010**.

[3] Long, P., Fanl, T., Liao, X., Liu, W., Zhang, H., Pan, J., Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning, *2018 IEEE International Conference on Robotics and Automation (ICRA)*, **2018**. URL http://dx.doi.org/10.1109/icra.2018.8461113.

[4] Lötzsch, W., Using deep reinforcement learning for the continuous control of robotic arms, **2018**.

[5] Bu, X., Rao, J., Xu, C.Z., A reinforcement learning approach to online web systems auto-configuration, *Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems*, IEEE Computer Society, Washington, DC, USA, **2009**, ICDCS '09, 2–11. URL https://doi.org/10.1109/ICDCS.2009.76.

[6] Zheng, G., Zhang, F., Zheng, Z., Xiang, Y., Jing Yuan, N., Xie, X., Li, Z., Drn: A deep reinforcement learning framework for news recommendation, **2018**, 167–176.

[7] Jin, J., Song, C., Li, H., Gai, K., Wang, J., Zhang, W., Real-time bidding with multi-agent reinforcement learning in display advertising, *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ACM, New York, NY, USA, **2018**, CIKM '18, 2193–2201. URL http://doi.acm.org/10.1145/3269206.3272021.

[8] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D., Mastering the game of go with deep neural networks

and tree search, *Nature*, 529, 484–503, **2016**. URL http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

[9] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L.R., Lai, M., Bolton, A., Chen, Y., Lillicrap, T.P., Hui, F.F.C., Sifre, L., van den Driessche, G., Graepel, T., Hassabis, D., Mastering the game of go without human knowledge, *Nature*, 550, 354–359, **2017**.

[10] OpenAI, Openai five, https://blog.openai.com/openai-five/, **2018**.

[11] Sutton, R.S., Barto, A.G., *Reinforcement Learning: An Introduction*, The MIT Press, second edition, **2018**. URL http://incompleteideas.net/book/the-book-2nd.html.

[12] Schulman, J., Levine, S., Moritz, P., Jordan, M.I., Abbeel, P., Trust region policy optimization, *CoRR*, abs/1502.05477, **2015**. URL http://arxiv.org/abs/1502.05477.

[13] Hui, J., Deep reinforcement learning series, https://medium.com/@jonathan_hui/rl-trust-region-policy-optimization-trpo-explained-a6

[14] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., Proximal policy optimization algorithms, **2017**.

[15] Hui, J., Deep reinforcement learning series, https://medium.com/@jonathan_hui/rl-proximal-policy-optimization-ppo-explained-77f014e

[16] Unity, Ml-agents toolkit overview, https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md.

[17] Unity, Training with proximal policy optimization, https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Training-PPO.md.

[18] B. Zordan, V., Macchietto, A., Medina, J., Soriano, M., Wu, C.C., Metoyer, R., Rose, R., Anticipation from example, **2007**, 81–84.

[19] Shum, H.P.H., Komura, T., Yamazaki, S., Simulating competitive interactions using singly captured motions, *Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology*, ACM, New York, NY, USA, **2007**, VRST '07, 65–72. URL http://doi.acm.org/10.1145/1315184.1315194.

[20] Oshita, M., Masaoka, N., Generating avoidance motion using motion graph, **2011**, 120–131.

[21] Oshita, M., Lattice-guided human motion deformation for collision avoidance, *Proceedings of the Tenth International Conference on Motion in Games*, ACM, New York, NY, USA, **2017**, MIG '17, 12:1–12:6. URL http://doi.acm.org/10.1145/3136457.3136475.

[22] Tan, J., Gu, Y., Liu, C.K., Turk, G., Learning bicycle stunts, *ACM Trans Graph*, 33(4), 50:1–50:12, **2014**. URL http://doi.acm.org/10.1145/2601097.2601121.

[23] Çimen, G., Kavafoglu, Z., Kavafoglu, E., Çapın, T., Gürçay, H., Skill learning based catching motion control, *Computer Animation and Virtual Worlds*, 26, **2015**.

[24] Peng, X.B., Berseth, G., van de Panne, M., Dynamic terrain traversal skills using reinforcement learning, *ACM Trans Graph*, 34(4), 80:1–80:11, **2015**. URL http://doi.acm.org/10.1145/2766910.

[25] Peng, X.B., Berseth, G., van de Panne, M., Terrain-adaptive locomotion skills using deep reinforcement learning, *ACM Trans Graph*, 35(4), 81:1–81:12, **2016**. URL http://doi.acm.org/10.1145/2897824.2925881.

[26] Peng, X.B., Berseth, G., Yin, K., Van De Panne, M., Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning, *ACM Trans Graph*, 36(4), 41:1–41:13, **2017**. URL http://doi.acm.org/10.1145/3072959.3073602.

[27] Peng, X.B., Abbeel, P., Levine, S., van de Panne, M., Deepmimic: Example-guided deep reinforcement learning of physics-based character skills, *ACM Trans Graph*, 37(4), 143:1–143:14, **2018**. URL http://doi.acm.org/10.1145/3197517.3201311.

[28] Peng, X.B., Kanazawa, A., Malik, J., Abbeel, P., Levine, S., Sfv: Reinforcement learning of physical skills from videos, *ACM Trans Graph*, 37(6), **2018**.

[29] Chentanez, N., Muller, M., Macklin, M., Makoviychuk, V., Jeschke, S., Physics-based motion capture imitation with deep reinforcement learning, *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*, ACM, New York, NY, USA, **2018**, MIG '18, 1:1–1:10. URL `http://doi.acm.org/10.1145/3274247.3274506`.

[30] Chapman, A., *Biomechanical Analysis of Fundamental Human Movements-Google*, Human Kinetics. URL `https://books.google.com.tr/books?id=YqxDzfyr3k4C`.